20MCA241 DATA SCIENCE LAB

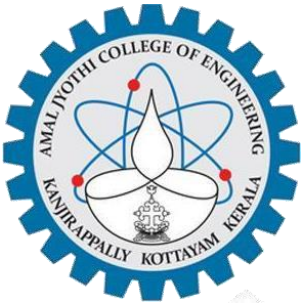
Lab Report SubmittedBy

ANJALI C ABRAHAM

Reg. No.:AJC20MCA-2018

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

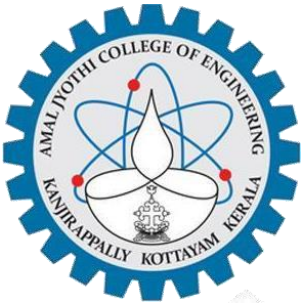
[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

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 **ANJALI C ABRAHAM (Reg.No:AJC20MCA-2018)** 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 2020-22.

Ms. Shelly Shiju George

Lab In-Charge

CONTENT

Date :15/12/2021

PROGRAM NO : 01

AIM : Perform all matrix operation using python

Program Code :

```
import numpy as np
m1 = np.array([[12,6,8], [9,6,7], [5,8,3]])
m2 = np.array([[10,9,15], [13,28,19], [13,5,7]])
m3 = m1 + m2
m4 = m1 - m2
m5 = m1.dot(m2)
m6 = m5.transpose()
m7 = np.divide(m1,m2)
print(" Addition : ",m3)
print(" Subtraction : ",m4)
print(" Multiplication : ",m5)
print(" Transpose : ",m6)
print(" Division : ",m7)
```

Output :

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Script
 Addition :  [[22 15 23]
 [22 34 26]
 [18 13 10]]
 Subtraction :  [[  2  -3  -7]
 [ -4 -22 -12]
 [ -8   3  -4]]
 Multiplication :  [[302 316 350]
 [259 284 298]
 [193 284 248]]
 Transpose :  [[302 259 193]
 [316 284 284]
 [350 298 248]]
 Division :  [[1.2        0.66666667 0.53333333]
 [0.69230769 0.21428571 0.36842105]
 [0.38461538 1.6        0.42857143]]

Process finished with exit code 0
```

Date :15/12/2021

PROGRAM NO : 02

AIM : Program to perform SVD using python

Program Code :

```
from numpy import array
from scipy.linalg import svd
A = array([[3,4,8,1], [7,2,6,9], [5,8,2,4], [7,3,9,5]])
print(A)
B,C,D = svd(A)
print(B)
print(C)
print(D)
```

Output :

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scr
[[3 4 8 1]
 [7 2 6 9]
 [5 8 2 4]
 [7 3 9 5]]
[[-0.38933231 -0.19211242 -0.72541307  0.53412456]
 [-0.5785974  -0.16220582  0.67208289  0.43268802]
 [-0.40690569  0.9064461  -0.08028435 -0.07960933]
 [-0.58997726 -0.33931925 -0.12503957 -0.72191024]]
[21.31948313  6.38539316  6.05791506  1.00402111]
[[-0.53390356 -0.36303419 -0.59616124 -0.47722629]
 [ 0.06972441  0.8050796  -0.58745252  0.04341524]
 [ 0.20661272 -0.42504393 -0.50458483  0.72252543]
 [-0.8169435   0.19847085  0.21186188  0.49832438]]

Process finished with exit code 0
```

Date :15/12/2021

PROGRAM NO : 03

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

Program Code :

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
irisData = load_iris()
A = irisData.data
B = irisData.target
A_train, A_test, B_train, B_test = train_test_split(
    A, B, test_size=.4)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(A_train, B_train)
print(knn.predict(A_test))

S = knn.predict(A_test)
T = accuracy_score(B_test, S)
print('Accuracy : ', T)
```

Output :

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajce
[2 0 0 1 2 1 0 1 2 1 0 0 2 1 2 0 1 1 0 1 2 2 1 2 0 0 1 0 1 2 1 0 0 0 0 1 2
 1 1 2 1 2 1 0 1 1 2 2 0 2 2 2 0 0 1 1 1 2 2 2]
Accuracy :  0.95

Process finished with exit code 0
```

Date :15/12/2021

PROGRAM NO : 04

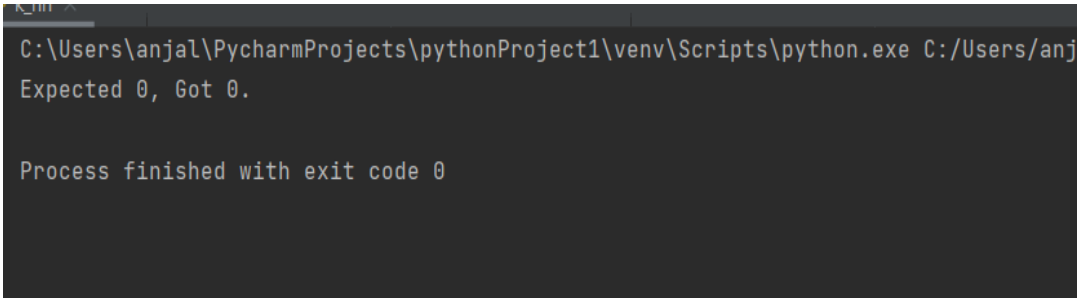
AIM : Program to implement K-NN Classification using any random dataset without using in-build packages

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]
    # 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 :



Date :22/12/2021

PROGRAM NO : 05

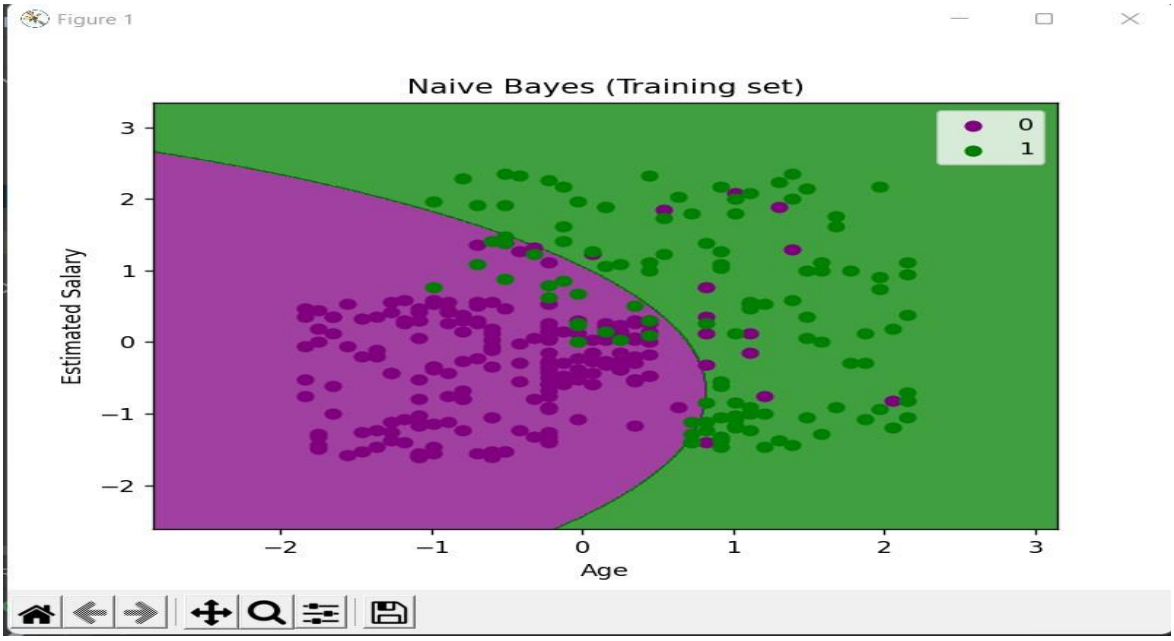
AIM: Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm

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 :




```
C:\Users\anjal\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/anjal/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
```

