

20MCA241 DATA SCIENCE LAB

Lab Report Submitted By

ASHA S

Reg. No.: AJC20MCA-2026

In Partial fulfillment for the Award of the Degree Of

**MASTER OF COMPUTER APPLICATIONS (2 Year)
(MCA)**

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



**AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY**

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

2020-2022

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Lab report, “**20MCA241 DATA SCIENCE LAB**” is the bonafide work of **ASHA S (Reg.No:AJC20MCA-2026)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

Ms. Shelly Shiju George

Lab In-Charge

CONTENT

Sl.No	Content	Date	Page No
1	Perform all matrix operation using python	24/11/2021	1
2	Program to perform SVD using python	01/12/2021	4
3	Program to implement k-NN Classification using any standard dataset available in the public domain and find the accuracy of the algorithm using in build function	01/12/2021	5
4	Program to implement k-NN Classification using any random dataset without using in-build functions	01/12/2021	6
5	Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm	08/12/2021	8
6	Program to implement linear and multiple regression techniques using any standard dataset available in the public domain	08/01/2022	11
7	Program to implement Linear and Multiple regression techniques using any standard dataset available in public domain and evaluate its performance	08/01/2022	12
8	Program to implement Linear and Multiple regression techniques using cars dataset available in public domain and evaluate its performance	15/01/2022	14
9	Program to implement multiple linear regression techniques using Boston dataset available in the public domain and evaluate its performance and plotting graph	15/01/2022	15
10	Program to implement decision tree using any standard dataset available in the public domain and find the accuracy of the algorithm	22/12/2021	17
11	Program to implement K-Means clustering technique using any standard dataset available in the public domain	05/01/2022	20
12	Program to implement K-Means clustering technique using any standard dataset available in the public domain	05/01/2022	23
13	Programs on convolutional neural network to classify images from any standard dataset in the public domain	02/02/2022	25

14	Program to implement a simple web crawler using python	16/02/2022	30
15	Program to implement a simple web crawler using python	16/02/2022	32
16	Program to implement scrap of any website	16/02/2022	34
17	Program for Natural Language Processing which performs n-grams	16/02/2022	36
18	Program for Natural Language Processing which performs n-grams (Using in built functions)	16/02/2022	37
19	Program for Natural Language Processing which performs speech tagging	16/02/2022	38
20	Python program which performs Natural language processing which perform Chunking	23/02/2022	40

PROGRAM NO : 01**Date:24/11/2021****AIM : Perform all matrix operation using python.****PROGRAM CODE**

```

import numpy as np
import random
def PrintMatrix(matrix_in):
    for x in range(0, matrix_in.shape[0]):
        for y in range(0, matrix_in.shape[1]):
            print("%d\t" % (matrix_in[x][y]), end="")
            if (y % 3 > 1):
                print("\n")
def FillMatrix(matrix_in):
    for x in range(0, matrix_in.shape[0]):
        for y in range(0, matrix_in.shape[1]):
            matrix_in[x][y] = random.randrange(2, 10) + 2
matrix1 = np.ndarray((3,3))
matrix2 = np.ndarray((3,3))
FillMatrix(matrix1)
FillMatrix(matrix2)
add_results = np.add(matrix1,matrix2)
sub_results=np.subtract(matrix1,matrix2)
mult_results=np.multiply(matrix1,matrix2)
div_results=np.divide(matrix1,matrix2)
dot_results=np.dot(matrix1,matrix2)
sqrt1_results=np.sqrt(matrix1)
sqrt2_results=np.sqrt(matrix2)
trans_results=add_results.T
print("Matrix1:")
PrintMatrix(matrix1)
print("Matrix2:")
PrintMatrix(matrix2)
print("Adding")

```

```

PrintMatrix(add_results)
print("Subtraction")
PrintMatrix(sub_results)
print("Multiplication")
PrintMatrix(mult_results)
print("Dot Operation")
PrintMatrix(dot_results)
print("squareroot Operation")
print("matrix 1")
PrintMatrix(sqrt1_results)
print("matrix 2")
PrintMatrix(sqrt2_results)
print("Transpose")
PrintMatrix(trans_results)

```

OUTPUT

Matrix1:

4	4	11
6	4	6
9	11	5

Matrix2:

8	10	10
11	9	8
8	11	10

Adding

12	14	21
17	13	14
17	22	15

Subtraction

-4	-6	1
-5	-5	-2
1	0	-5

Multiplication

32	40	110
66	36	48
72	121	50

Dot Operation

164	197	182
140	162	152
233	244	228

Squareroot Operation matrix 1

2	2	3
2	2	2
3	3	2

matrix 2

2	3	3
3	3	2
2	3	3

Transpose

12	17	17
14	13	22
21	14	15

Process finished with exit code 0

PROGRAM NO : 02**Date :01/12/2021****AIM: Program to perform SVD (Singular value Decomposition) using Python.****PROGRAM CODE**

```

from numpy import array
from print(U)
print(S)
print(VT)scipy.linalg import svd
A=([[1,2,3,4,5],[4,5,6,44,55],[7,8,9,55,77]])
print(A)
U,S,VT= svd(A)

```

OUTPUT

```

C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/svd.py
[[1, 2, 3, 4, 5], [4, 5, 6, 44, 55], [7, 8, 9, 55, 77]]
[[-0.0573526  0.47097924 -0.88027793]
 [-0.59475368 -0.72430716 -0.34877958]
 [-0.80185955  0.50354512  0.32165754]]
[119.25381152  3.53502671  2.65179639]
[[-0.06749792 -0.07969012 -0.09188231 -0.59118318 -0.79445176]
 [ 0.31076609  0.38154835  0.45233062 -0.64797719  0.36519566]
 [-0.00897259 -0.35115571 -0.69333884 -0.44356672  0.44624991]]

```

```

[119.25381152  3.53502671  2.65179639]
[[-0.06749792 -0.07969012 -0.09188231 -0.59118318 -0.79445176]
 [ 0.31076609  0.38154835  0.45233062 -0.64797719  0.36519566]
 [-0.00897259 -0.35115571 -0.69333884 -0.44356672  0.44624991]
 [-0.13719266  0.85125735 -0.50040838  0.05438524 -0.05632757]
 [-0.9380651  0.01099681  0.23627814 -0.17583749  0.1821174 ]]

