Date:24/11/2021

Program no:1

Aim: Perform all Matrix operations in python.

Program:

```
import numpy as num

m1 = num.array([[5, 8], [10, 12]]) m2
= num.array([[4, 3], [7, 9]])

print("First  matrix : ", m1)
print("Second  matrix : ", m2)
addition= num.add(m1,m2)
print("Addition  of matrices is: ", addition)
Difference=num.subtract(m1,m2)
print("Difference of matrix is: ",Difference)
Division=num.divide(m1,m2)
print("Matrix after division: ",Division)
product=num.multiply(m1,m2)
print("Product of matrix is: ",product)
squarerootm1=num.sqrt(m1)
print("Square root of first matrix is: ",squarerootm1)
squarerootm2=num.sqrt(m2)
print("Squareroot of second matrix is: ",squarerootm2)
sum1=num.sum(m1)
print("Sum of first matrix m1  is: ",sum1) sum2=num.sum(m2)
print("sum of second matrix m2 is: ",sum2) dot=num.dot(m1,m2)
```

```
print("Matrix after dot operation: ",dot)
print("transpose of matrix m1 is: ",m1.T)
print("transpose of second matrix is: ",m2.T)
```

Output:

```
First  matrix :  [[ 5  8]
 [10 12]]
Second  matrix :  [[4 3]
 [7 9]]
Addition  of matrices is:  [[ 9 11]
 [17 21]]
Difference of matrix is:  [[1 5]
 [3 3]]
Matrix after division:  [[1.25       2.66666667]
 [1.42857143 1.33333333]]
Product of matrix is:  [[ 20  24]
 [ 70 108]]
Square root of first matrix is:  [[2.23606798 2.82842712]
 [3.16227766 3.46410162]]
Squareroot of second matrix is:  [[2.         1.73205081]
 [2.64575131 3.        ]]
Sum of first matrix m1  is:  35
sum of second matrix m2 is:  23
Matrix after dot operation:  [[ 76  87]
 [124 138]]
transpose of matrix m1 is:  [[ 5 10]
 [ 8 12]]
transpose of second matrix is:  [[4 7]
 [3 9]]
```

Date:20/12/2021

Program no:2

Aim: Perform SVD(Singular Value Decomposition)

Program:

```
import numpy as np from
scipy.linalg import svd
a=np.array([[1,2,5,5,5],[2,6,6,6,6],[6,5,8,8,8],[7,8,9,9,9]])
print(a)
U,S,VT=svd(a)
print(f"Decomposed mattrix is\n{U}\n")
print(f"inverse mattrix is \n{S}\n")
print(f"Transpose mattrix is \n{VT}\n")
```

Output:

```
C:\Users\AMAL\PycharmProjects\pythonProject2\venv\Scripts\python.exe
[[1 2 5 5 5]
 [2 6 6 6 6]
 [6 5 8 8 8]
 [7 8 9 9 9]]
Decomposed mattrix is
[[-0.29976923 -0.70827835  0.49111594 -0.40900529]
 [-0.41725461 -0.50034844 -0.64875677  0.39327432]
 [-0.55243902  0.25166615  0.52358188  0.59777696]
 [-0.65639022  0.42971778 -0.25255061 -0.56631501]]

inverse mattrix is
[2.86593163e+01 3.04716127e+00 2.52158599e+00 1.00825876e-16]

Transpose mattrix is
[[-3.15556870e-01 -3.87880258e-01 -4.99990922e-01 -4.99990922e-01
  -4.99990922e-01]
 [ 9.21856727e-01  9.10439613e-02 -2.17478867e-01 -2.17478867e-01
  -2.17478867e-01]
 [ 2.24953413e-01 -9.17202214e-01  1.89856138e-01  1.89856138e-01
   1.89856138e-01]
 [ 9.25762299e-17 -1.04045554e-16 -8.16496581e-01  4.08248290e-01
   4.08248290e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 -7.07106781e-01
   7.07106781e-01]]
```

Date:20/12/2021

Program no:3

Aim: Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm.

Program:

```
from sklearn.datasets import load_iris
from sklearn.neighbors import
KNeighborsClassifier from sklearn.model_selection
import train_test_split from sklearn.metrics import
accuracy_score import matplotlib.pyplot as pl
idata=load_iris() X=idata.data
Y=idata.target
#print(X)
#print(Y)
X_tr,X_ts,Y_tr,Y_ts=train_test_split(X,Y,test_size=0.4,random_state=101
) knn=KNeighborsClassifier(n_neighbors=8) knn.fit(X_tr,Y_tr)
Y_pred=knn.predict(X_ts) p=[[6.5,9., 9.2,2.]]
print(f"prediction for [6.5,9., 9.2,2.] is {knn.predict(p)}") print(f"accuracy
of the algorithm {accuracy_score(Y_ts,Y_pred)}")
```

Output

```
C:\Users\AMAL\PycharmProjects\pythonProject2\venv\Scripts\python.exe
prediction for [6.5,9., 9.2,2.] is [2]
accuracy of the algorithm 0.9666666666666667

Process finished with exit code 0
```

Date:20/12/2021

Program no:4

Aim: Program to implement k-NN classification using any random data set without using any inbuilt packages