Date :22/01/2022

PROGRAM NO : 06

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([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_)
y_pred = model.predict(x)
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 :

```
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]
Predicted response:
[ 5. 10. 15. 20. 25. 30.]

Process finished with exit code 1
```

Date :22/01/2022

PROGRAM NO : 07

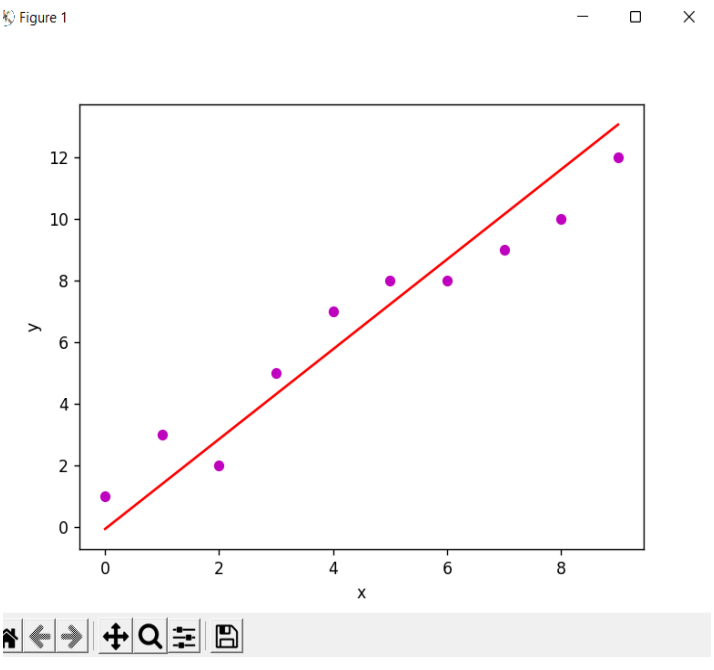
AIM :Program to implement Linear and Multiple regression techniques using any standard dataset available in public domain and evaluate

Program Code :

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


def estimate_coef(x, y):
    n = np.size(x)
   m_x, m_y = np.mean(x), np.mean(y)


    SS_xy = np.sum(y * x - n * m_y * m_x)
   SS_xx = np.sum(x * x - n * m_x * m_x)


    b_1 = SS_xy / SS_xx
   b_0 = m_y - b_1 * m_x
   return b_0, b_1


def plot_regression_line(x, y, b):
     plt.scatter(x, y, color="m", marker="o", s=30)
   y_pred = b[0] + b[1] * x
   plt.plot(x, y_pred, color="r")
   plt.xlabel('x')
   plt.ylabel('y')
   plt.show()


def main():
```

```
    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])

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

    plot_regression_line(x, y, b)

if__name__== "_main_":
    main()
```

Output :




```
C:\Users\anjal\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/anjal/Py
Estimated coefficients are:
b_0 = -0.05862068965517242
b_1 = 1.457471264367816
```

Date :05/01/2022

PROGRAM NO : 08

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

Program Code :

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

Output :

[107.2087328]

Date :05/01/2022

PROGRAM NO : 09

AIM :Program to implement multiple linear regression techniques using boston dataset available in the public domain and evaluate accuracy and plotting point.

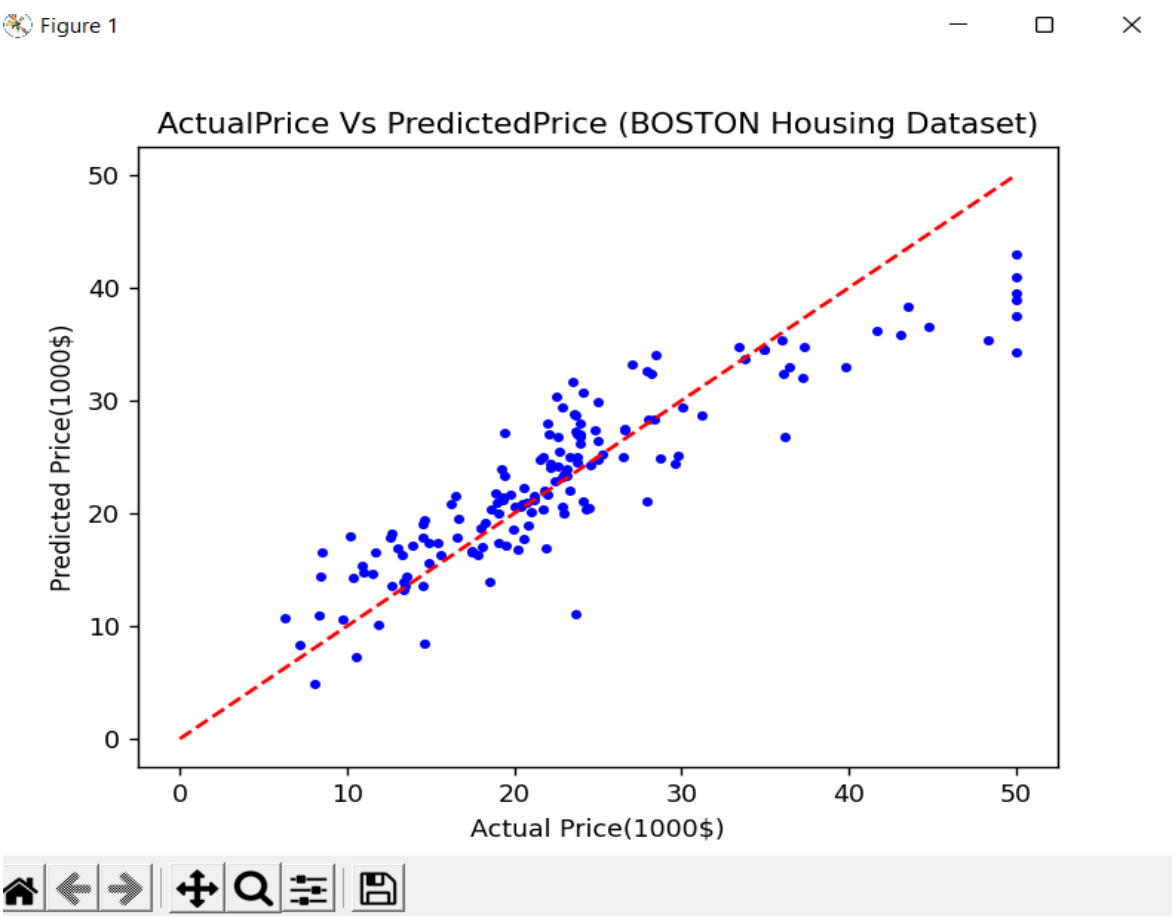
Program Code :

```
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)
print('Coefficients are:\n', reg.coef_)
print('\nIntercept : ', reg.intercept_)

print('Variance score: ', reg.score(X_test, y_test))
print("Mean squared error: %.2f" % mean_squared_error(y_test, predicted))

expected = y_test

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]

