20MCA241 DATA SCIENCE LAB

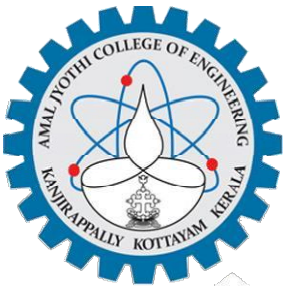
Lab Report Submitted By

AKSA ANNA JOSE

Reg. No.:AJC20MCA-2006

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

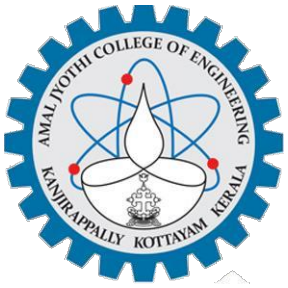
[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

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 **AKSA ANNA JOSE (Reg.No:AJC20MCA-2006)** 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.Meera Rose Mathew — **Staff In-Charge**

Rev. Fr. Dr. RubinThottupurathu Jose — **Head of the Department**

Internal Examiner

External Examiner

CONTENT

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**	**19**	
11	**Program to implement K-Means Clustering technique using any standard dataset available in the public domain.**	**05/01/2022**	**26**	
12	**Program to implement K-Means Clustering technique using any standard dataset available in the public domain.**	**05/01/2022**	**29**	
13	**Programs on Convolutional Neural Network(CNN) to classify images from any standard dataset in the public domain.**	**02/02/2022**	**32**	
14	**Program to implement a simple Web Crawler using python.**	**16/02/2022**	**37**	
15	**Program to implement a simple Web Crawler using python.**	**16/02/2022**	**41**	
16	**Program to implement scraping of any webpage of any popular website.**	**16/02/2022**	**43**	
17	**Program for Natural Language Processing which performs n-grams.**	**16/02/2022**	**45**	
18	**Program for Natural Language Processing which performs n-grams(Using built-in functions).**	**16/02/2022**	**46**	
19	**Program for Natural Language Processing which performs speech tagging.**	**16/02/2022**	**47**	
20	**Write a python program for natural program language processing with chunking.**	**23/02/2022**	**48**	
21	**Write a python program for natural program language processing with chunking.**	**23/02/2022**	**50**	

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

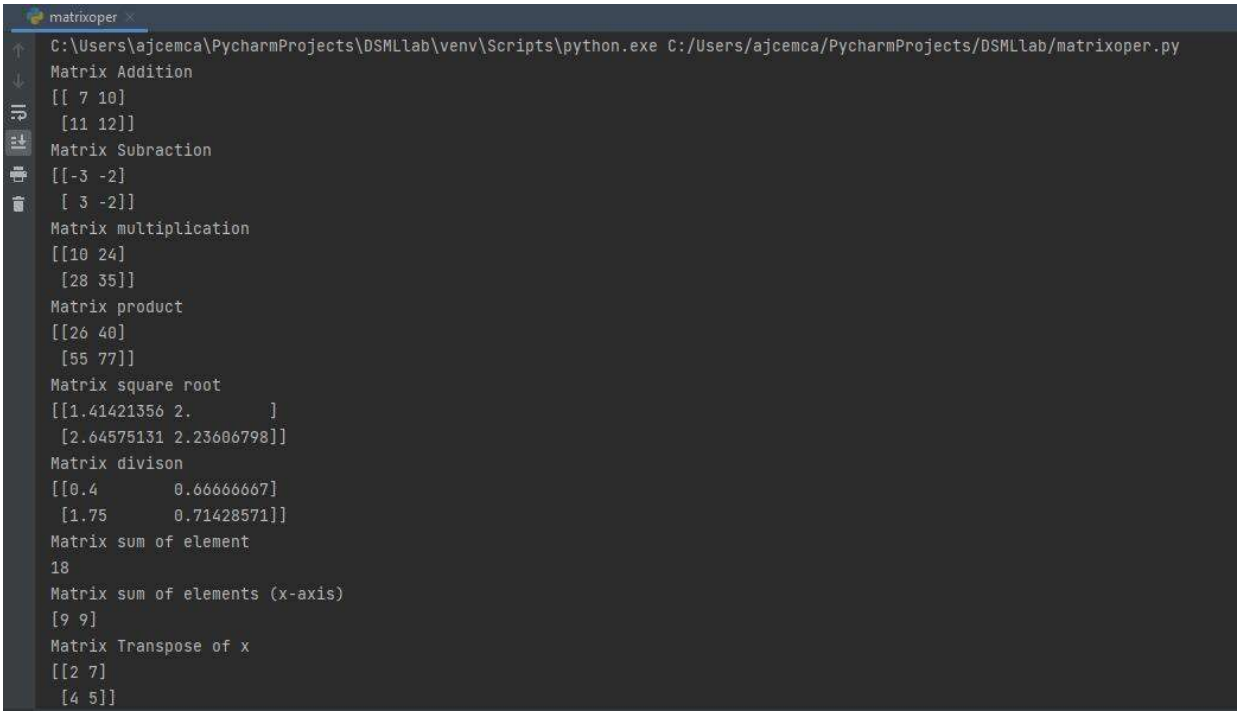
AIM: Perform all matrix operation using python

Program Code:

```
import numpy

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

print ("The matrices are: ")
print ("First matrix:")
print (x)
print ("Second matrix: ")
print(y)

#Addition--- add()
print ("matrix addition:")
print (numpy.add(x,y))

#Subtraction ------ subtract()
print ("matrix Subtraction:")
print (numpy.subtract(x,y))

#Division -------- divide()
print ("matrix Division")
print (numpy.divide(x,y))

#Multiplication -------- multiply
print ("matrix Multiplication")
print (numpy.multiply(x,y))

#Product of matrix ---- dot()
print ("Product of 2 matrix: ")
print (numpy.dot(x,y))

#Square root ----sqrt()
print ("Square root of matrix X: ")
print (numpy.sqrt(x))


#Summation ---- sum()
print ("Summation of matrix X: ")
print  (numpy.sum(x,axis=0))
print ("Summation of matrix Y: ")
print (numpy.sum(y,axis=1))
```

```
#Transposition------T
print ("Transposition of matrix X: ")
print(x.T)
print ("Transposition of matrix Y: ")
print(y.T)
```

Output:



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

AIM: Program to perform SVD using python

Program Code:

```
from numpy import array
from scipy.linalg import svd                    #a function in SCIPY
A = ([[8,4,5,7], [4,1,6,9], [6,1,0,9]]);

print("ACTUAL MATRIX IS: ")
print(A);
U, s, VT = svd(A)                               # U, s, VT are just 3 variables
                                        # U= decomposed  s=inverse   #VT=transpos

print("DECOMPOSED MATRIX: ")
print(U);

print("INVERSE MATRIX: ")
print(s);

print("TRANSPOS MATRIX: ")
print(VT);
```

OUTPUT:

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scrip
[[ 1  2  3  4]
 [ 7  8  3  5]
 [ 4  6  9 10]]
Decomposed Matrix
 [[-0.27122739  0.25018762  0.92943093]
 [-0.575834   -0.81593689  0.05159647]
 [-0.77126579  0.52120355 -0.36537097]]
Inverse Matrix
 [19.40153082  5.77253959  0.5083193 ]
Transpose matrix
 [[-0.38074978 -0.50391495 -0.48875402 -0.60184619]
 [-0.5849343  -0.50236097  0.5185905   0.36952567]
 [-0.336162    0.15621646 -0.67921184  0.63345308]
 [-0.63235795  0.68505445  0.17565499 -0.31617898]]

Process finished with exit code 0
```

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

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

Program Code:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

 #Split arrays or matrices into random train and test subsets

from  sklearn.datasets import load_iris

#Load and return the iris dataset (classification).

from sklearn.metrics import accuracy_score

# to load dataset values
dataset = load_iris()

# features & target
d = dataset.data                    # feature
t = dataset.target                    # target

d_train, d_test, t_train, t_test = train_test_split(d, t, test_size=0.2, random_state=40)

knn = KNeighborsClassifier(n_neighbors=10)

knn.fit(d_train, t_train)

   #Fit the k-nearest neighbors classifier from      the training dataset.

print(knn.predict(d_test))

a = knn.predict(d_test)
ac = accuracy_score(t_test, a)     #store accuracy value
print("Accuracy value is : ")

print(ac)
```

Output:

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/aj
[0 0 0 2 2 0 2 2 0 0 0 1 2 0 2 2 1 0 1 2 1 2 1 2 1 1 2 1 2 1 0 0 2 2 0 1 1
 0 2 1 2 0 1 0 1]
Accuracy score :  0.9555555555555556

Process finished with exit code 0
```

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

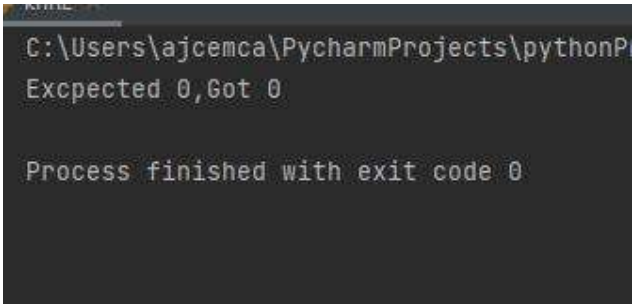
AIM: Program to implement k-NN Classification using any random dataset without using in- build functions

Program Code:

```
from math import sqrt

def euclidian_distance(row1, row2):

distance = 0.0

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

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

return sqrt(distance)

# locat the most similar neighbor

def get_neighbors(train, test_row, num_neighbors):

distances = list()

for train_row in train:

dist = euclidian_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]                #store the data of neighbors

prediction = max(set(output_values), key=output_values.count)

return prediction
```

```
# test distance function
dataset = [[2.5477838, 2.753590, 0],
[1.45778788,2.7767373, 0],
[3.678838, 4.6788288, 0],
[1.436773, 1.53773, 0],
[3.76888389, 3.6748, 0],
[7.7848848, 2.759256, 1],
[5.782356, 2.246378, 1],
[6.777878, 1.49078, 1],
[8.677728889, -0.7588392, 1],
 [7.675637, 3.59340, 1]]
prediction = predict_classification(dataset, dataset[0], 3)
print("Expexted %d,Got %d."%(dataset[0][-1],prediction))
```

Output:



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:

```
# Random Forest Classification

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

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(y_pred)
plt.plot(y_pred)
plt.title('Gender Salary')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
#plt.legend()
plt.show()
```

Output:

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajce
[0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0
 1 1 0 1 0 0 0 0 0 0 1 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0
 1 1 0 1 0 0]
0.875
```

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

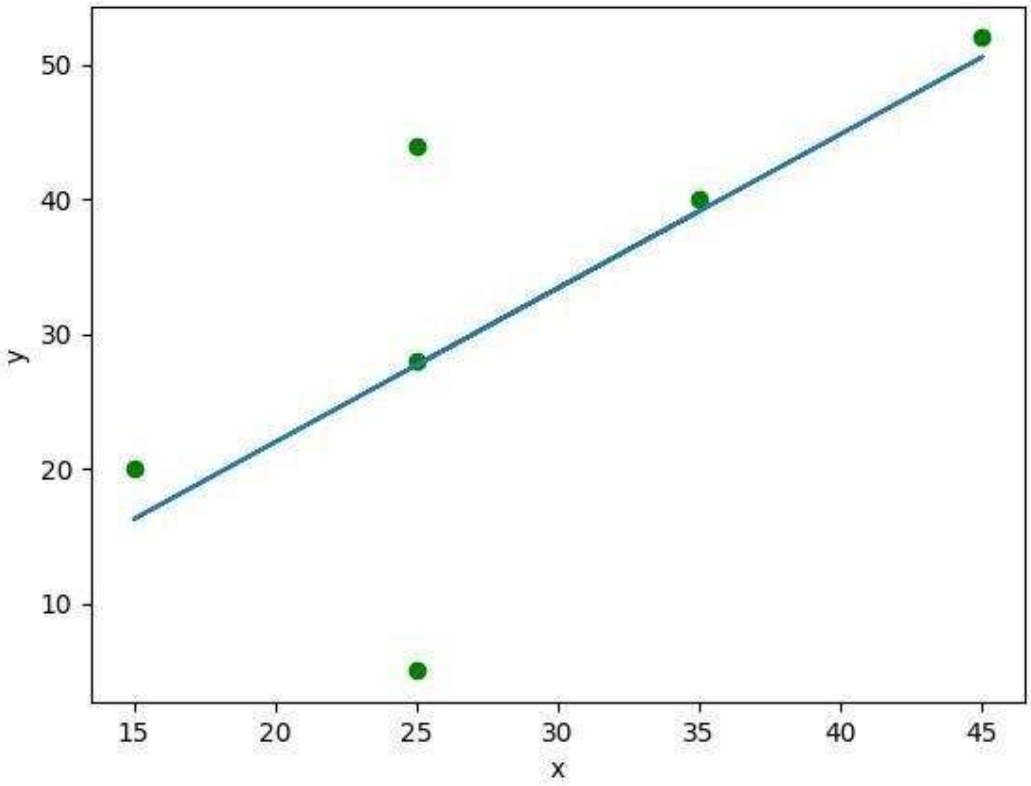
AIM: Program to implement linear and multiple regression techniques using any standard dataset available in the public domain

Program Code:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
x = np.array([1,5,1,9,33,2]).reshape((-1, 1))
y = np.array([2,7,1,9,3,40])
print(x)
print(y)
model=LinearRegression()
model.fit(x, y)
r_sq=model.score(x,y)
print('Coefficient od determination: ', r_sq)
print('Intercept: ', model.intercept_)
print('Slope: ', model.coef_)
y_pred=model.predict(x)

plt.plot(x, y_pred, color="r")
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Output:

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



PROGRAM NO: 07 **Date:** 15/12/2021

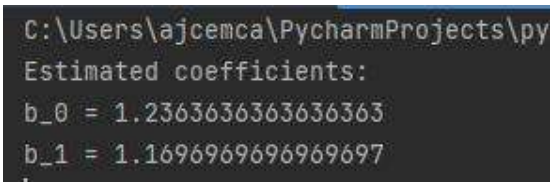
AIM: Program to implement linear and multiple regression techniques using any standard dataset available in the public domain

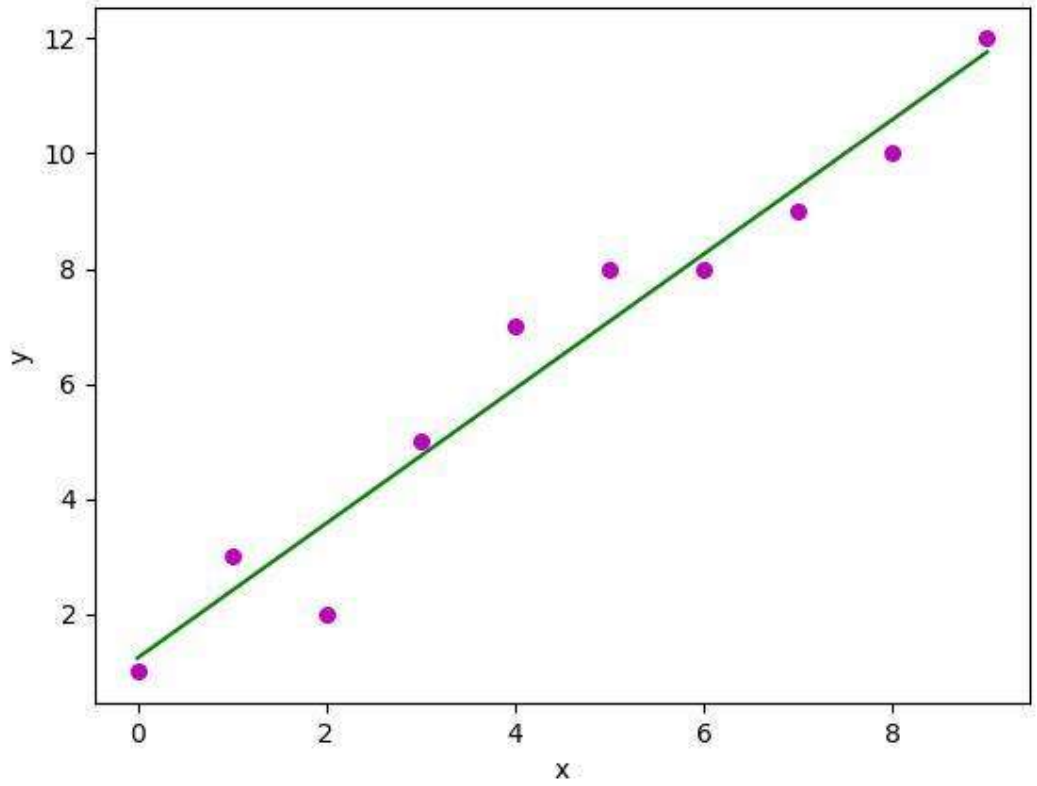
Program Code:

```
import numpy as np
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="g",
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 / data
x = np.array([0, 2, 2, 3, 5, 5, 6, 8, 9, 10])
y = np.array([1, 3, 4, 6, 8, 10, 12, 13, 14, 16])

# 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:




12
10
8
6
4
2
y
0
2
4
6
8
x

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 as pd
import numpy as np
import matplotlib.pyplot as plt
dataset = pd.read_csv("cars.csv")
dataset.head()
dataset.describe()
X = dataset[['Weight', 'Volume']]
y = dataset['CO2']
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=0)
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train, y_train)
r2_score = regressor.score(X_test,y_test)
print("Accuracy: ")
print(r2_score*100,'%')

coeff_df = pd.DataFrame(regressor.coef_, X.columns, columns=['Coefficient'])
coeff_df
print("co-efficient of correlation: ")
print(regressor.coef_)
```