```

```

Process finished with exit code 0

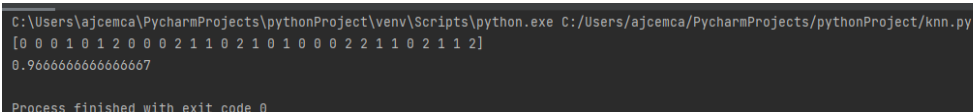
```


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

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

PROGRAM CODE

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
irisData=load_iris()
x=irisData.data
y=irisData.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=80)
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
print(knn.predict(x_test))
z=knn.predict(x_test)
i=accuracy_score(y_test,z)
print(i)
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/knn.py
[0 0 0 1 0 1 2 0 0 0 2 1 1 0 2 1 0 1 0 0 0 2 2 1 1 0 2 1 1 2]
0.9666666666666667

Process finished with exit code 0
```

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

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

PROGRAM CODE

```
from math import sqrt

def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1) - 1):
        distance += (row1[i] - row2[i]) ** 2
    return sqrt(distance)

def get_neighbors(train, test_row, num_neighbors):
    distances = list()
    for train_row in train:
        dist = euclidean_distance(test_row, train_row)
        distances.append((train_row, dist))
    distances.sort(key=lambda tup: tup[1])
    neighbors = list()
    for i in range(num_neighbors):
        neighbors.append(distances[i][0])
    return neighbors

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

```
dataset = [[2.7810836, 2.550537003, 0],  
           [1.465489372, 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[2], 8)  
print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/knnnotorg.py  
Expected 0, Got 0.  
  
Process finished with exit code 0
```

PROGRAM NO : 05**Date :08/12/2021**

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

PROGRAM CODE

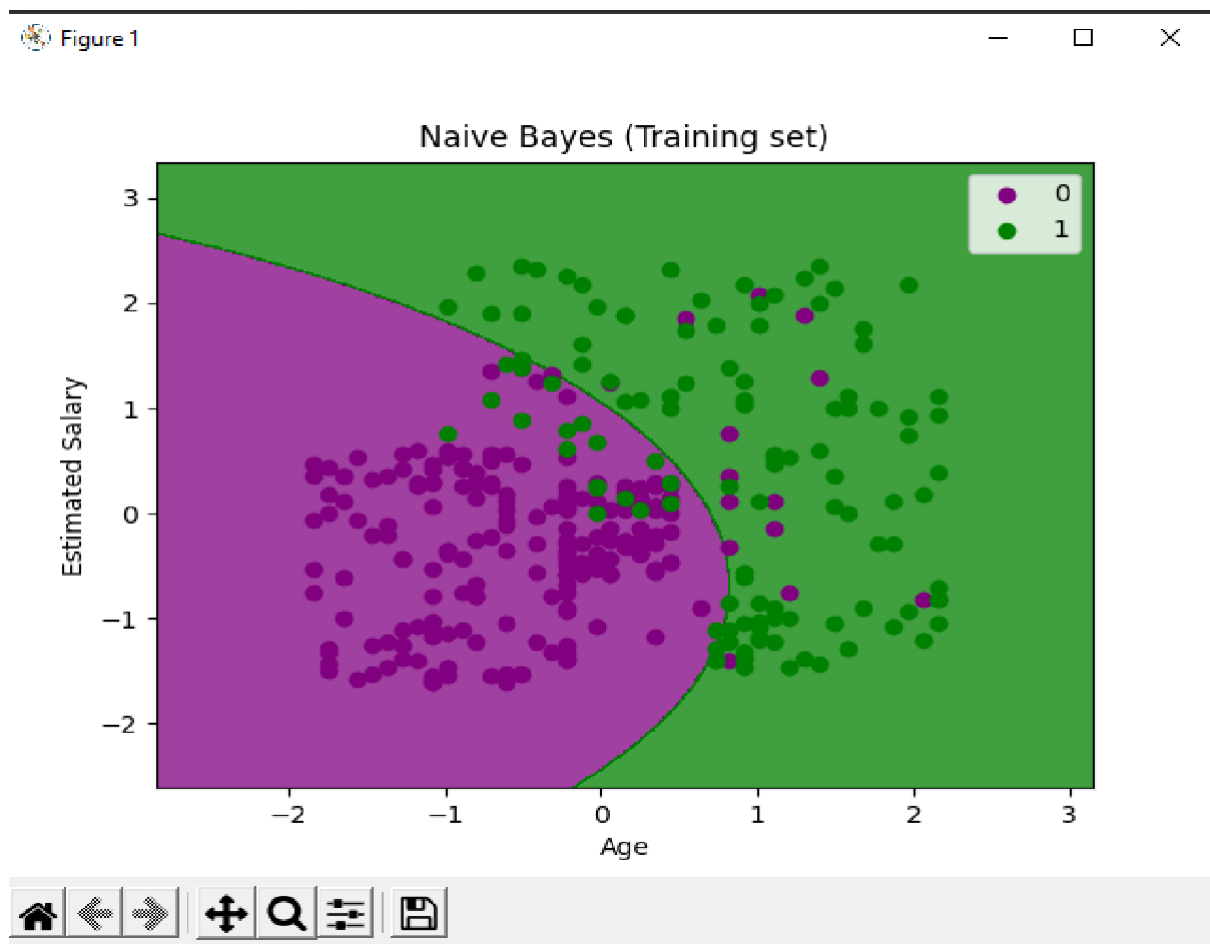
```
import pandas as pd
dataset = pd.read_csv('Social_Network_Ads.csv')
x = dataset.iloc[:, [2,3]].values
y = dataset.iloc[:, -1].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=10)
from sklearn.preprocessing 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



```
[0011010100001110000100011001100001101
1000000001000011000101101011101000000
000011]
```

Accuracy 91.25

PROGRAM NO : 06**Date :08/12/2021**

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

PROGRAM CODE

```
import numpy as np
from sklearn.linear_model import LinearRegression
x=np.array([5,15,25,35,45,55]).reshape((-1,1))
y=np.array([5,20,14,32,22,38])
print(x)
print(y)
model=LinearRegression()
model.fit(x,y)
r_sq=model.score(x,y)
print("coefficient of determinator",r_sq)
print("intersect",model.coef_)
y_pred=model.predict(x)
print("predicted response",y_pred)
```

OUTPUT