Intercept :  46.39649387182355
Variance score:  0.7836295385076291
Mean squared error: 19.83
```



Date :22/12/2021

PROGRAM NO : 10

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('one.png')

target = df['species']
df1 = df.copy()
```

```
df1 = df1.drop('species', axis=1)
print(df1.shape)
print(df1.head())
x = df1
print(target)
le = LabelEncoder()
target = le.fit_transform(target)
print(target)
y = target
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

print("Training split input- ", X_train.shape)
print("Testing split input- ", X_test.shape)
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)

print('Decision Tree Classifier Created')
y_pred = dtree.predict(X_test)
print('Classification report - \n', classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5, 5))
sns.heatmap(data=cm, 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, class_names=['setosa', 'vercicolor', 'verginica'], filled=True, precision=4, rounded=True)

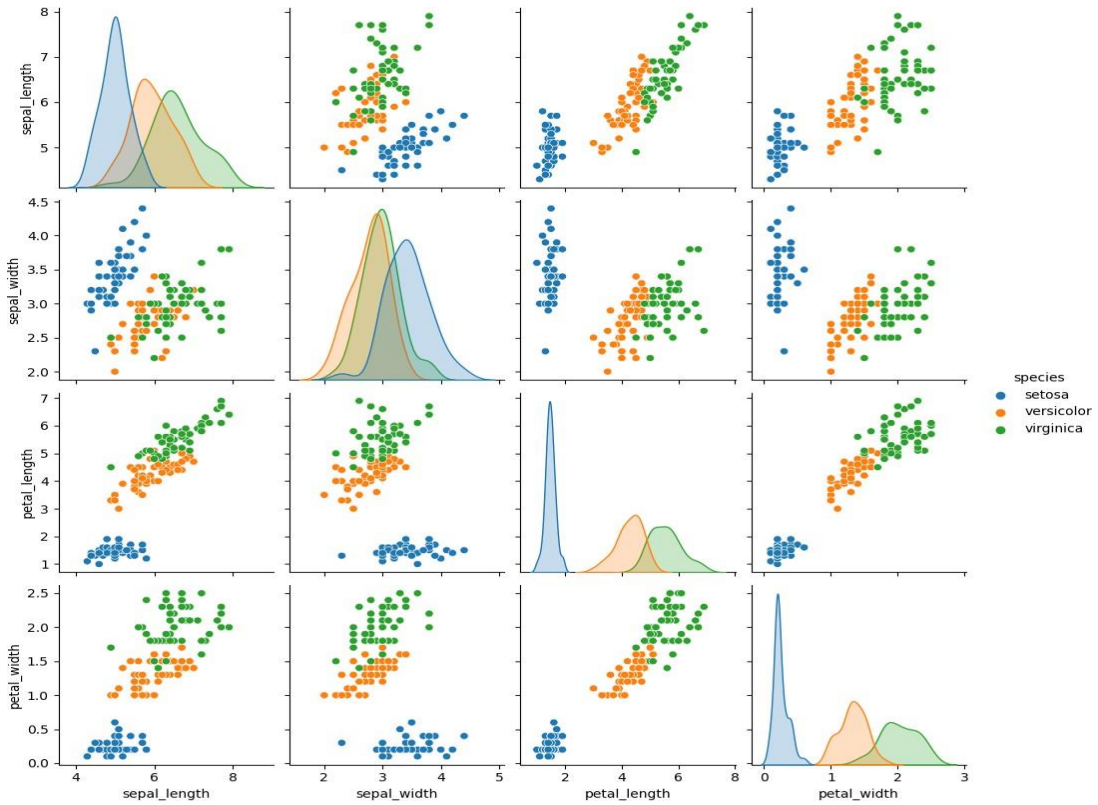
plt.savefig('tree.png')

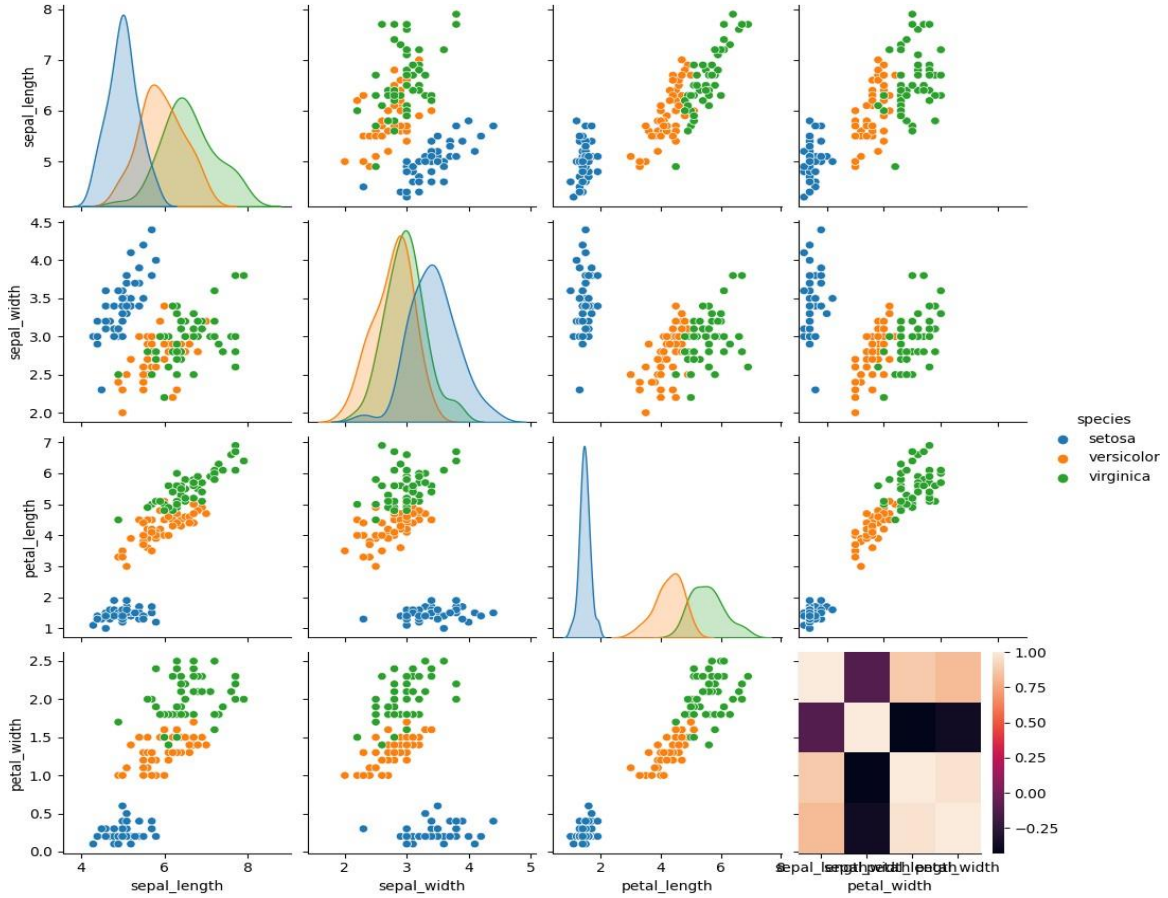
Output :

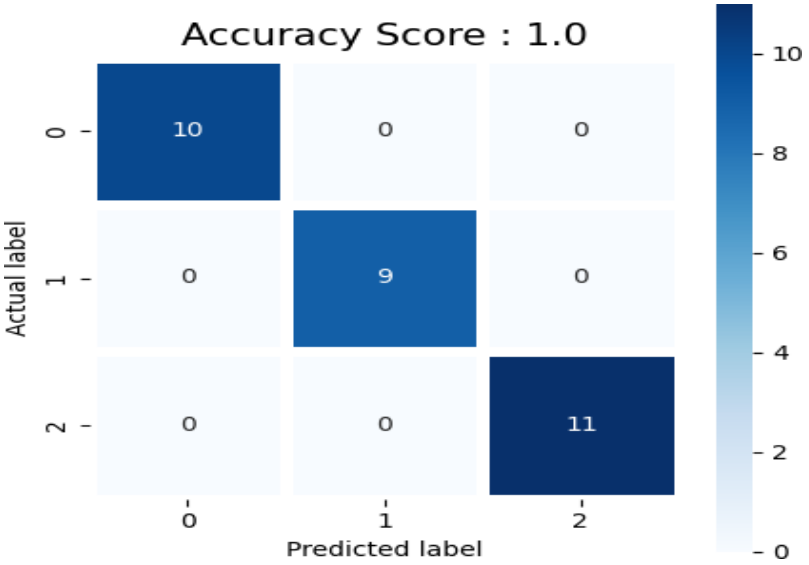
```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe "C
Connected to pydev debugger (build 212.5457.59)
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
(150, 5)
(150, 4)
```

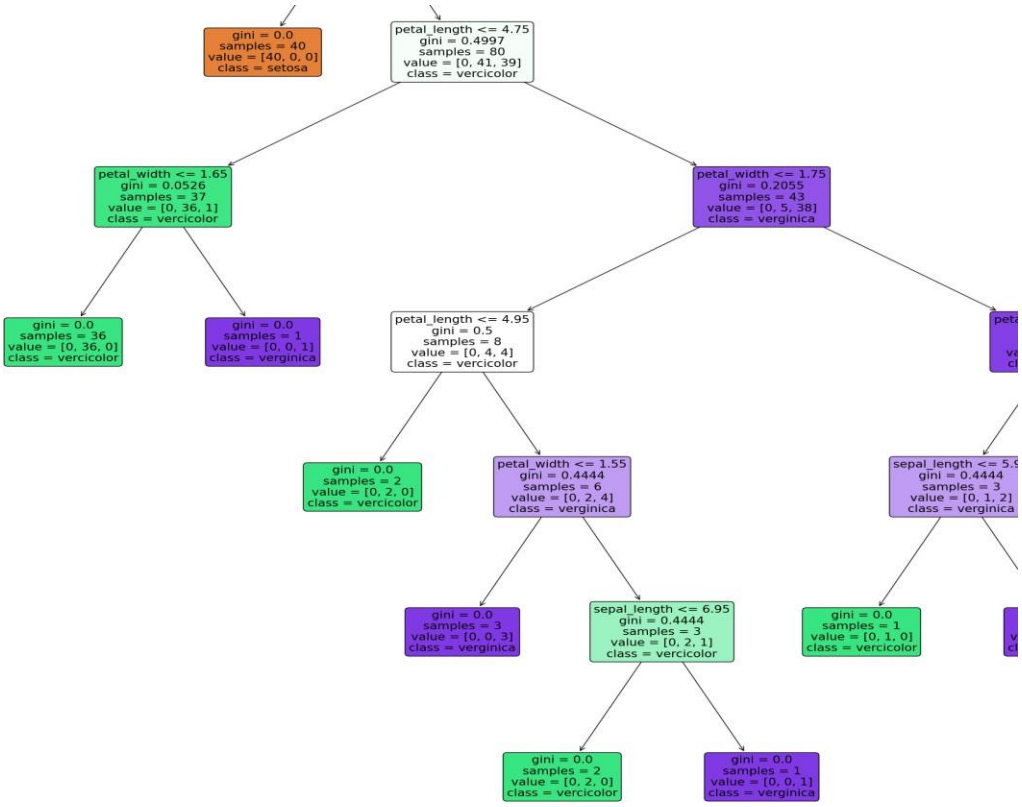
```
   sepal_length  sepal_width  petal_length  petal_width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
        ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
Training split input-  (120, 4)
Testing split input-  (30, 4)
Decision Tree Classifier Created
Classification 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