Output:

```
[107.2087328]
[0.00755095 0.00780526]
```

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
from sklearn import datasets, linear_model, metrics
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_)
# Intecept
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('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:

```
Prediction :  [32.65503184 28.0934953  18.02901829 21.47671576 18.8254387  19.87997758
 32.42014863 18.06597765 24.42277848 27.00977832 27.04081017 28.75196794
 21.15677699 26.85200196 23.38835945 20.66241266 17.33082198 38.24813601
 30.50550873  8.74436733 20.80203902 16.26328126 25.21805656 24.85175752
 31.384365   10.71311063 13.80434635 16.65930389 36.52625779 14.66750528
 21.12114902 13.95558618 43.16210242 17.97539649 21.80116017 20.58294808
 17.59938821 27.2212319   9.46139365 19.82963781 24.30751863 21.18528812
 29.57235682 16.3431752  19.31483171 14.56343172 39.20885479 18.10887551
 25.91223267 20.33018802 25.16282007 24.42921237 25.07123258 26.6603279
  4.56151258 24.0818735  10.88682673 26.88926656 16.85598381 35.88704363
 19.55733853 27.51928921 16.58436103 18.77551029 11.13872875 32.36392607
 36.72833773 21.95924582 24.57949647 25.14868695 23.42841301  6.90732017
 16.56298149 20.41940517 20.80403418 21.54219598 33.85383463 27.94645899
 25.17281456 34.65883942 18.62487738 23.97375565 34.6419296  13.34754896
 20.71097982 30.0803549  17.13421671 24.30528434 19.25576671 16.98006722
 27.00622638 41.85509074 14.11131512 23.25736073 14.66302672 21.86977175
 23.02527624 29.0899182  37.11937872 20.53271022 17.36840034 17.71399314]
Coefficients:  [-1.12386867e-01  5.80587074e-02  1.83593559e-02  2.12997760e+00
 -1.95811012e+01  3.09546166e+00  4.45265228e-03 -1.50047624e+00
  3.05358969e-01 -1.11230879e-02 -9.89007562e-01  7.32130017e-03
 -5.44644997e-01]
Variance Score:0.763417443213847
```

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:

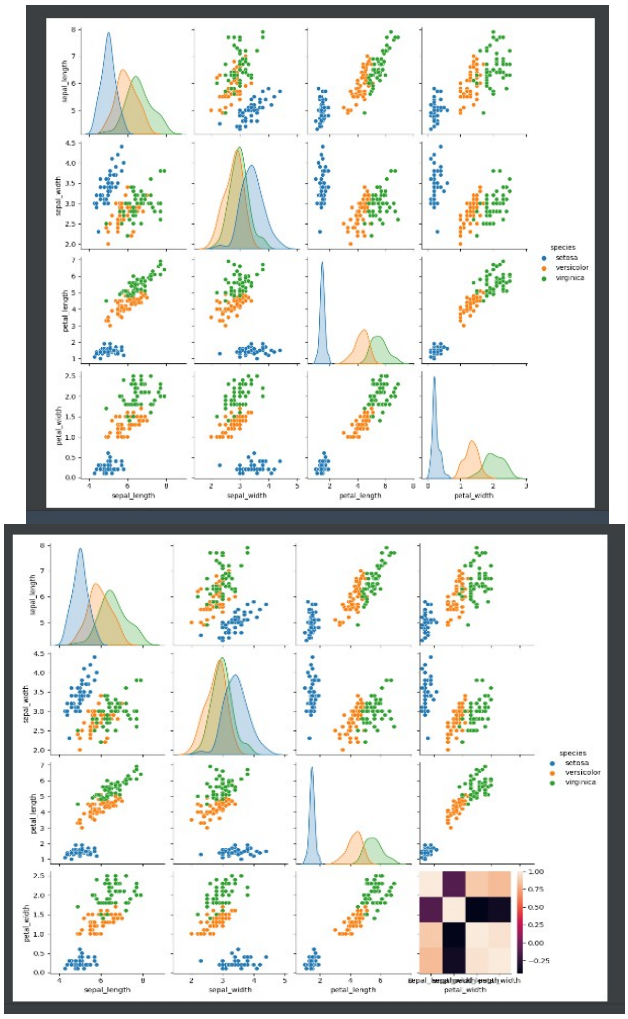
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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")

#correlation 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 attributes
x=df1
print(target)

#label encoding
le = LabelEncoder()
target = le.fit_transform(target)      #learn scaling parameters(species)
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)
```

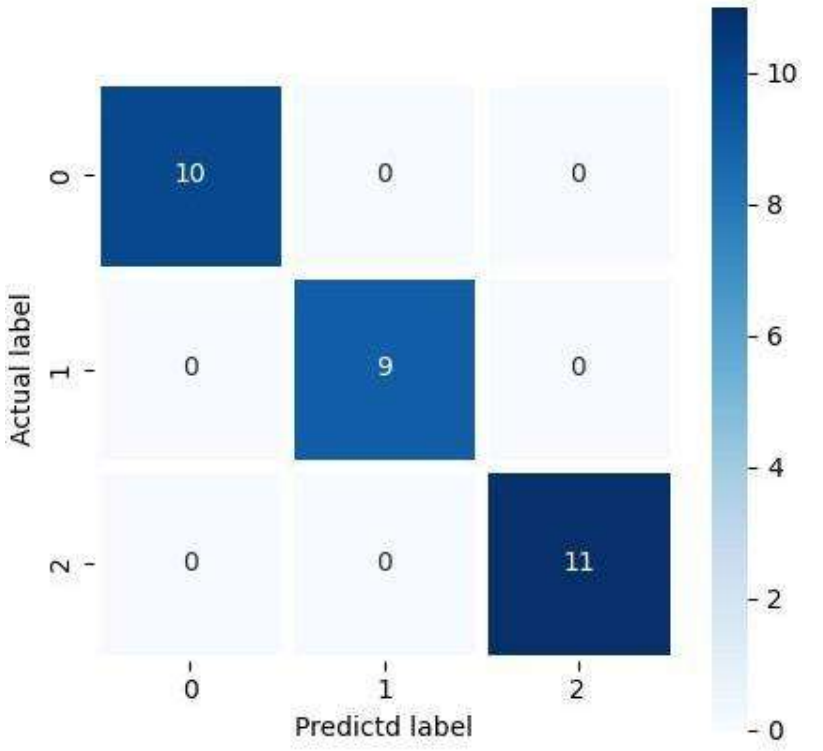
```
#defining the decision tree algorithm
dtree = DecisionTreeClassifier()
dtree.fit(x_train, y_train)
print('Decision tree classifier created')

#predicting the value of test data
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")
```

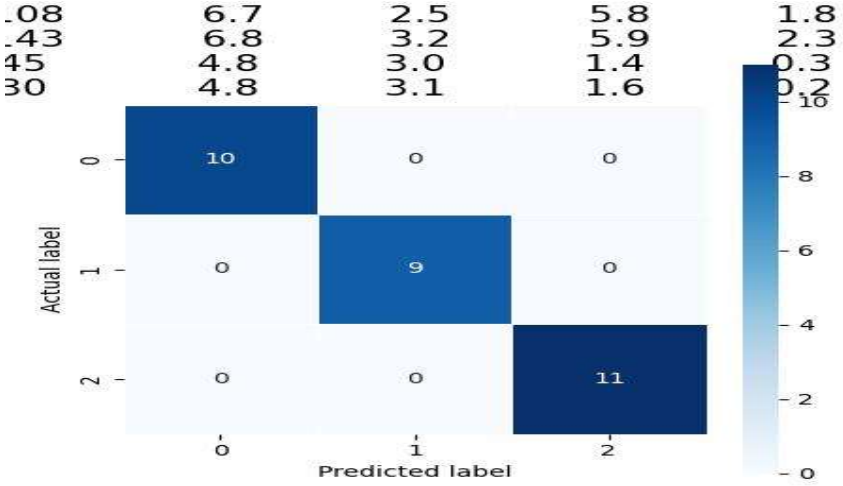
Output:

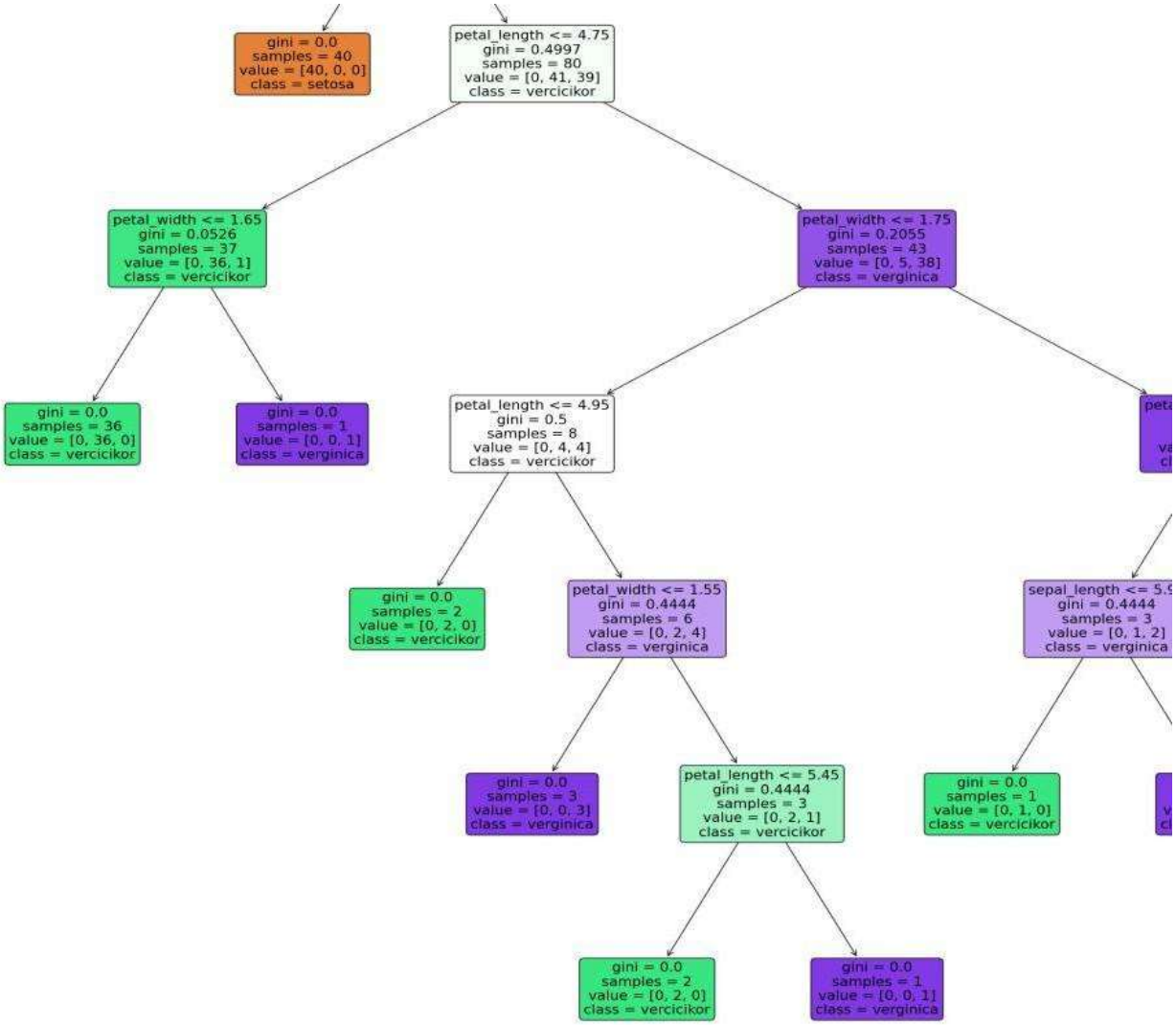
```
C:\Users\ashis\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/ashis/PycharmProjects/pythonProject1/venv/d
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
(150, 5)
(150, 4)
   sepal_length  sepal_width  petal_length  petal_width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
0         setosa
1         setosa
```