```
[45]
[55]]
[ 5 20 14 32 22 38]
coefficient of determinator 0.7158756137479542
intersect [0.54]
predicted response [ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]

Process finished with exit code 0
```

PROGRAM NO : 07**Date :08/12/2021**

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

PROGRAM CODE

```
import numpy as np
import matplotlib.pyplot as plt
def estimate_coef(x, y):
# number of observations/points
n = np.size(x)
# mean of x and y vector
m_x = np.mean(x)
m_y = 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 = 'g')
# putting labels
plt.xlabel('x')
plt.ylabel('y')
# function to show plot
```

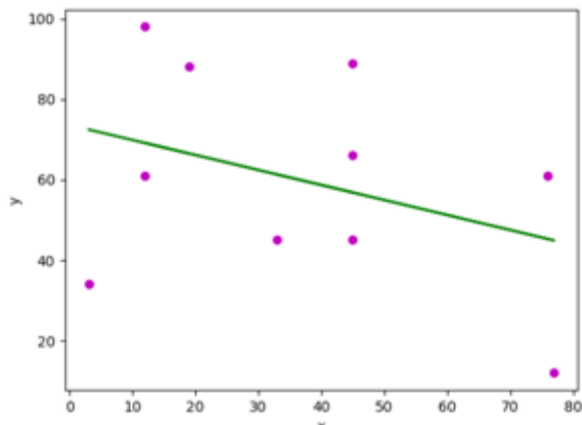


```
plt.show()

def main():
    # observations / data
    x = np.array([12,45,45,77,3,19,12,45,33,76])
    y = np.array([98,45,66,12,34,88,61,89,45,61])
    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {} \
\nb_1 = {}".format(b[0], b[1]))
    # plotting regression line
    plot_regression_line(x, y, b)
if __name__ == "__main__":
    main()
```

OUTPUT

OUTPUT



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

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

PROGRAM CODE

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

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

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

PROGRAM CODE

```
import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear_model, metrics

from sklearn.metrics import r2_score

boston = datasets.load_boston(return_X_y=False)

X = boston.data

y = boston.target

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

reg = linear_model.LinearRegression()

reg.fit(X_train, y_train)

V=reg.predict(X_test)

result=r2_score(y_test, V)

print("accuracy :", result)

print('Coefficients: ', reg.coef_)

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

OUTPUT

accuracy : 0.7209056672661767

Coefficients: [-8.95714048e-02 6.73132853e-02 5.04649248e-02 2.18579583e+00

-1.72053975e+01 3.63606995e+00 2.05579939e-03 -1.36602886e+00

2.89576718e-01 -1.22700072e-02 -8.34881849e-01 9.40360790e-03

-5.04008320e-01]

Variance score:0.720905667266176

Process finished with exit code 0

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

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

PROGRAM CODE

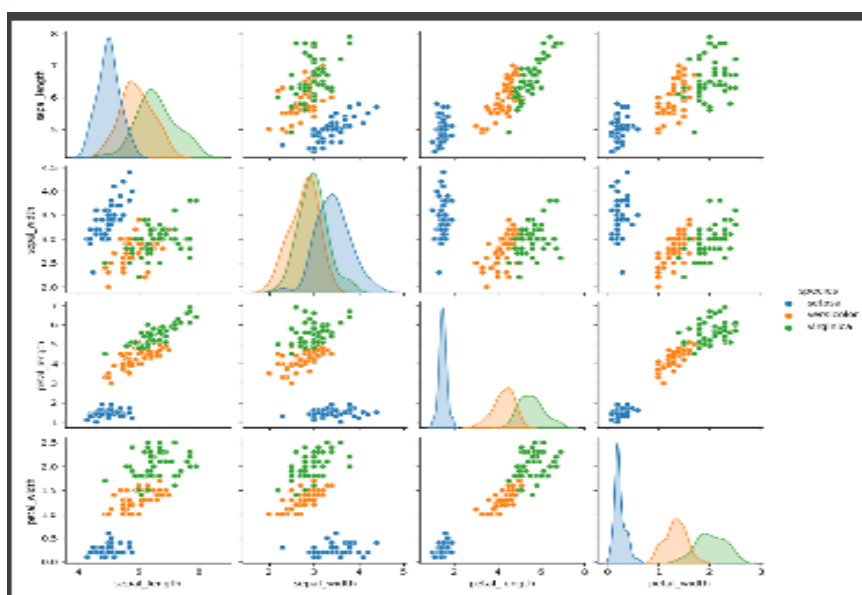
```
Import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import plot_tree
df=sns.load_dataset('iris')
print(df.head())
print(df.info())
df.isnull().any()
print(df.shape)
sns.pairplot(data=df, hue='species')
plt.savefig("pne.png")
sns.heatmap(df.corr())
plt.savefig("next.png")
target =df['species']
df1 = df.copy()
df1 = df1.drop('species', axis=1)
print(df1.shape)
print(df1.head())
x=df1
print(target)
le = LabelEncoder()
target = le.fit_transform(target)
print(target)
y= target
```