Process finished with exit code 0
```


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica


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


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 = vercicolor
petal_width <= 1.65
gini = 0.0526
samples = 37
value = [0, 36, 1]
class = vercicolor
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 = vercicolor
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 = vercicolor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicolor
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
sepal_length <= 6.95
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = vercicolor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = verginica

Date :05/01/2022

PROGRAM NO : 11

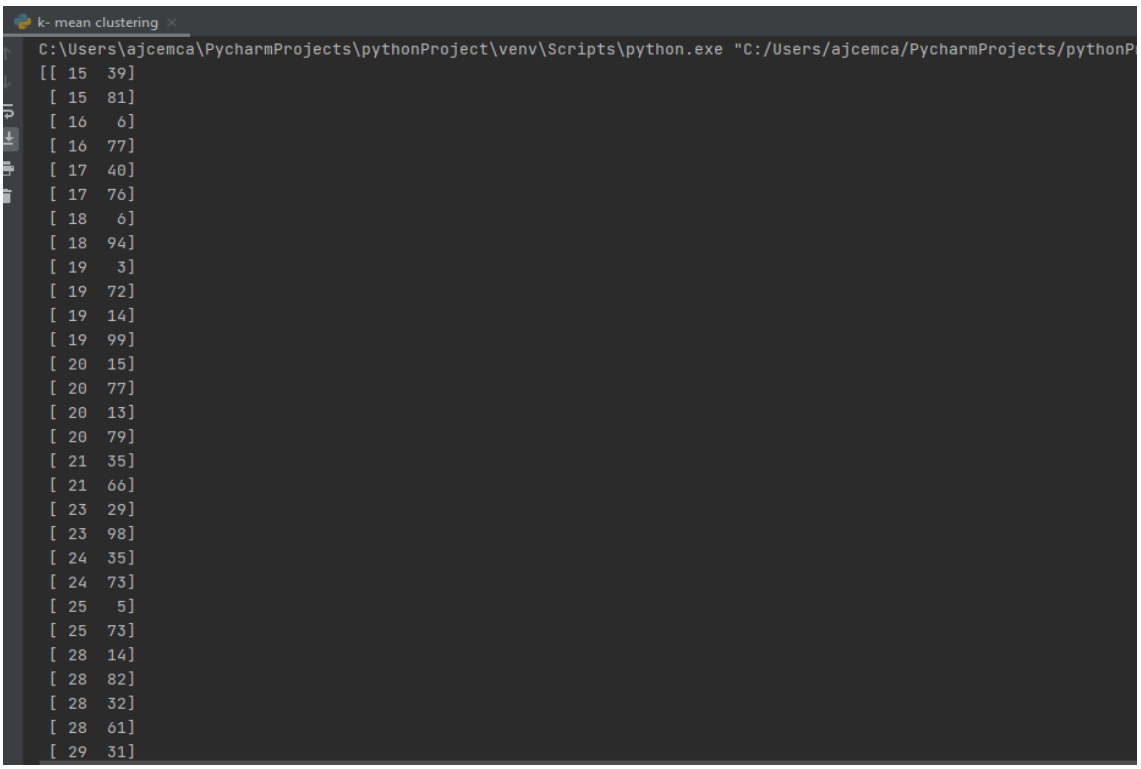
AIM :Program to implement k-means clustering technique using any standard dataset available in the public domain

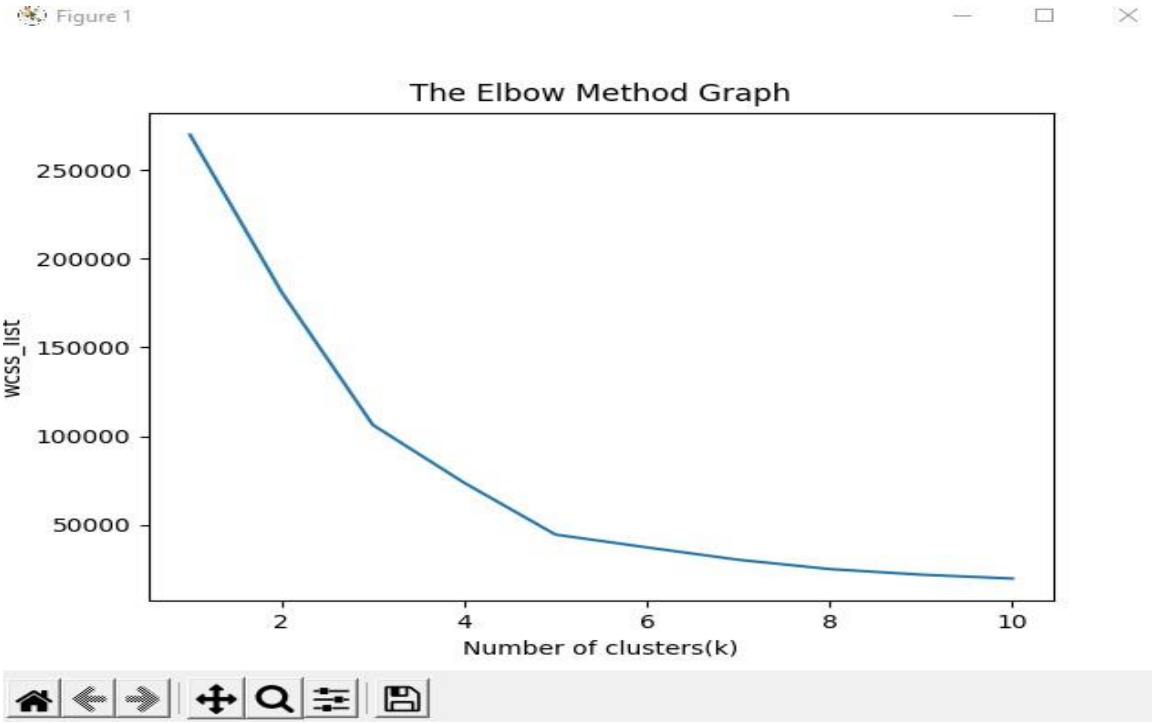
Program Code :