```
2          setosa
3          setosa
4          setosa
          ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 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)
Decision tree classifier created
classsification report
              precision    recall  f1-score   support

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

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


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica


10
0
0
0
9
0
0
0
11
0
1
2
Actual label
Predictd label
10
8
6
4
2
0


.08 6.7 2.5 5.8 1.8
.43 6.8 3.2 5.9 2.3
45 4.8 3.0 1.4 0.3
30 4.8 3.1 1.6 0.2
10
0
0
0
9
0
0
0
11
0
1
2
Actual label
Predicted label
10
8
6
4
2
0


gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal_length <= 4.75
gini = 0.4997
samples = 80
value = [0, 41, 39]
class = vercicikor
petal_width <= 1.65
gini = 0.0526
samples = 37
value = [0, 36, 1]
class = vercicikor
petal_width <= 1.75
gini = 0.2055
samples = 43
value = [0, 5, 38]
class = verginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = vercicikor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = verginica
petal_length <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
class = vercicikor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicikor
petal_width <= 1.55
gini = 0.4444
samples = 6
value = [0, 2, 4]
class = verginica
sepal_length <= 5.9
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = verginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = verginica
petal_length <= 5.45
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = vercicikor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = vercicikor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicikor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = verginica

PROGRAM NO: 11 **Date:** 05/01/2021

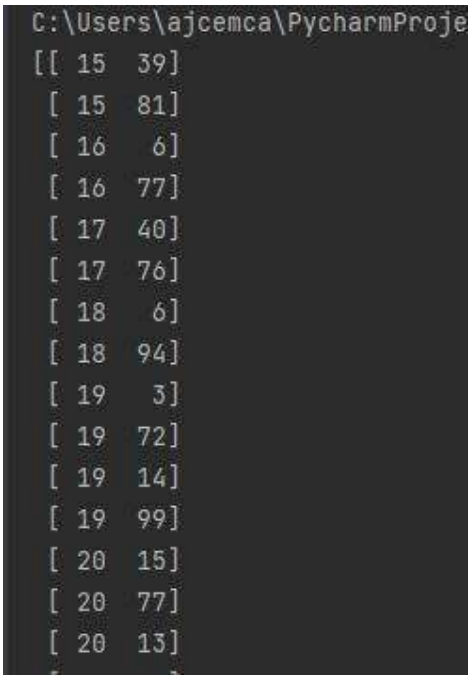
AIM: Program to implement K-Means clustering technique using any standard dataset available in the public domain.

Program Code:

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

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

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

print(x)

from sklearn.cluster import KMeans

wcss_list=[]

for i in range(1,11):

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

kmeans.fit(x)

wcss_list.append(kmeans.inertia_)

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

mtp.title('The Elbow Method Graph')

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

mtp.ylabel('wcss_list')

mtp.show()

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


y_predict=kmeans.fit_predict(x)

print(y_predict)
```

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

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

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

mtp.scatter(x[y_predict ==3,0],x[y_predict ==3,1],s=100,c='cyan',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',label='clust
er')

mtp.title('Clusters of customers')

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

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

mtp.legend()

mtp.show()
```

Output:



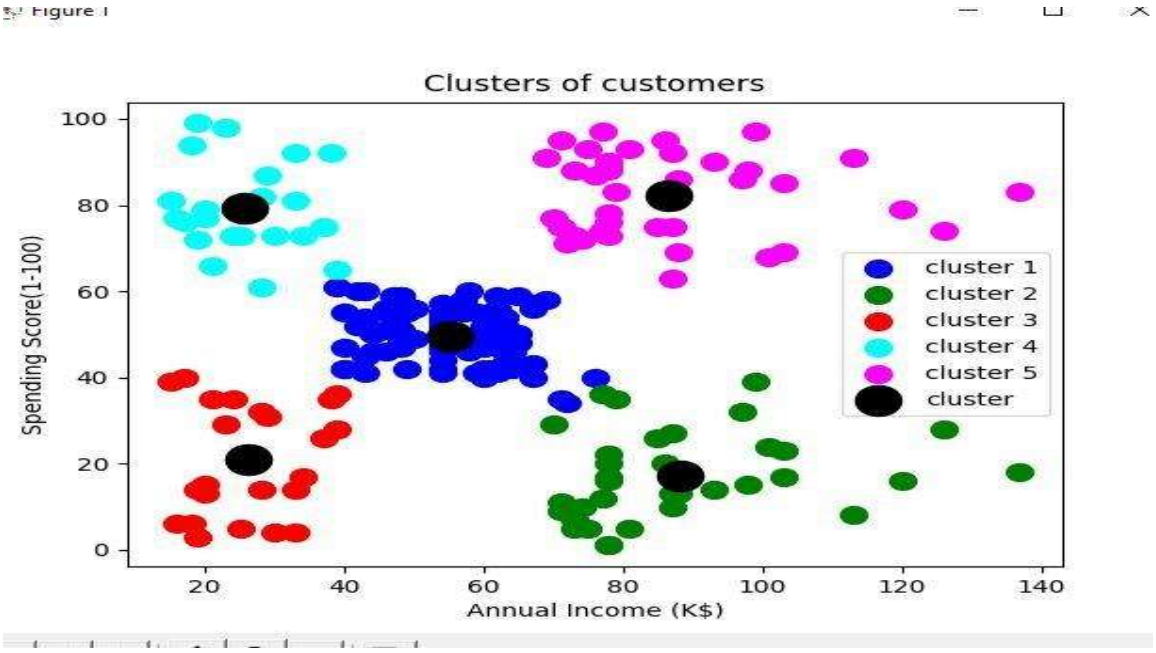


Figure 1
Clusters of customers
Spending Score(1-100)
Annual Income (K$)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster
0
20
40
60
80
100
120
140