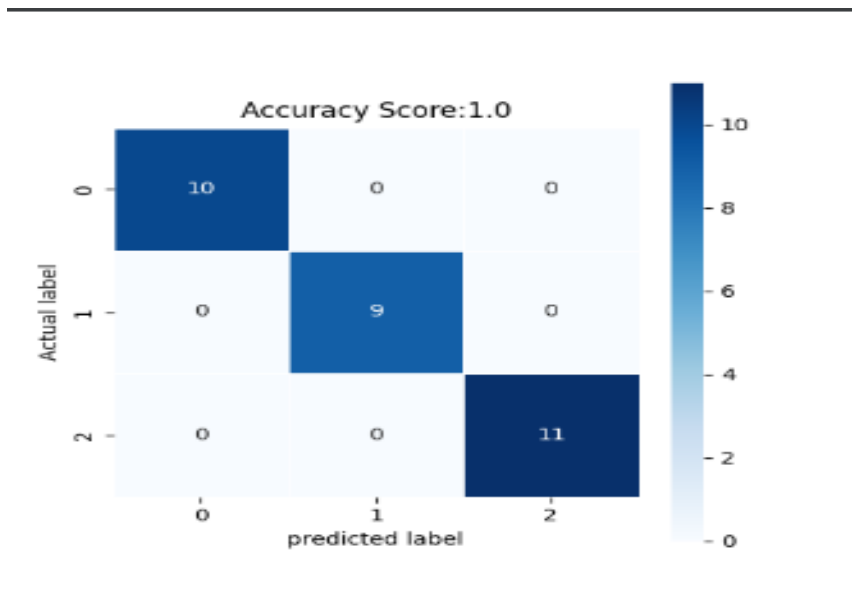
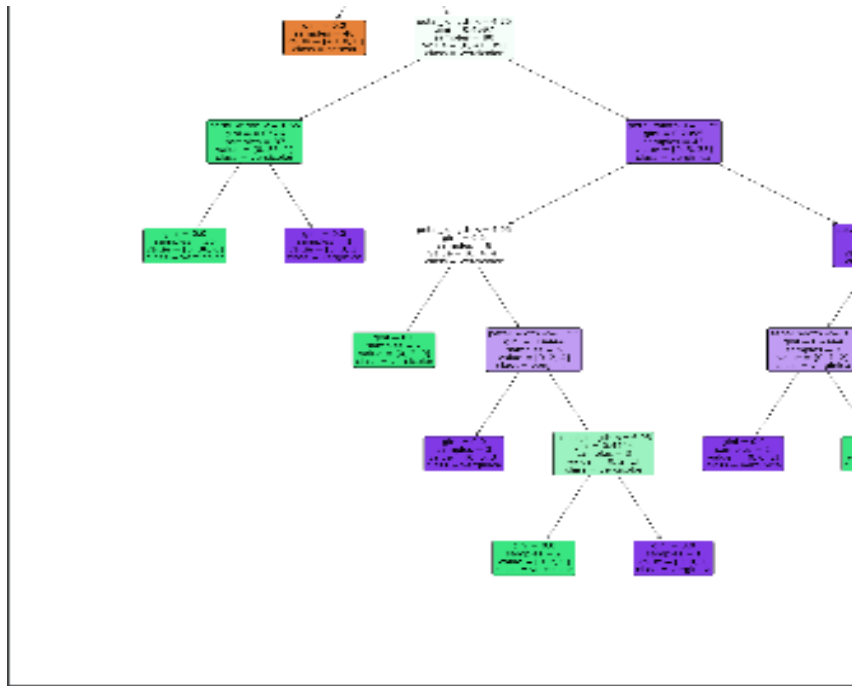
```

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

```

OUTPUT





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

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

PROGRAM CODE

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

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

x=dataset.iloc[:,[3,4]].values

print(x)

from sklearn.cluster import KMeans

wcss_list=[]

for i in range(1,11):

    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)

    kmeans.fit(x)

    wcss_list.append(kmeans.inertia_)

mtp.plot(range(1,11), wcss_list)

mtp.title('The Elbow Method Graph')

mtp.xlabel('Number of clusters(k)')

mtp.ylabel('wcss_list')

mtp.show()
```



```
kmeans=KMeans(n_clusters=5,init='k-means++',random_state=42)

y_predict=kmeans.fit_predict(x)

print('predict=',y_predict)

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

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

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

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

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

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

mtp.title('Clusters of Customer')

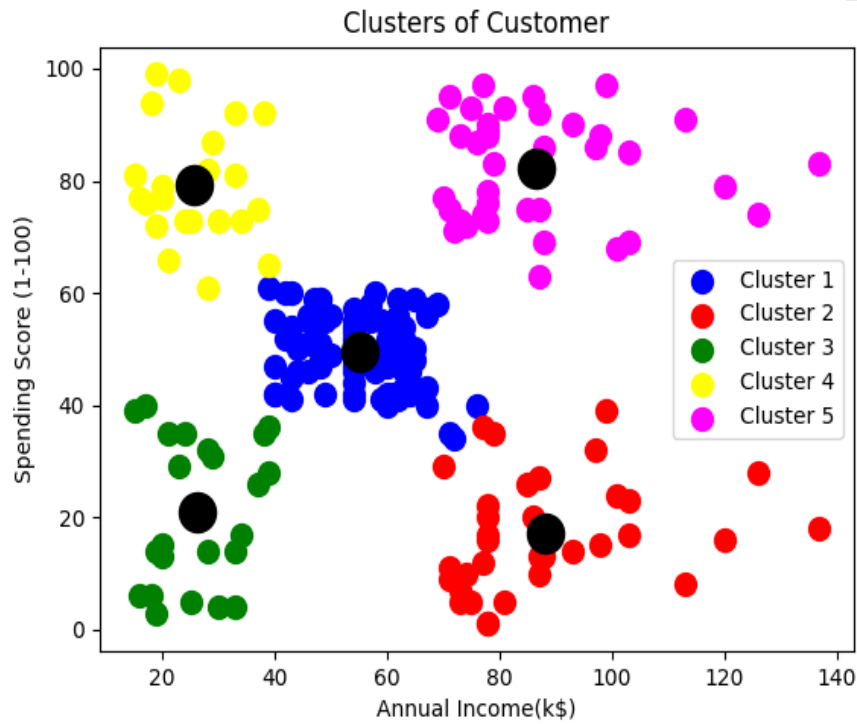
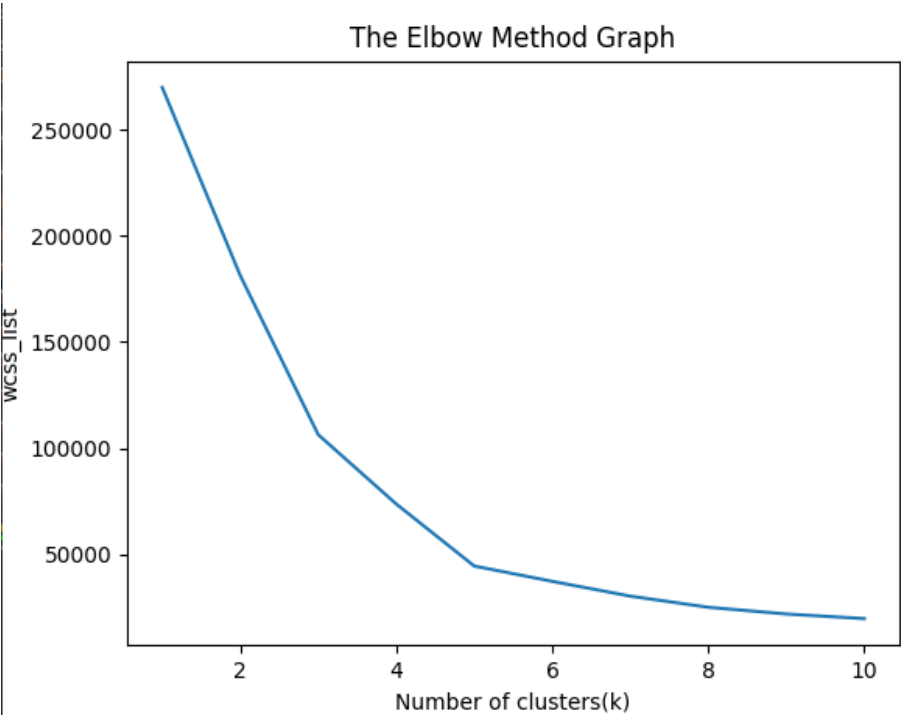
mtp.xlabel('Annual Income(k$)')