```
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=60, c='red', label='Cluster1')
mtp.scatter(X[y_predict == 1, 0], X[y_predict == 1, 1], s=60, c='blue', label='Cluster2')
mtp.scatter(X[y_predict == 2, 0], X[y_predict == 2, 1], s=60, c='green', label='Cluster3')
mtp.scatter(X[y_predict == 3, 0], X[y_predict == 3, 1], s=60, c='violet', label='Cluster4')
mtp.scatter(X[y_predict == 4, 0], X[y_predict == 4, 1], s=60, c='yellow', label='Cluster5')
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100, c='black',
label='Centroids')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

Output :






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

Process finished with exit code 0
```

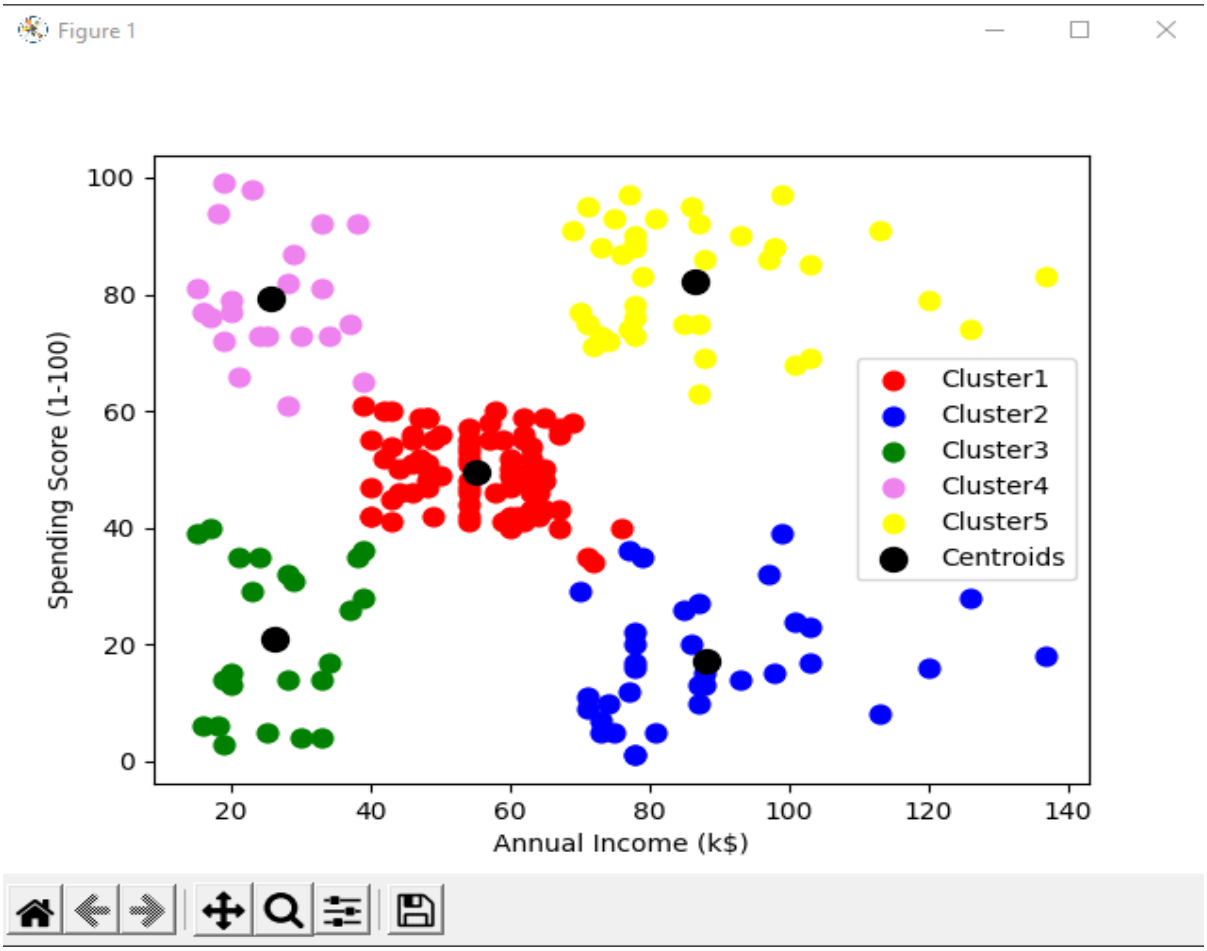


Figure 1
Spending Score (1-100)
Annual Income (k$)
Cluster1
Cluster2
Cluster3
Cluster4
Cluster5
Centroids

Date :05/01/2022

PROGRAM NO : 12

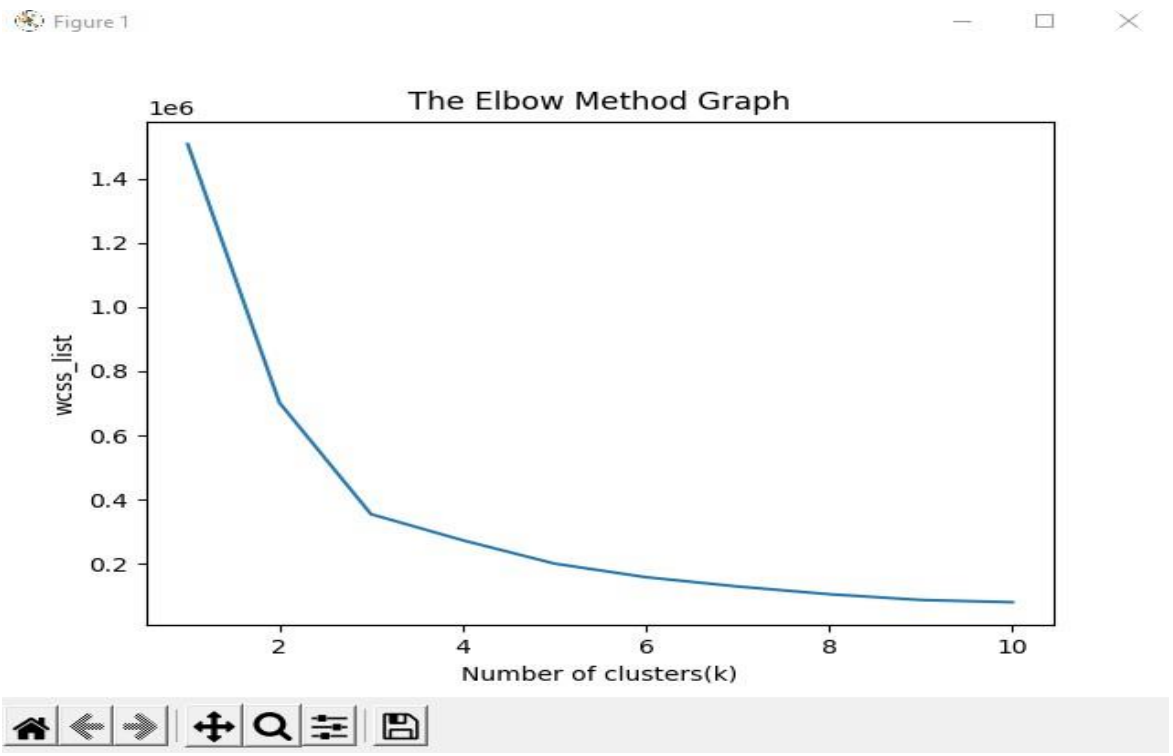
AIM :Program to implement k-means clustering technique using any standard dataset available in the public domain

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=60, c='red', label='Cluster1')
mtp.scatter(X[y_predict == 1, 0], X[y_predict == 1, 1], s=60, c='blue', label='Cluster2')
mtp.scatter(X[y_predict == 2, 0], X[y_predict == 2, 1], s=60, c='green', label='Cluster3')
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100, c='black',
label='Centroids')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()

mtp.show()
```