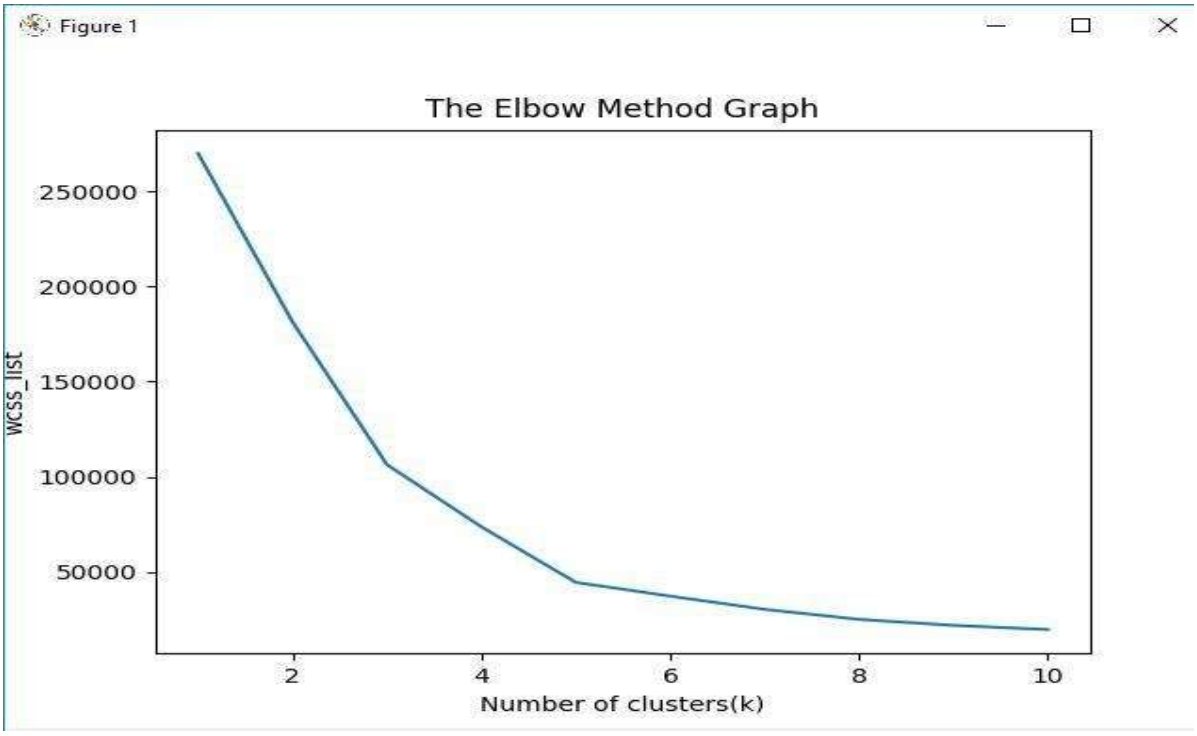


Figure 1
The Elbow Method Graph
WCSS_list
Number of clusters(k)
50000
100000
150000
200000
250000
2
4
6
8
10

PROGRAM NO: 12 **Date:** 05/01/2021

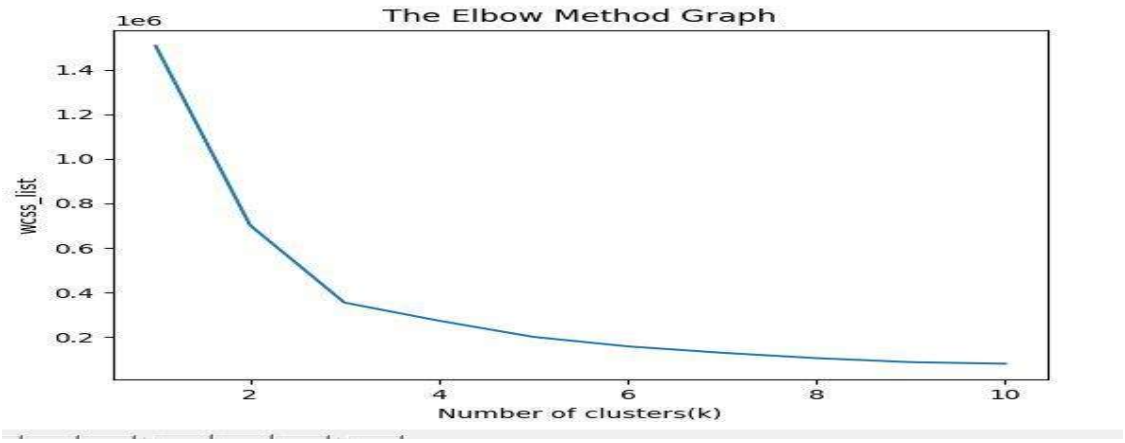
AIM: Program to implement K-Means clustering technique using any standard dataset available in the public domain

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(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='green',label='cluster 2')
mtp.scatter(x[y_predict ==2,0],x[y_predict ==2,1],s=100,c='red',label='cluster 3')
mtp.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='black',label='clust
er')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (K$)')
mtp.ylabel('Spending Score(1-100)')
mtp.legend()
mtp.show()
```

Output:

```
C:\Users\ajcemca\PycharmProjects\Rmca_DLMLLab_28
[[ 4.25462450e+01  1.60155400e+00]
 [ 2.34240760e+01  5.38478180e+01]
 [ 3.39391100e+01  6.77099530e+01]
 [ 1.70608160e+01 -6.17964280e+01]
 [ 1.82205540e+01 -6.30686150e+01]
 [ 4.11533320e+01  2.01683310e+01]
 [ 4.00690990e+01  4.50381890e+01]
 [ 1.22260790e+01 -6.90600870e+01]
 [-1.12026920e+01  1.78738870e+01]
 [-7.52509730e+01 -7.13890000e-02]
 [-3.84160970e+01 -6.36166720e+01]
 [-1.42709720e+01 -1.70132217e+02]
 [ 4.75162310e+01  1.45500720e+01]
 [-2.52743980e+01  1.33775136e+02]
 [ 1.25211100e+01 -6.99683380e+01]
 [ 4.01431050e+01  4.75769270e+01]
 [ 4.39158860e+01  1.76790760e+01]
 [ 1.31938870e+01 -5.95431980e+01]
 [ 2.36849940e+01  9.03563310e+01]
```



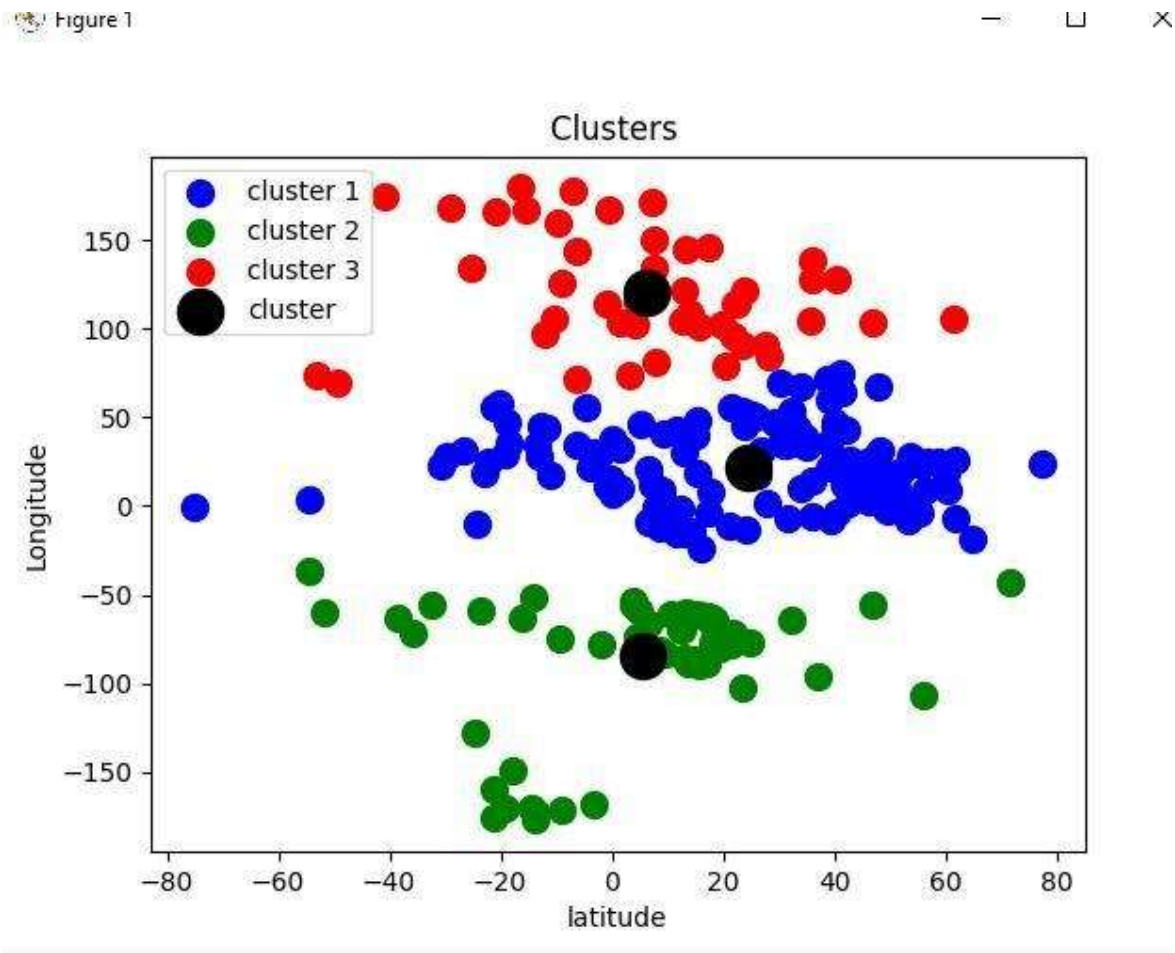


Figure 1
Clusters
cluster 1
cluster 2
cluster 3
cluster
Longitude
latitude
150
100
50
0
−50
−100
−150
−80
−60
−40
−20
0
20
40
60
80

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

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

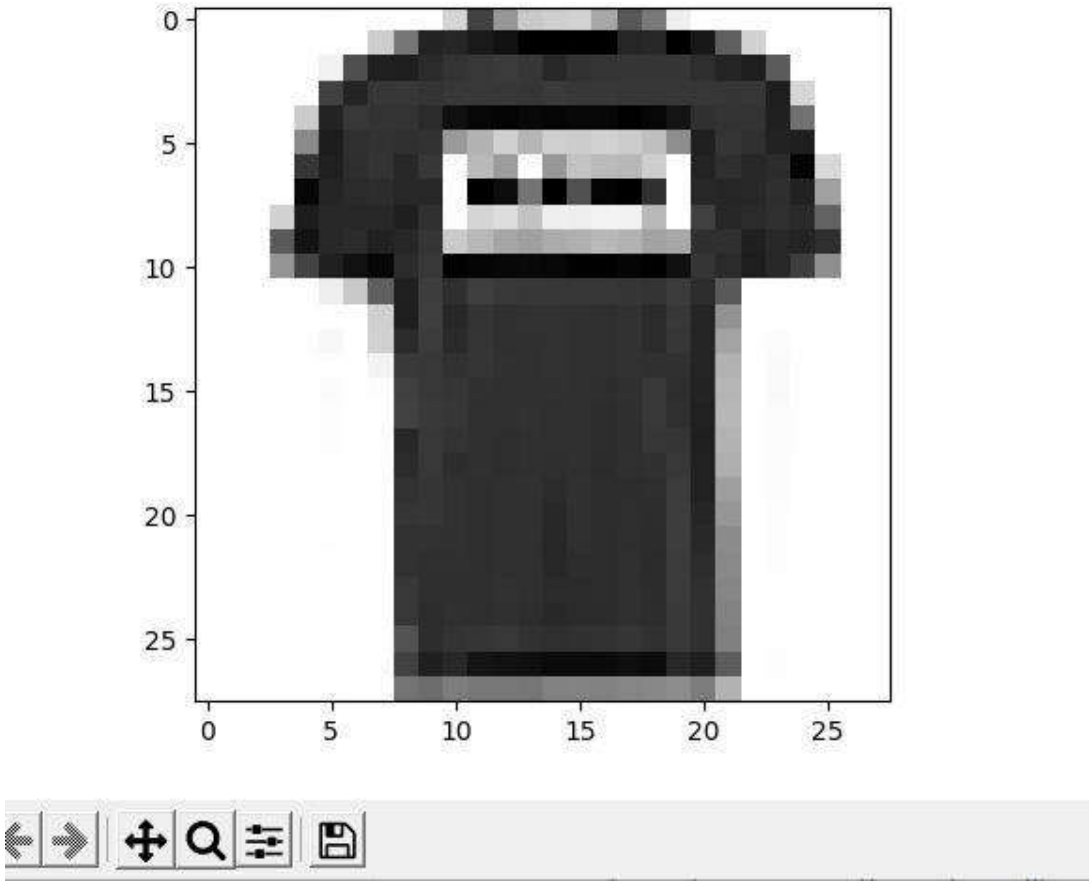
Program Code:

```
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)

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','Bag','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
```

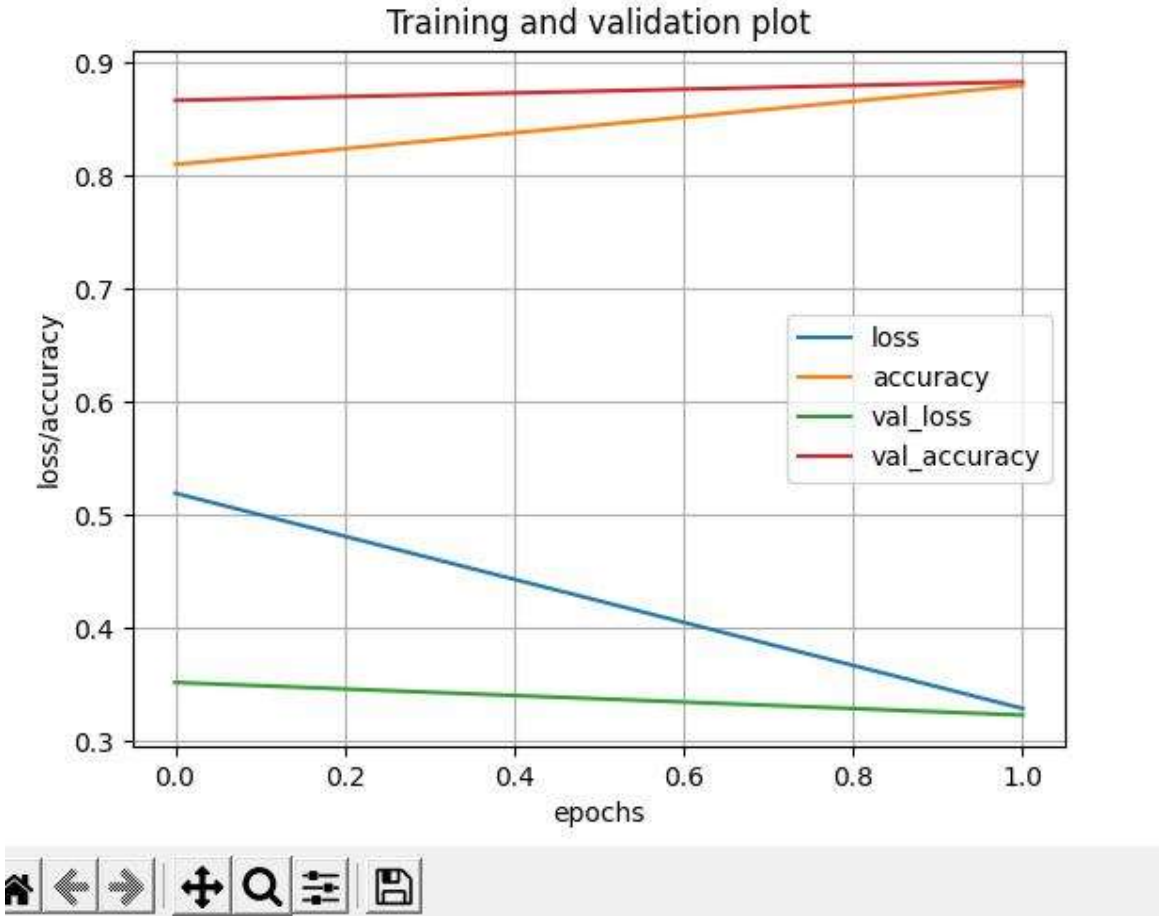
```
plt.subplot(n_rows,n_cols,index+1)

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',i
nput_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:




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


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

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
url = "https://rottentomatoes.com/top/bestofrt/"
header = {
   '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=header)
movies_lst = []
soup = BeautifulSoup(f.content, 'lxml')
movies = soup.find('table',{
   'class': 'table'
}) .find_all('a')
print(movies)
num = 0
for anchor in movies:
urls = 'https://rottentomatoes.com' + anchor['href']
movies_lst.append(urls)
print(movies_lst)
num += 1
movie_url = urls
movie_f = requests.get(movie_url, headers=header)
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', 'Movies: ' + anchor.string.strip())
print('Movies 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 class="unstyled articleLink"
href="/m/1003707-casablanca">
Casablanca (1942), <a class="unstyled articleLink"
href="/m/knives_out">Knives Out (2019), <a class="unstyled
articleLink" href="/m/us_2019"> Us (2019), <a class="unstyled
articleLink" href="/m/toy_story_4">
Toy Story 4 (2019), <a class="unstyled articleLink"
href="/m/lady_bird">Lady Bird (2017), <a class="unstyled articleLink"
href="/m/mission_impossible_fallout">
Mission: Impossible - Fallout (2018), <a class="unstyled
articleLink"href="/m/blackkklansman">
BlacKkKlansman (2018), <a class="unstyled articleLink"
href="/m/get_out">Get Out (2017), <a class="unstyled articleLink"
href="/m/the_irishman"> The Irishman (2019), <a class="unstyled
articleLink" href="/m/godfather"> The Godfather (1972), <a
class="unstyled articleLink"
href="/m/mad_max_fury_road">
Mad Max: Fury Road (2015), <a class="unstyled
articleLink"href="/m/spider_man_into_the_spider_verse">
Spider-Man: Into the Spider-Verse (2018), <a class="unstyled
articleLink"href="/m/moonlight_2016">
Moonlight (2016), <a class="unstyled articleLink"
href="/m/sunset_boulevard">Sunset Boulevard (1950), <a class="unstyled
articleLink" href="/m/1000626-
all_about_eve">
All About Eve (1950), <a class="unstyled
articleLink"href="/m/the_cabinet_of_dr_caligari">
The Cabinet of Dr. Caligari (Das Cabinet des Dr. Caligari) (1920),
<aclass="unstyled articleLink" href="/m/philadelphia_story">
The Philadelphia Story (1940), <a class="unstyled
articleLink"href="/m/roma_2018">
Roma (2018), <a class="unstyled articleLink"
href="/m/wonder_woman_2017">Wonder Woman (2017), <a class="unstyled
articleLink"
href="/m/a_star_is_born_2018">