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

mtp.legend();

mtp.show()
```

OUTPUT



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

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

PROGRAM CODE

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

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

x=dataset.iloc[:,[1,2]].values

print(x)

from sklearn.cluster import KMeans

wcss_list=[]

for i in range(1,11):

    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)

    kmeans.fit(x)

    wcss_list.append(kmeans.inertia_)

mtp.plot(range(1,11), wcss_list)

mtp.title('The Elbow Method Graph')

mtp.xlabel('Number of clusters(k)')

mtp.ylabel('wcss_list')

mtp.show()

kmeans=KMeans(n_clusters=3,init='k-means++',random_state=42)

y_predict=kmeans.fit_predict(x)

print('predict=',y_predict)
```

```

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

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

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

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

mtp.title('Clusters of world Country')

mtp.xlabel('latitude')

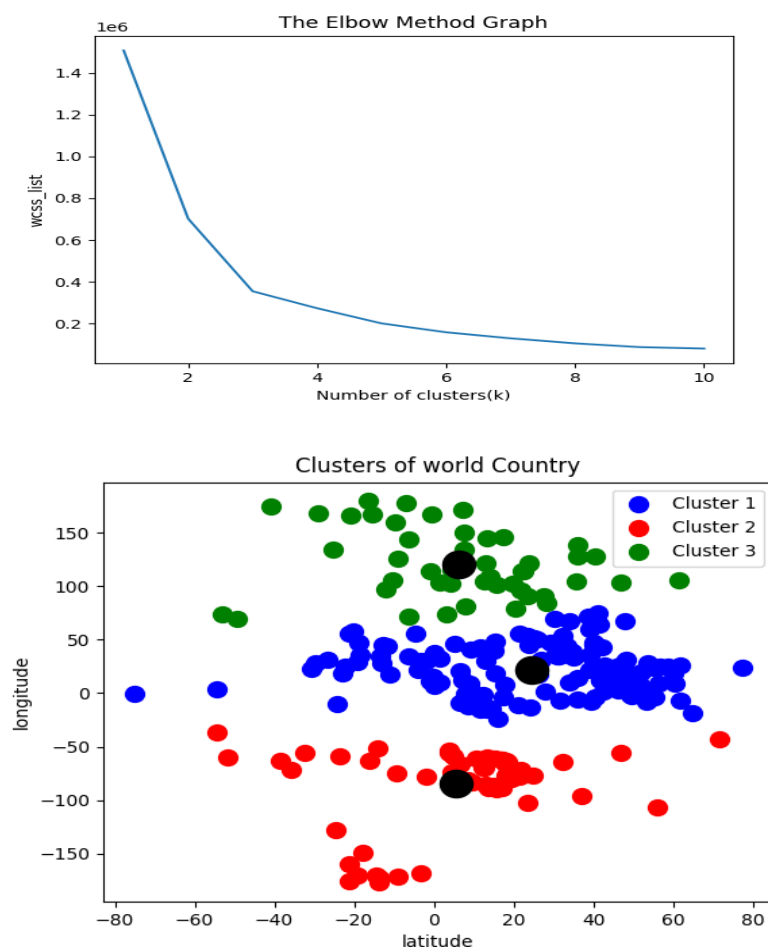
mtp.ylabel('longitude')

mtp.legend();

mtp.show()

```

OUTPUT



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

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

PROGRAM CODE

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow import keras

np.random.seed(42)

# tf.set.random. seed(42)

fashion_mnist = keras.datasets.fashion_mnist

(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

print(X_train.shape, X_test.shape)

X_train = X_train / 255.0

X_test = X_test / 255.0

plt.imshow(X_train[1], cmap='binary')

plt.show()

np.unique(y_test)

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

n_rows = 5

n_cols = 10

plt.figure(figsize=(n_cols * 1.4, n_rows * 1.6))
```

```

for row in range(n_rows):
    for col in range(n_cols):
        index = n_cols * row + col

        plt.subplot(n_rows, n_cols, index + 1)

        plt.imshow(X_train[index], cmap='binary', interpolation='nearest')

        plt.axis('off')

        plt.title(class_names[y_train[index]])

plt.show()

model_CNN = keras.models.Sequential()

model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size=7, padding='same',
activation='relu', input_shape=[28, 28, 1]))

model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))

model_CNN.add(keras.layers.Conv2D(filters=64, kernel_size=3, padding='same',
activation='relu'))

model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))

model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size=3, padding='same',
activation='relu'))

model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))

model_CNN.summary()

model_CNN.add(keras.layers.Flatten())

model_CNN.add(keras.layers.Dense(units=128, activation='relu'))

model_CNN.add(keras.layers.Dense(units=64, activation='relu'))

model_CNN.add(keras.layers.Dense(units=10, activation='softmax'))

model_CNN.summary()

```

```
model_CNN.compile(loss='sparse_categorical_crossentropy', optimizer='adam',  
metrics=['accuracy'])  
  