Output :



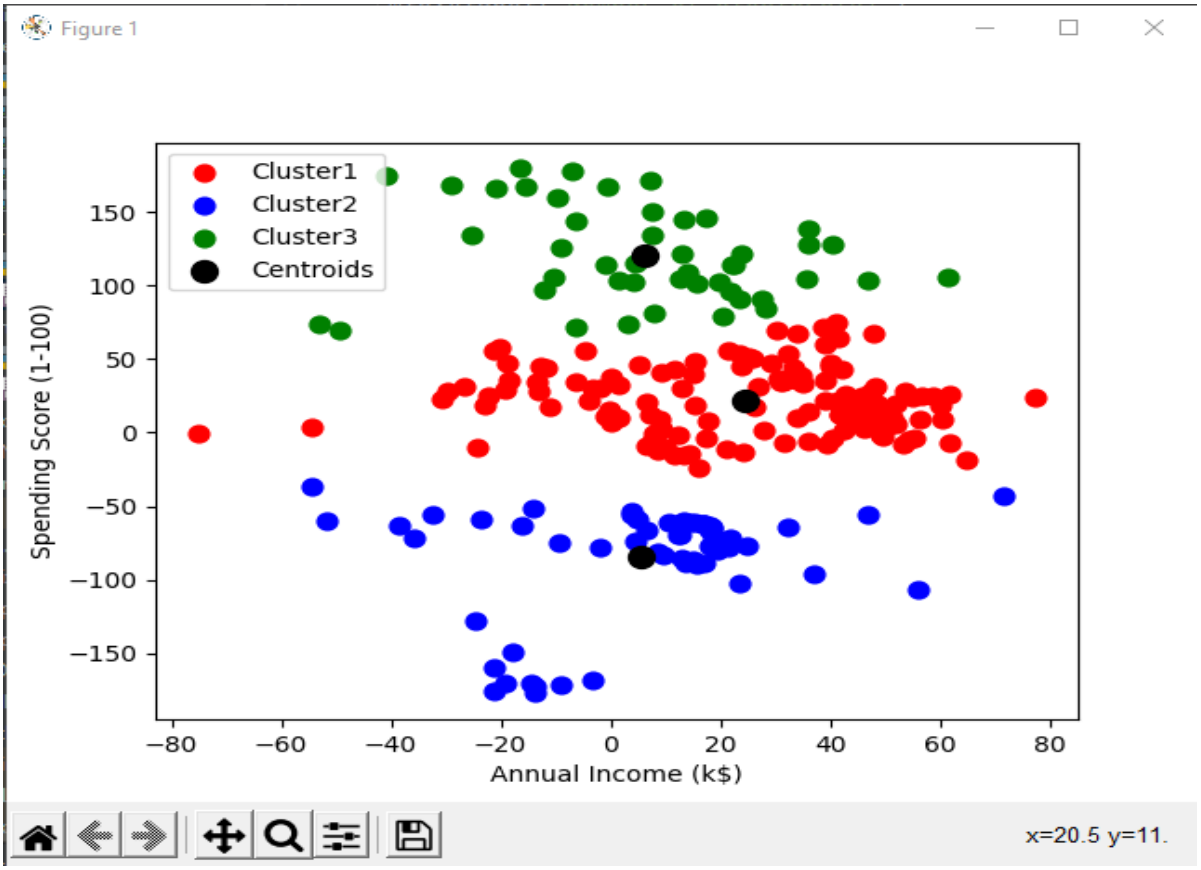


Figure 1
Cluster1
Cluster2
Cluster3
Centroids
Spending Score (1-100)
Annual Income (k$)
150
100
50
0
−50
−100
−150
−80
−60
−40
−20
0
20
40
60
80
x=20.5 y=11.

Date :02/02/2022

PROGRAM NO : 13

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', 'Pillover', '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_rows):
        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 :

```
(60000, 28, 28) (10000, 28, 28)
```

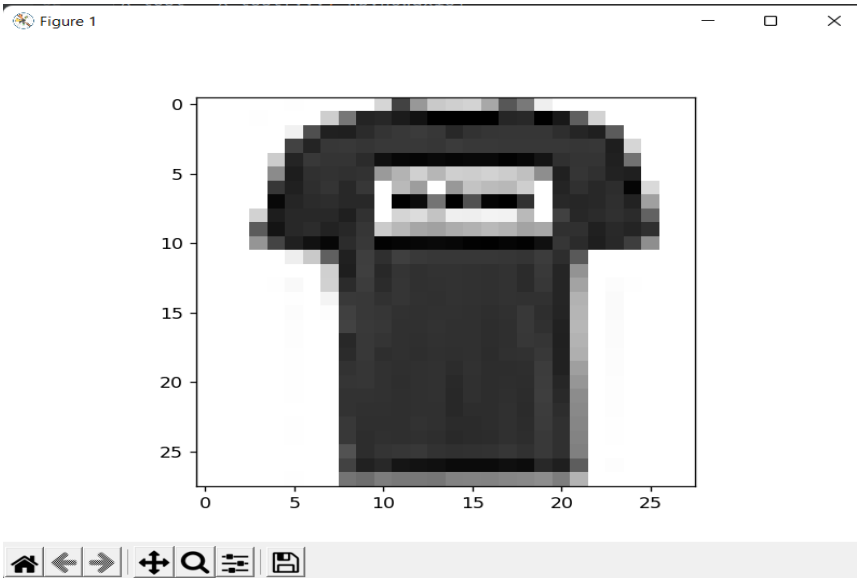
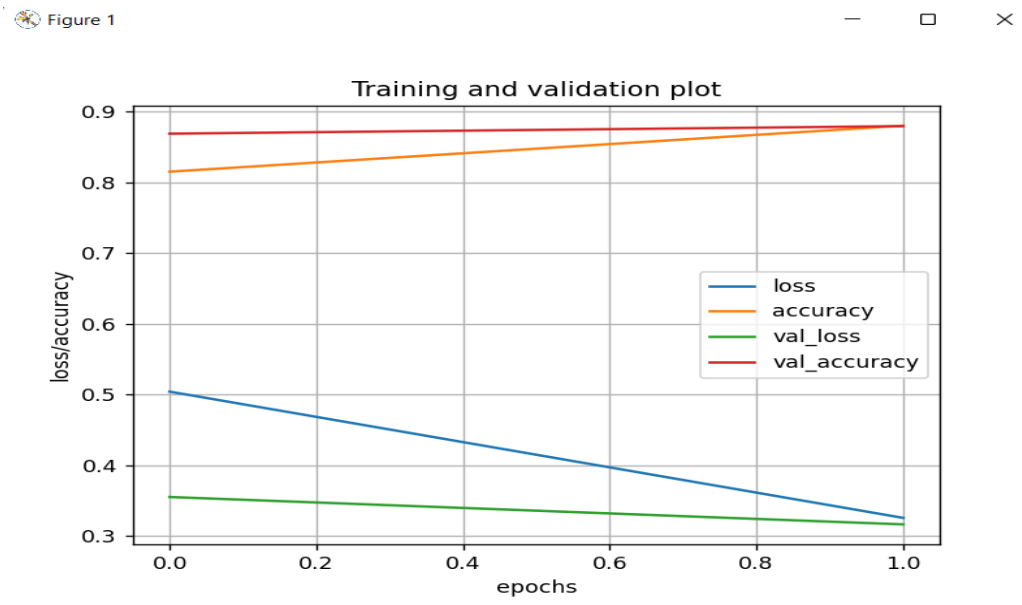