href="/m/eighth_grade">Eighth Grade (2018), <a class="unstyled articleLink"
href="/m/1017293-rebecca"> Rebecca (1940), <a class="unstyled articleLink"
href="/m/booksmart">
Booksmart (2019), <a class="unstyled articleLink"
href="/m/logan_2017">Logan (2017), <a class="unstyled articleLink"
href="/m/his_girl_friday"> His Girl Friday (1940), <a class="unstyled
articleLink"
href="/m/portrait_of_a_lady_on_firee
Boyhood (2014),
Gravity (2013),
Leave No Trace (2018), <a class="unstyled articleLink" href="/m/1013139-
maltese_falcon">
The Maltese Falcon (1941), <a class="unstyled
articleLink"href="/m/the_invisible_man_2020">
Rear Window (1954), <a class="unstyled
articleLink"href="/m/et_the_extraterrestrial">
E.T. The Extra-Terrestrial (1982), <a class="unstyled
articleLink"href="/m/seven_samurai_1956">
Seven Samurai (Shichinin no Samurai) (1956), <a class="unstyled articleLink"
href="/m/la_grande_illusion">
La Grande illusion (Grand Illusion) (1938), <a class="unstyled
articleLink"href="/m/arrival_2016">
Arrival (2016), <a class="unstyled articleLink"
href="/m/singin_in_the_rain">Singin' in the Rain (1952), <a
class="unstyled articleLink"
href="/m/the_favourite_2018">
The Favourite (2018), <a class="unstyled articleLink"
href="/m/double_indemnity">Double Indemnity (1944), <a class="unstyled
articleLink" href="/m/1000642-
all_quiet_on_the_western_front">
All Quiet on the Western Front (1930), <a class="unstyled
articleLink"href="/m/1048445-snow_white_and_the_seven_dwarfs">
Snow White and the Seven Dwarfs (1937), <a class="unstyled
articleLink"href="/m/marriage_story_2019">
Marriage Story (2019),

All Quiet on the Western Front (1930), <a class="unstyled
articleLink"href="/m/1048445-snow_white_and_the_seven_dwarfs">
Snow White and the Seven Dwarfs (1937), <a class="unstyled
articleLink"href="/m/marriage_story_2019">
Marriage Story (2019), <a class="unstyled articleLink" href="/m/the_big_sick">

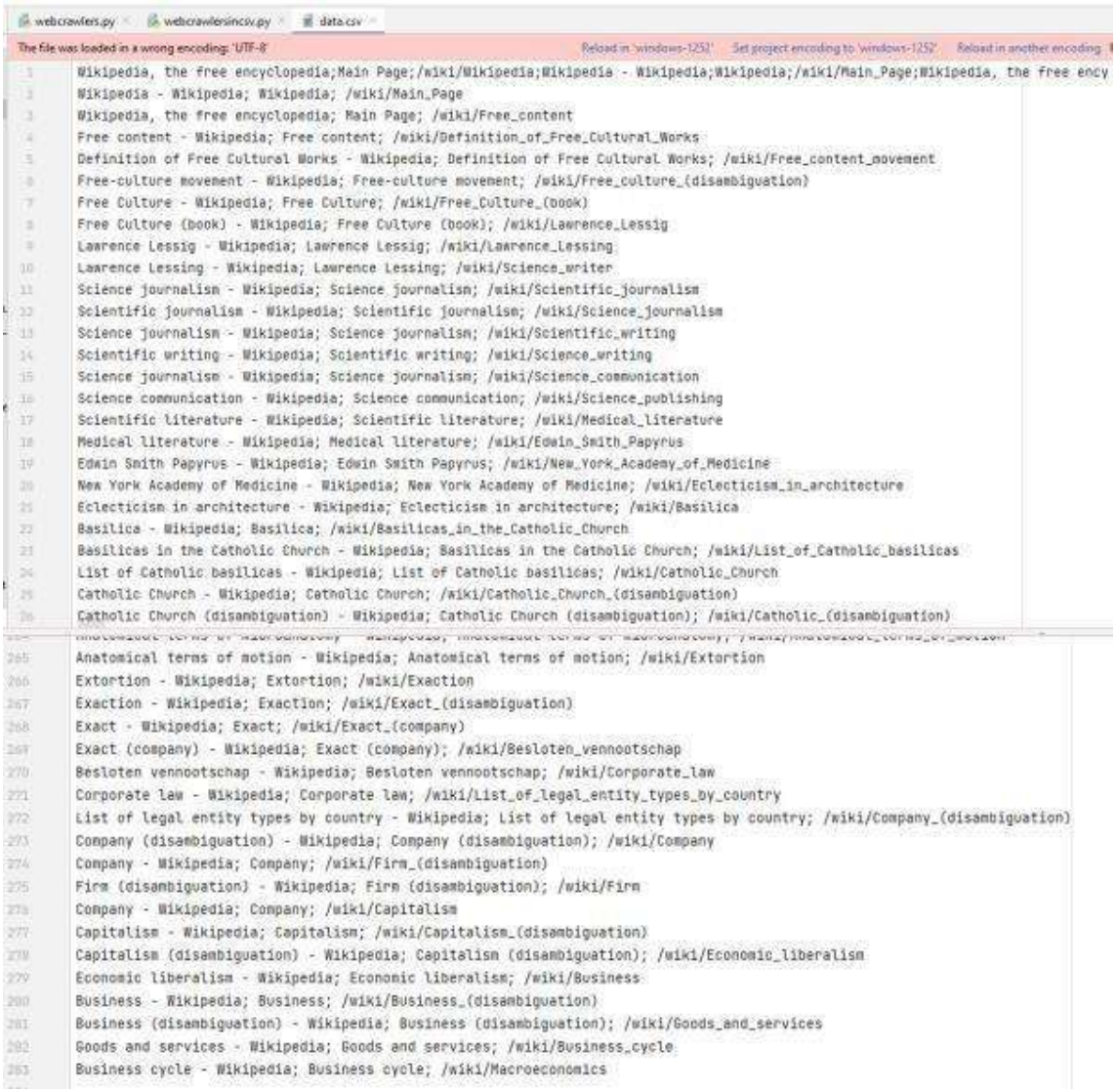
PROGRAM NO: 15 **Date:** 16/02/2022

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 = fhttps://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:



webcrawlers.py | webcrawlersincsv.py | data.csv

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

```
Wikipedia, the free encyclopedia;Main Page;/wiki/Wikipedia;Wikipedia - Wikipedia;Wikipedia;/wiki/Main_Page;Wikipedia, the free ency
Wikipedia - Wikipedia; Wikipedia; /wiki/Main_Page
Wikipedia, the free encyclopedia; Main Page; /wiki/Free_content
Free content - Wikipedia; Free content; /wiki/Definition_of_Free_Cultural_Works
Definition of Free Cultural Works - Wikipedia; Definition of Free Cultural Works; /wiki/Free_content_movement
Free-culture movement - Wikipedia; Free-culture movement; /wiki/Free_culture_(disambiguation)
Free Culture - Wikipedia; Free Culture; /wiki/Free_Culture_(book)
Free Culture (book) - Wikipedia; Free Culture (book); /wiki/Lawrence_Lessig
Lawrence Lessig - Wikipedia; Lawrence Lessig; /wiki/Lawrence_Lessing
Lawrence Lessing - Wikipedia; Lawrence Lessing; /wiki/Science_writer
Science journalism - Wikipedia; Science journalism; /wiki/Scientific_journalism
Scientific journalism - Wikipedia; Scientific journalism; /wiki/Science_journalism
Science journalism - Wikipedia; Science journalism; /wiki/Scientific_writing
Scientific writing - Wikipedia; Scientific writing; /wiki/Science_writing
Science journalism - Wikipedia; Science journalism; /wiki/Science_communication
Science communication - Wikipedia; Science communication; /wiki/Science_publishing
Scientific literature - Wikipedia; Scientific literature; /wiki/Medical_literature
Medical literature - Wikipedia; Medical literature; /wiki/Edwin_Smith_Papyrus
Edwin Smith Papyrus - Wikipedia; Edwin Smith Papyrus; /wiki/New_York_Academy_of_Medicine
New York Academy of Medicine - Wikipedia; New York Academy of Medicine; /wiki/Eclecticism_in_architecture
Eclecticism in architecture - Wikipedia; Eclecticism in architecture; /wiki/Basilica
Basilica - Wikipedia; Basilica; /wiki/Basilicas_in_the_Catholic_Church
Basilicas in the Catholic Church - Wikipedia; Basilicas in the Catholic Church; /wiki/List_of_Catholic_basilicas
List of Catholic basilicas - Wikipedia; List of Catholic basilicas; /wiki/Catholic_Church
Catholic Church - Wikipedia; Catholic Church; /wiki/Catholic_Church_(disambiguation)
Catholic Church (disambiguation) - Wikipedia; Catholic Church (disambiguation); /wiki/Catholic_(disambiguation)
...
Anatomical terms of motion - Wikipedia; Anatomical terms of motion; /wiki/Extortion
Extortion - Wikipedia; Extortion; /wiki/Exaction
Exaction - Wikipedia; Exaction; /wiki/Exact_(disambiguation)
Exact - Wikipedia; Exact; /wiki/Exact_(company)
Exact (company) - Wikipedia; Exact (company); /wiki/Besloten_vennootschap
Besloten vennootschap - Wikipedia; Besloten vennootschap; /wiki/Corporate_law
Corporate law - Wikipedia; Corporate law; /wiki/List_of_legal_entity_types_by_country
List of legal entity types by country - Wikipedia; List of legal entity types by country; /wiki/Company_(disambiguation)
Company (disambiguation) - Wikipedia; Company (disambiguation); /wiki/Company
Company - Wikipedia; Company; /wiki/Firm_(disambiguation)
Firm (disambiguation) - Wikipedia; Firm (disambiguation); /wiki/Firm
Company - Wikipedia; Company; /wiki/Capitalism
Capitalism - Wikipedia; Capitalism; /wiki/Capitalism_(disambiguation)
Capitalism (disambiguation) - Wikipedia; Capitalism (disambiguation); /wiki/Economic_liberalism
Economic liberalism - Wikipedia; Economic liberalism; /wiki/Business
Business - Wikipedia; Business; /wiki/Business_(disambiguation)
Business (disambiguation) - Wikipedia; Business (disambiguation); /wiki/Goods_and_services
Goods and services - Wikipedia; Goods and services; /wiki/Business_cycle
Business cycle - Wikipedia; Business cycle; /wiki/Macroeconomics
```