X_train = X_train[..., np.newaxis]  
  
X_test = X_test[..., np.newaxis]  
  
history_CNN = model_CNN.fit(X_train, y_train, epochs=2, validation_split=0.1)  
  
pd.DataFrame(history_CNN.history).plot()  
  
plt.grid(True)  
  
plt.xlabel('epochs')  
  
plt.ylabel('loss/accuracy')  
  
plt.title('Training and validation plot')  
  
plt.show()  
  
test_loss, test_accuracy = model_CNN.evaluate(X_test, y_test)  
  
print(' Test Loss :{ }, Test Accuracy : { }'.format(test_loss, test_accuracy))
```

OUTPUT

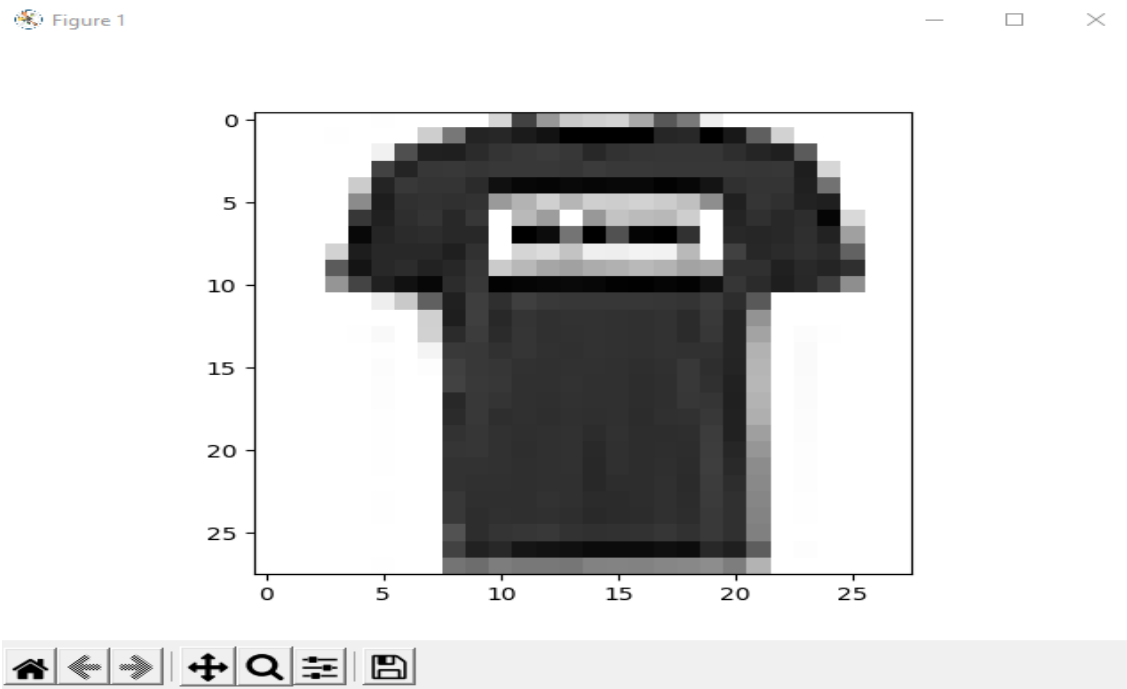
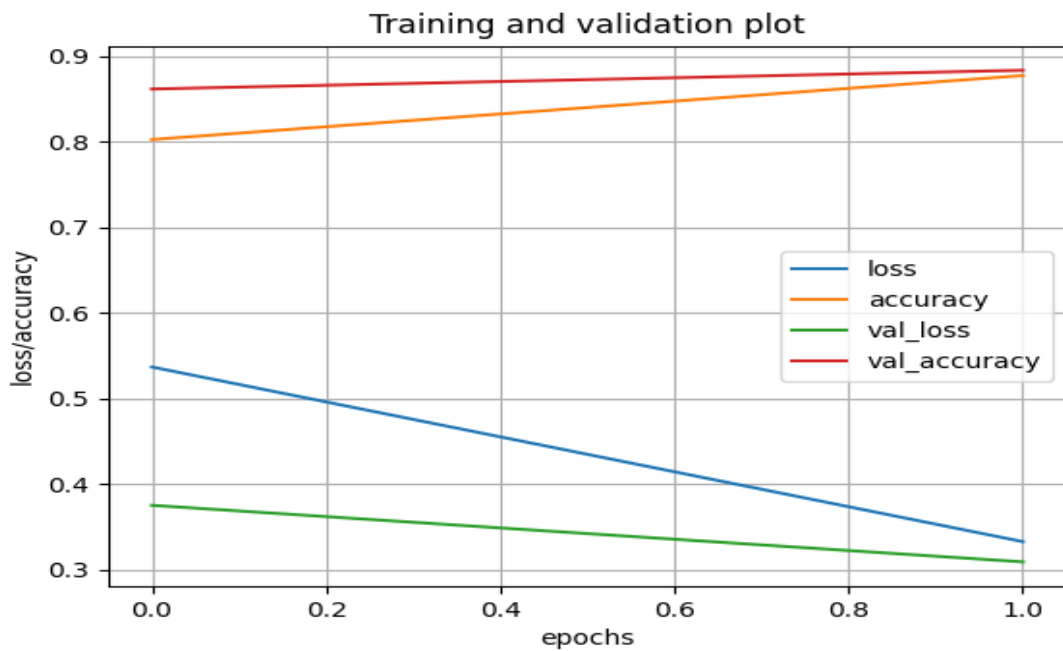


Figure 1



```
conv2d_2 (Conv2D)      (None, 7, 7, 32)      18464
max_pooling2d_2 (MaxPooling 2D)      (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 [=====] - 104s 61ms/step - loss: 0.5369 - accuracy: 0.8024 - val_loss: 0.3755 - val_accuracy: 0.8613
Epoch 2/2
1688/1688 [=====] - 103s 61ms/step - loss: 0.3332 - accuracy: 0.8770 - val_loss: 0.3096 - val_accuracy: 0.8832
```

PROGRAM NO : 14**Date :16/02/2022****AIM : Program to implement a simple web crawler using python.****PROGRAM CODE**