Figure 1
0
5
10
15
20
25
0
5
10
15
20
25


Figure 1
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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        1600

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

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

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

=================================================================
Total params: 38,560
Trainable params: 38,560
Non-trainable params: 0
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        1600

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

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

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

 flatten (Flatten)           (None, 288)               0

 dense (Dense)               (None, 128)               36992

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 84,458
Trainable params: 84,458
Non-trainable params: 0
_________________________________________________________________
```




```
Epoch 1/2
1688/1688 [==============================] - 100s 58ms/step - loss: 0.5043 - accuracy: 0.8151 - val_loss: 0.3554 - val_accuracy: 0.8688
Epoch 2/2
1688/1688 [==============================] - 82s 49ms/step - loss: 0.3259 - accuracy: 0.8802 - val_loss: 0.3167 - val_accuracy: 0.8795
313/313 [==============================] - 4s 14ms/step - loss: 0.3361 - accuracy: 0.8753
Test loss : 0.33612382411956787, Test Accuracy : 0.8752999901771545

Process finished with exit code 0
```

Date :16/02/2022

PROGRAM NO : 14

AIM : Implement a simple web crawler

Program Code :

```
import requests
import lxml
from bs4 import BeautifulSoup
url = "https://www.rottentomatoes.com/top/bestofrt/"
headers = {
   'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 QIHU 360SE'
}
f = requests.get(url,headers = headers)
movies_lst = []
soup = BeautifulSoup(f.content, 'html.parser')
movies = soup.find('table', {
   'class': 'table'
}) .find_all('a')
print(movies)
num =0
for anchor in movies:
   urls = 'https://www.rottentomatoes.com' + anchor['href']
   movies_lst.append(urls)
print(movies_lst)
num += 1
movie_url = urls

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

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/venv/web_crawl/wdmwc.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/blackkklansman">
            BlacKkKlansman (2018)</a>, <a class="unstyled articleLink" href="/m/get_out">
            Get Out (2017)</a>, <a class="unstyled articleLink" href="/m/the_irishman">
            The Irishman (2019)</a>, <a class="unstyled articleLink" href="/m/godfather">
            The Godfather (1972)</a>, <a class="unstyled articleLink" href="/m/mad_max_fury_road">
            Mad Max: Fury Road (2015)</a>, <a class="unstyled articleLink" href="/m/spider_man_into_the_spider_verse">
            Spider-Man: Into the Spider-Verse (2018)</a>, <a class="unstyled articleLink" href="/m/moonlight_2016">
            Moonlight (2016)</a>, <a class="unstyled articleLink" href="/m/sunset_boulevard">
            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">
```

```
            Laura (1944)</a>, <a class="unstyled articleLink" href="/m/spider_man_far_from_home">
            Spider-Man: Far From Home (2019)</a>, <a class="unstyled articleLink" href="/m/incredibles_2">
            Incredibles 2 (2018)</a>, <a class="unstyled articleLink" href="/m/zootopia">
            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_
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
```

Date :16/02/2022

PROGRAM NO : 15

AIM : Implement a simple web crawler

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}; {soup.h1.text}; {link["href"]}\n')
                        crawler(new_link)
                    except:
                        continue

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

Output :

```
Wikipedia, the free encyclopedia; Main Page; /wiki/WikipediaWikipedia - Wikipedia; Wikipedia; /wiki/Main_PageW
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)
```

```
W. S. Van Dyke - Wikipedia; W. S. Van Dyke; /wiki/San_Diego,_California
San Diego - Wikipedia; San Diego; /wiki/San_Diego_County,_California
San Diego County, California - Wikipedia; San Diego County, California; /wiki/List_of_counties_in_California
List of counties in California - Wikipedia; List of counties in California; /wiki/List_of_United_States_counties_and_
List of United States counties and county equivalents - Wikipedia; List of United States counties and county equivale
County (United States) - Wikipedia; County (United States); /wiki/County
County - Wikipedia; County; /wiki/County_(disambiguation)
County (disambiguation) - Wikipedia; County (disambiguation); /wiki/Counties_of_China
Counties of China - Wikipedia; Counties of China; /wiki/Simplified_Chinese_characters
Simplified Chinese characters - Wikipedia; Simplified Chinese characters; /wiki/Logographic
Logogram - Wikipedia; Logogram; /wiki/Logography_(printing)
The Times - Wikipedia; The Times; /wiki/The_Times_(disambiguation)
The Times (disambiguation) - Wikipedia; The Times (disambiguation); /wiki/The_Times
The Times - Wikipedia; The Times; /wiki/Daily_newspaper
Newspaper - Wikipedia; Newspaper; /wiki/Journalism
Journalism - Wikipedia; Journalism; /wiki/Reportage_(disambiguation)
Reportage (disambiguation) - Wikipedia; Reportage (disambiguation); /wiki/Reportage
Reportage (disambiguation) - Wikipedia; Reportage (disambiguation); /wiki/Reportage_(album)
Reportage (album) - Wikipedia; Reportage (album); /wiki/Duran_Duran
Duran Duran - Wikipedia; Duran Duran; /wiki/Duran_Duran_(disambiguation)
Duran Duran (disambiguation) - Wikipedia; Duran Duran (disambiguation); /wiki/Duran_Duran_(1981_album)
Duran Duran (1981 album) - Wikipedia; Duran Duran (1981 album); /wiki/Chipping_Norton_Recording_Studios
Chipping Norton Recording Studios - Wikipedia; Chipping Norton Recording Studios; /wiki/Recording_studio
Recording studio - Wikipedia; Recording studio; /wiki/Film_studio
Film studio - Wikipedia; Film studio; /wiki/Studio
```

Date :16/02/2022

PROGRAM NO : 16

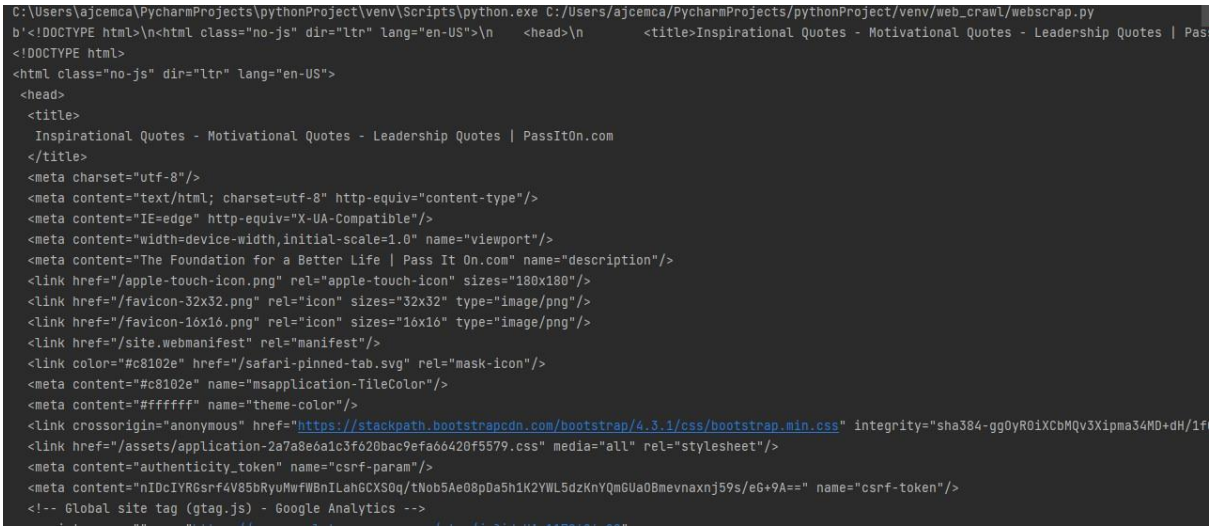
AIM : Implement a program to scrap the web page of any popular website

Program Code :

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

URL = "http://www.values.com/inspirational-quotes"
r = requests.get(URL)
print(r.content)

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

quotes = []

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

for row in table.findAll('div', attrs={'class': 'col-6 col-lg-3 text-center margin-30px-bottom
sm-margin-30px-top'}):

    quote = {}
    quote['theme'] = row.h5.text
    quote['url'] = row.a['href']
    quote['img'] = row.img['src']
    quote['lines'] = row.img['alt'].split("#")[0]
    quote['author'] = row.img['alt'].split("#")[1]
```

```
    quotes.append(quote)


filename = 'inspirational_quotes.csv'
with open(filename, 'w', newline='') as f:
    w = csv.DictWriter(f,['theme', 'url', 'img', 'lines','author'])
    w.writeheader()
    for quote in quotes:
        w.writerow(quote)
```