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 = 'insp_QT.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:


webscraping
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProject1/webscraping.py
b'<!DOCTYPE html>\n<html class="no-js" dir="ltr" lang="en-US">\n <head>\n <title>Inspirational Quotes - Motivational Quotes - Leadership Quotes | Pa
<!DOCTYPE html>
<html class="no-js" dir="ltr" lang="en-US">
<head>
<title>
Inspirational Quotes - Motivational Quotes - Leadership Quotes | PassItOn.com
</title>
<meta charset="utf-8"/>
<meta content="text/html; charset=utf-8" http-equiv="content-type"/>
<meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<meta content="width=device-width,initial-scale=1.0" name="viewport"/>
<meta content="The Foundation for a Better Life | Pass It On.com" name="description"/>
<link href="/apple-touch-icon.png" rel="apple-touch-icon" sizes="180x180"/>
<link href="/favicon-32x32.png" rel="icon" sizes="32x32" type="image/png"/>
<link href="/favicon-16x16.png" rel="icon" sizes="16x16" type="image/png"/>
<link href="/site.webmanifest" rel="manifest"/>
<link color="#c8102e" href="/safari-pinned-tab.svg" rel="mask-icon"/>
<meta content="#c8102e" name="msapplication-TileColor"/>
<meta content="#ffffff" name="theme-color"/>
<link crossorigin="anonymous" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1
<link href="/assets/application-2a7a8e6a1c3f628bac9efaab420f5079.css" media="all" rel="stylesheet"/>
<meta content="authenticity_token" name="csrf-param"/>
<meta content="Fqv8/v9RaDX9ByidoU7d9ufmfCer2JwrQoIBole5Ho5++qbGLw2KNFIm2h/+R42pgwv05lhoOBgVnDz0IQIpeg==" name="csrf-token"/>
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async="" src="https://www.googletagmanager.com/gtag/js?id=UA-1179606-29">
</script>

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

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

Program Code:

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

Output:

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

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

Program Code:

```
import nltk
from nltk.util import ngrams
nltk.download('punkt')
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:

```
('this', 'is')
('is', 'very')
('very', 'good')
('good', 'book')
('book', 'to')
('to', 'study')
```

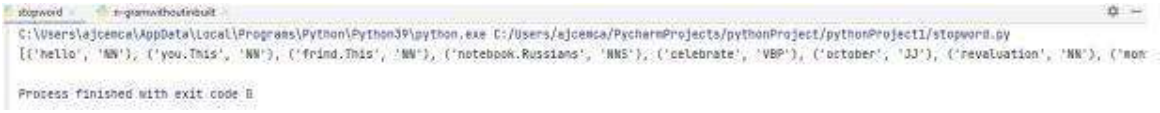
PROGRAM NO: 19 **Date:** 16/02/2022

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

Program Code:

```
import nltk
nltk.download()
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

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

txt = "Sukanya, Rajib and Naba are my good friends. " \
"Sukanya is getting married next year. " \
"Marriage is a big step in one’s life." \
"It is both exciting and frightening. " \
"But friendship is a sacred bond between people." \
"It is a special kind of love between us. " \
"Many of you must have tried searching for a friend " \
"but never found the right one."
tokenized = sent_tokenize(txt)
for i in tokenized:
wordsList = nltk.word_tokenize(i)

wordsList = [w for w in wordsList if not w in stop_words]
tagged = nltk.pos_tag(wordsList)
print(tagged)
```

Output



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

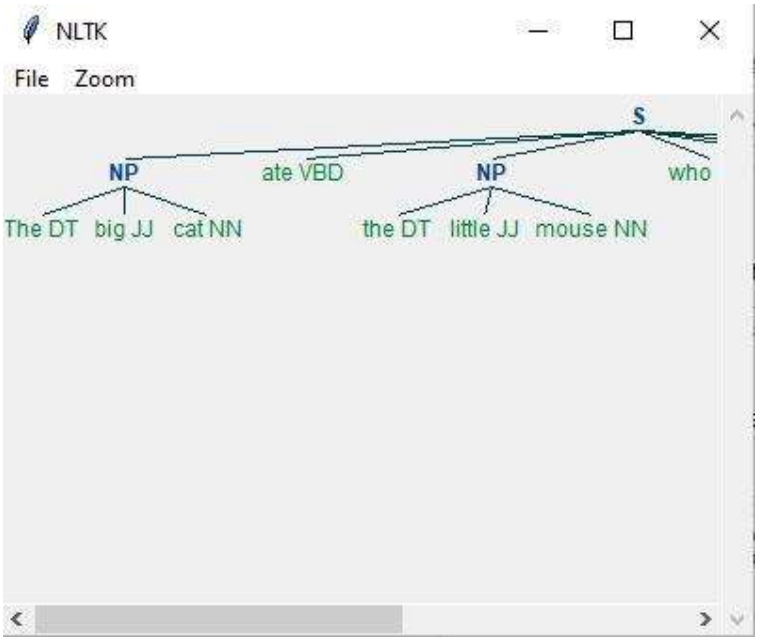
AIM: Write python program for natural language processing which perform chunking.

Program Code:

```
import nltk
new = "The big cat ate the little mouse who was after the fresh cheese"
new_tokens = nltk.word_tokenize(new)
print(new_tokens)

new_tag = nltk.pos_tag(new_tokens)
print(new_tag)

grammer = "NP: {<DT>?<JJ>*<NN>}"
chunkParser = nltk.RegexpParser(grammer)
chunked = chunkParser.parse(new_tag)
print(chunked)
chunked.draw()
```

Output:

```
['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))
```


NLTK
File Zoom
S
NP
ate VBD
NP
who
The DT big JJ cat NN
the DT little JJ mouse NN

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

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

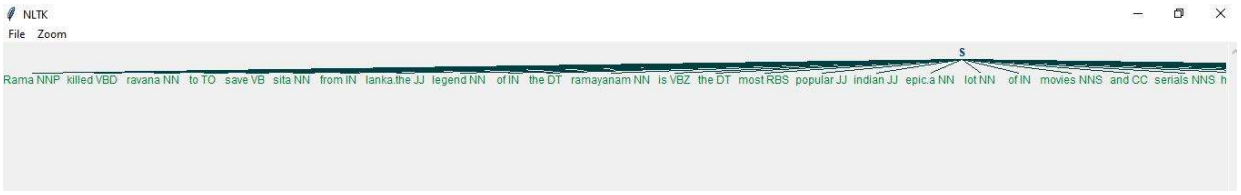
Program Code:

```
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/NN
to/TO
save/VB
sita/NN
from/IN
lanka.the/JJ
legend/NN
of/IN
the/DT
ramayanam/NN
is/VBZ
the/DT
most/RBS
popular/JJ
indian/JJ
epic.a/NN
lot/NN

serials/NNS
have/VBP
already/RB
been/VBN
shot/VBN
in/IN
several/JJ
languages.here/NNS
in/IN
india/NN
based/VBN
on/IN
the/DT
ramayana/NN)


NLTK
File Zoom
S
Rama NNP killed VBD ravana NN to TO save VB sita NN from IN lanka.the JJ legend NN of IN the DT ramayanam NN is VBZ the DT most RBS popular JJ indian JJ epic.a NN lot NN of IN movies NNS and CC serials NNS h