```
import requests

import lxml

from bs4 import BeautifulSoup

#import beautifulsoup4

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

headers = { 'User-Agents' : 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 QIHU 360SE' }

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

movies_list = []

soup = BeautifulSoup(f.content, 'html.parser')

movies = soup.find('table', {'class': 'table'}) .find_all('a')

print(movies)

num = 0

for anchor in movies:

    urls = 'https://www.rottentomatoes.com' + anchor['href']

    movies_list.append(urls)

print(movies_list)

num +=1

movie_url=urls

#movie_url=movies_lst

movie_f=requests.get(movie_url,headers=headers)
```

```

movie_soup=BeautifulSoup(movie_f.content,'lxml')

movie_content=movie_soup.find('div',{

    'class':'movie_synopsis clamp clamp-6 js-clamp'

}))

print(num,urls,'\n','Movie:' + anchor.string.strip())

print('Movie info:' + movie_content.string.strip())

```

OUTPUT

```

webcrawler
C:\Users\ajcemca\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject2/webcrawler.py
[<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">

```

```

webcrawler
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/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_wizard_of_oz_1939',
1 https://www.rottentomatoes.com/m/the_battle_of_algiers
Movie:The Battle of Algiers (La Battaglia di Algeri) (1967)
Movie info:Paratrooper commander Colonel Mathieu (Jean Martin), a former French Resistance fighter during World War II, is sent to 1950s Algeria to reinforce
Process finished with exit code 0

```

PROGRAM NO : 15**Date :16/02/2022****AIM : Program to implement a simple web crawler using python.****PROGRAM CODE**

```
from bs4 import BeautifulSoup

import requests

pages_crawled =[ ]

def crawler(url):

    page =requests.get(url)

    soup=BeautifulSoup(page.text,'html.parser')

    links=soup.find_all('a')

    for link in links:

        if 'href' in link.attrs:

            if link['href'].startswith('/wiki') and ':' not in link['href']:

                if link['href'] not in pages_crawled:

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

                    pages_crawled.append(link['href'])

                    try:

                        with open('data.csv','a') as file:

                            file.write(f'{soup.title.text}:{link["href"]}\n')

                            crawler(new_link)

                    except:

                        continue

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

OUTPUT



```
Wikipedia, the free encyclopedia; Main Page; /wiki/Wikipedia
Wikipedia, the free encyclopedia; Main Page; /wiki/Free_content
Wikipedia, the free encyclopedia; Main Page; /wiki/Encyclopedia
Wikipedia, the free encyclopedia; Main Page; /wiki/English_language
Wikipedia, the free encyclopedia; Main Page; /wiki/SS_Choctaw
Wikipedia, the free encyclopedia; Main Page; /wiki/Cargo_ship
Wikipedia, the free encyclopedia; Main Page; /wiki/Great_Lakes
Wikipedia, the free encyclopedia; Main Page; /wiki/Lake_freighter
Wikipedia, the free encyclopedia; Main Page; /wiki/Whaleback
Wikipedia, the free encyclopedia; Main Page; /wiki/Alexander_McDougall_(ship_designer)
Wikipedia, the free encyclopedia; Main Page; /wiki/American_Ship_Building_Company
Wikipedia, the free encyclopedia; Main Page; /wiki/Cleveland
Wikipedia, the free encyclopedia; Main Page; /wiki/Michigan
Wikipedia, the free encyclopedia; Main Page; /wiki/Detroit
Wikipedia, the free encyclopedia; Main Page; /wiki/Escanaba,_Michigan
Wikipedia, the free encyclopedia; Main Page; /wiki/Marquette,_Michigan
Wikipedia, the free encyclopedia; Main Page; /wiki/Glossary_of_nautical_terms#upbound
Wikipedia, the free encyclopedia; Main Page; /wiki/Iron_ore
Wikipedia, the free encyclopedia; Main Page; /wiki/Lake_Huron
Wikipedia, the free encyclopedia; Main Page; /wiki/New_Presque_Isle_Light
Wikipedia, the free encyclopedia; Main Page; /wiki/Glossary_of_nautical_terms#canaller
Wikipedia, the free encyclopedia; Main Page; /wiki/National_Register_of_Historic_Places
```

PROGRAM NO : 16**Date :16/02/2022****AIM : Program to implement scrap of any website.****PROGRAM CODE**

```
import requests

from bs4 import BeautifulSoup

import csv

URL = "http://www.values.com/inspirational-quotes"

r = requests.get(URL)

print(r.content)

soup = BeautifulSoup(r.content, 'lxml')

print(soup.prettify())

quotes = []

table = soup.find('div', attrs={'id': 'all_quotes'})

for row in table.findAll('div',

                        attrs={'class': 'col-6 col-lg-3 text-center margin-30px-bottom sm-margin-30px-top'}):

    quote = { }

    quote['theme'] = row.h5.text

    quote['url'] = row.a['href']

    quote['img'] = row.img['src']

    quote['lines'] = row.img['alt'].split(" #")[0]

    quote['author'] = row.img['alt'].split(" #")[1]

    quotes.append(quote)
```

```
filename = 'inspirational_quotes.csv'
```

```
with open(filename, 'w', newline='') as f:
```

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

```
    w.writeheader()
```

```
    for quote in quotes:
```