Output :



```
theme,url,img,lines,author
LOVE,/inspirational-quotes/7444-where-there-is-love-there-is-life,https://assets.passiton.com/quotes/quote_artwork/74
LOVE,/inspirational-quotes/7439-at-the-touch-of-love-everyone-becomes-a-poet,https://assets.passiton.com/quotes/quote
FRIENDSHIP,/inspirational-quotes/8304-a-friend-may-be-waiting-behind-a-stranger-s-face,https://assets.passiton.com/qu
FRIENDSHIP,/inspirational-quotes/3331-wherever-we-are-it-is-our-friends-that-make,https://assets.passiton.com/quotes/
FRIENDSHIP,/inspirational-quotes/8303-find-a-group-of-people-who-challenge-and,https://assets.passiton.com/quotes/quo
FRIENDSHIP,/inspirational-quotes/8302-there-s-not-a-word-yet-for-old-friends-who-ve,https://assets.passiton.com/quote
FRIENDSHIP,/inspirational-quotes/7435-there-are-good-ships-and-wood-ships-ships-that,https://assets.passiton.com/quot
PERSISTENCE,/inspirational-quotes/6377-at-211-degrees-water-is-hot-at-212-degrees,https://assets.passiton.com/quotes/
PERSISTENCE,/inspirational-quotes/8301-the-key-of-persistence-opens-all-doors-closed,https://assets.passiton.com/quot
PERSISTENCE,/inspirational-quotes/7918-you-keep-putting-one-foot-in-front-of-the,https://assets.passiton.com/quotes/q
PERSISTENCE,/inspirational-quotes/7919-to-persist-with-a-goal-you-must-treasure-the,https://assets.passiton.com/quote
PERSISTENCE,/inspirational-quotes/8300-failure-cannot-cope-with-persistence,https://assets.passiton.com/quotes/quote_
INSPIRATION,/inspirational-quotes/8298-though-no-one-can-go-back-and-make-a-brand-new,https://assets.passiton.com/quo
INSPIRATION,/inspirational-quotes/8297-a-highly-developed-values-system-is-like-a,https://assets.passiton.com/quotes/
INSPIRATION,/inspirational-quotes/7066-just-don-t-give-up-trying-what-you-really-want,https://assets.passiton.com/quo
INSPIRATION,/inspirational-quotes/8296-when-we-strive-to-become-better-than-we-are,https://assets.passiton.com/quotes/
INSPIRATION,/inspirational-quotes/8299-the-most-important-thing-is-to-try-and-inspire,https://assets.passiton.com/quo
OVERCOMING,/inspirational-quotes/6828-bad-things-do-happen-how-i-respond-to-them,https://assets.passiton.com/quotes/q
OVERCOMING,/inspirational-quotes/8294-show-me-someone-who-has-done-something,https://assets.passiton.com/quotes/quote
OVERCOMING,/inspirational-quotes/6137-its-not-the-load-that-breaks-you-down-its-the,https://assets.passiton.com/quote
OVERCOMING,/inspirational-quotes/6805-getting-over-a-painful-experience-is-much-like,https://assets.passiton.com/quot
OVERCOMING,/inspirational-quotes/8293-if-you-cant-fly-then-run-if-you-cant-run-then,https://assets.passiton.com/quote
CREATIVITY,/inspirational-quotes/5577-the-creative-is-the-place-where-no-one-else-has,https://assets.passiton.com/quo
CREATIVITY,/inspirational-quotes/7345-creativity-is-allowing-yourself-to-make,https://assets.passiton.com/quotes/quot
CREATIVITY,/inspirational-quotes/7487-creativity-requires-the-courage-to-let-go-of,https://assets.passiton.com/quotes/
```

Date :16/02/2022

PROGRAM NO : 17

AIM : Program for National Program Language Processing 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 + WordsToCombine])
    return output

x=generate_ngrams(text='this is very good book to study', WordsToCombine=3)
print(x)
```

Output :

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

Process finished with exit code 0
```

Date :16/02/2022

PROGRAM NO : 18

AIM : Program for National Program Language Processing N-grams in-built)

Program Code :

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

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

Output :

```
C:\Users\ajcemca\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject1/main.py
('this', 'is')
('is', 'very')
('very', 'good')
('good', 'book')
('book', 'to')
('to', 'study')

Process finished with exit code 0
```

Date :16/02/2022

PROGRAM NO : 19

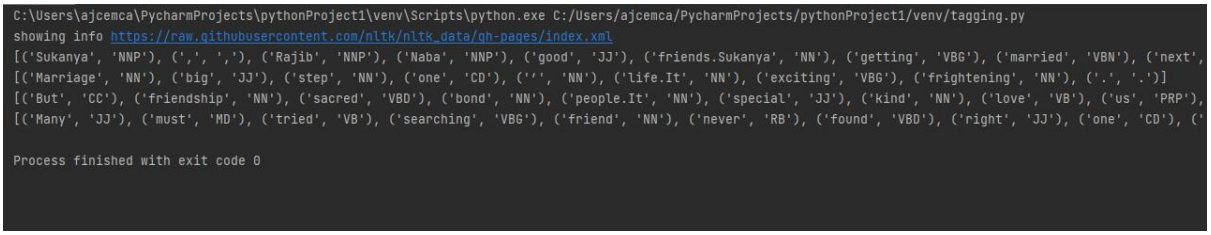
AIM : Program for National Program Language Processing part of speech tagging

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 :



Date :23/02/2022

PROGRAM NO : 20

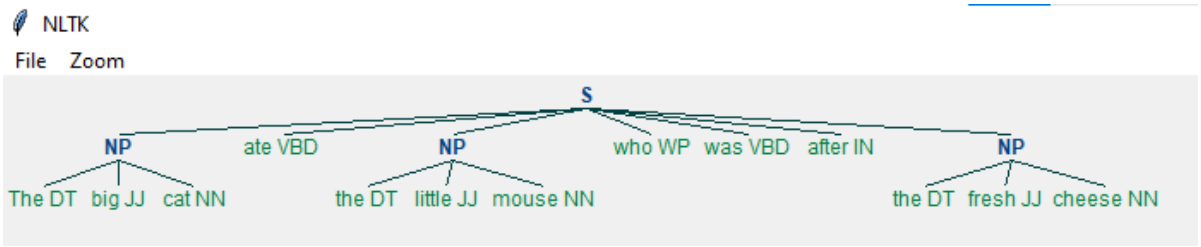
AIM : Write python program for Natural Program Language Processing chunking

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\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/venv/chunking/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'),
(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))
```



Date :23/02/2022

PROGRAM NO : 21

AIM : Write python program for Natural Program Language Processing 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/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
```