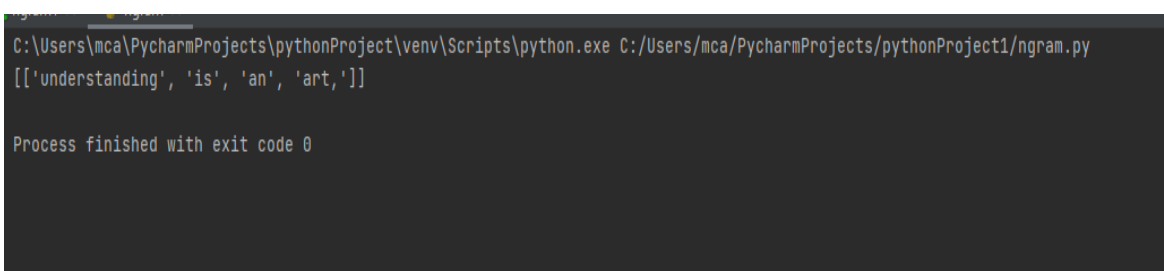
```
        w.writerow(quote)
```

OUTPUT

```
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 Quotes | PassItOn.com\n  </title>\n  <meta charset="utf-8"/>\n  <meta content="text/html; charset=utf-8" http-equiv="content-type"/>\n  <meta content="IE=edge" http-equiv="X-UA-Compatible"/>\n  <meta content="width=device-width,initial-scale=1.0" name="viewport"/>\n  <meta content="The Foundation for a Better Life | Pass It On.com" name="description"/>\n  <link href="/apple-touch-icon.png" rel="apple-touch-icon" sizes="180x180"/>\n  <link href="/favicon-32x32.png" rel="icon" sizes="32x32" type="image/png"/>\n  <link href="/favicon-16x16.png" rel="icon" sizes="16x16" type="image/png"/>\n  <link href="/site.webmanifest" rel="manifest"/>\n  <link color="#c8102e" href="/safari-pinned-tab.svg" rel="mask-icon"/>\n  <meta content="#c8102e" name="msapplication-TileColor"/>\n  <meta content="#ffffff" name="theme-color"/>\n  <link crossorigin="anonymous" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma1\n  <link href="/assets/application-2a7a8e0a1c3f028bac9efab0420f5579.css" media="all" rel="stylesheet"/>\n  <meta content="authenticity token" name="csrf-param"/>\n  </head>\n</html>\n'
```

PROGRAM NO : 17**Date :16/02/2022****AIM : Program for Natural Language Processing which performs n-grams.****PROGRAM CODE**

```
def generate_ngrams(text, WordsToCombine):  
  
    words = text.split()  
  
    output = []  
  
    for i in range(len(words) - WordsToCombine + 1):  
  
        output.append(words[i:i+1 + WordsToCombine])  
  
    return output  
  
x=generate_ngrams(text='understanding is an art, not everyone is an artist',  
WordsToCombine=3)  
  
print(x)
```

OUTPUT

```
C:\Users\mca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\mca\PycharmProjects\pythonProject1\ngram.py  
[[\'understanding\', \'is\', \'an\', \'art,\']]  
  
Process finished with exit code 0
```


PROGRAM NO : 18**Date :16/02/2022****AIM : Program for Natural Language Processing which performs n-grams (Using in built functions).****PROGRAM CODE**

```
import nltk

nltk.download('punkt')

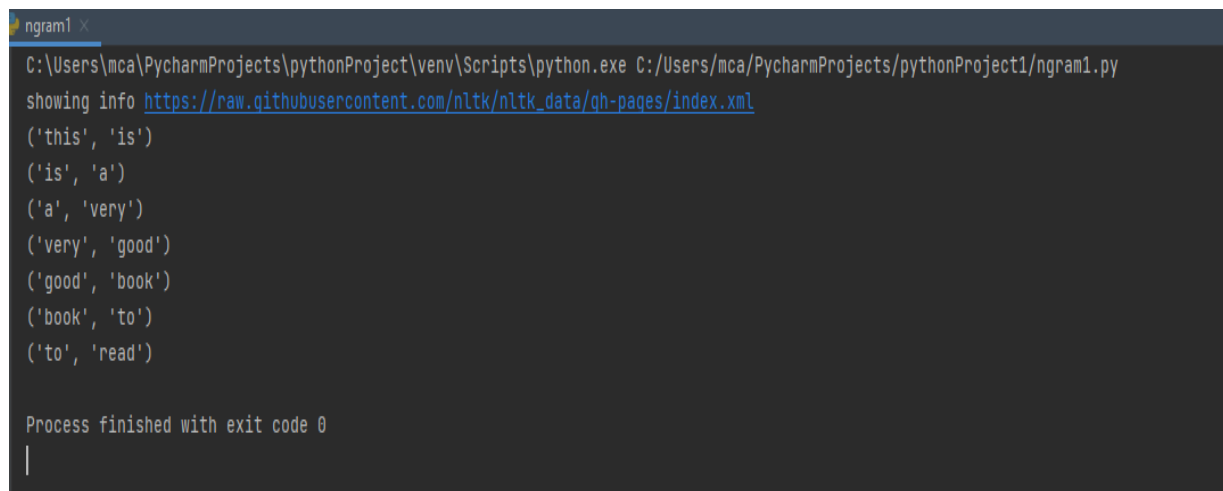
from nltk.util import ngrams

sampleText='this is a very good book to study'

NGRAMS=ngrams(sequence=nltk.word_tokenize(sampleText),n=2)

for grams in NGRAMS:

    print(grams)
```

OUTPUT

```
ngram1 x
C:\Users\mca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/mca/PycharmProjects/pythonProject1/ngram1.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', 'read')

Process finished with exit code 0
|
```

PROGRAM NO : 19**Date :16/02/2022****AIM : Program for Natural Language Processing which performs speech tagging.****PROGRAM CODE**

```
import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize, sent_tokenize

stop_words = set(stopwords.words('english'))

txt = "Sukanya, Rajib and Naba are my good friends. " \
      "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

```
nlp x
C:\Users\mca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/mca/PycharmProjects/pythonProject1/nlp.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.It', 'NN'), ('exciting', 'VBG'), ('frightening', 'NN'), ('.', '.')]
[('But', 'CC'), ('friendship', 'NN'), ('sacred', 'VBD'), ('bond', 'NN'), ('people.It', 'NN'), ('special', 'JJ'), ('kind', 'NN'), ('love', 'VB'), ('us', 'PR')]
[('Many', 'JJ'), ('must', 'MD'), ('tried', 'VB'), ('searching', 'VBG'), ('friend', 'NN'), ('never', 'RB'), ('found', 'VBD'), ('right', 'JJ'), ('one', 'CD')]

Process finished with exit code 0
```

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

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

PROGRAM CODE

```
import nltk

new="The big cat ate the little mouse who was after the fresh cheese"

new_tokens=nltk.word_tokenize(new)

print(new_tokens)

new_tag=nltk.pos_tag(new_tokens)

print(new_tag)

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

chunkParser=nltk.RegexpParser(grammer)

chunked=chunkParser.parse(new_tag)

print(chunked)

chunked.draw()
```

OUTPUT

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