Program:

```
from math import sqrt

def e_dis(r1,r2):
   dist=0.0     for i in
range(len(r1)-1):
dist+=(r1[i]-r2[i])**2
return sqrt(dist)

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

def predict_classif(train,test_row,num_neig):
neighbors = get_ne(train,test_row,num_neig)
out_val=[row[-1] for row in neighbors]
prediction=max(set(out_val),key=out_val.count)
return prediction
```

```
dataset=[[2.734,2.55,0],
        [1.45,3.36,0],
        [2.334, 2.355, 0],
        [1.45, 3.36, 0],
        [2.334, 2.55, 0],
        [1.45, 3.336, 0],
        [3.334, 3.55, 1],
        [1.45, 3.36, 1],
        [3.734, 4.55, 1],
        [3.45, 4.36, 1],
        [4.734, 5.55, 1],
        [3.45, 5.36, 1]
        ]

prediction=predict_classif(dataset,dataset[0],3) print('Excpected
%d,Got %d'%(dataset[0][-1],prediction))
```

Output

```
C:\Users\AMAL\PycharmProjects\pythonProject2\venv\Scripts\python.exe
Excpected 0,Got 0

Process finished with exit code 0
```

Date:20/12/2021

Program - 5

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

Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,accuracy_score

dataset=pd.read_csv('Social_Network_Ads.csv')
x=dataset.iloc[:,[2,3]].values y=dataset.iloc[:,-
1].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
sc=StandardScaler() x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test) classifier=GaussianNB()
classifier.fit(x_train,y_train) y_pred=classifier.predict(x_test)
print(y_pred)

ac = accuracy_score(y_test,y_pred) print(ac)
```

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
```

Date:08/01/2022

Program no:6

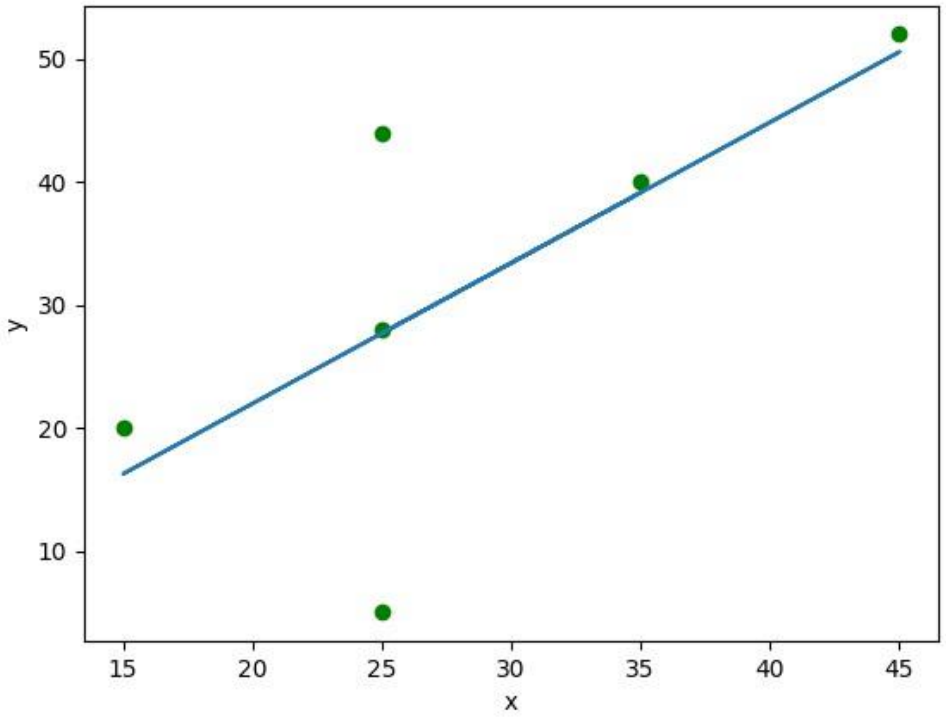
Aim: Program to implement linear and multiple regression techniques using any standard dataset available in the public domain.

Program:

```
import numpy as np from sklearn.linear_model
import LinearRegression
x=np.array([5,12,25,35,45,55]).reshape((-1,1))
y=np.array([5,20,16,32,22,38]) print(x)
model=LinearRegression() model.fit(x,y)
r_sq=model.score(x,y) print('coefficent of
determination',r_sq)
print("intercept",model.intercept_)
print("slope",model.coef_)
y_pred=model.predict(x) print('predicted
value',y_pred) import matplotlib.pyplot as pl
pl.scatter(x,y) pl.plot(x,y_pred)
```

Output:

```
C:\Users\AMAL\PycharmProjects\pythonProject2\venv\Scripts\python.exe "C:/Users/AMAL/Pychar
[[ 5]
 [12]
 [25]
 [35]
 [45]
 [55]]
coefficent of determination 0.7006018999845549
intercept 7.133243121335141
slope [0.50960758]
predicted value [ 9.68128101 13.24853406 19.87343257 24.96950834 30.06558412 35.1616599 ]
```


50
40
30
20
10
y
15
20
25
30
35
40
45
x

Date:15/01/2022

Program no:7

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

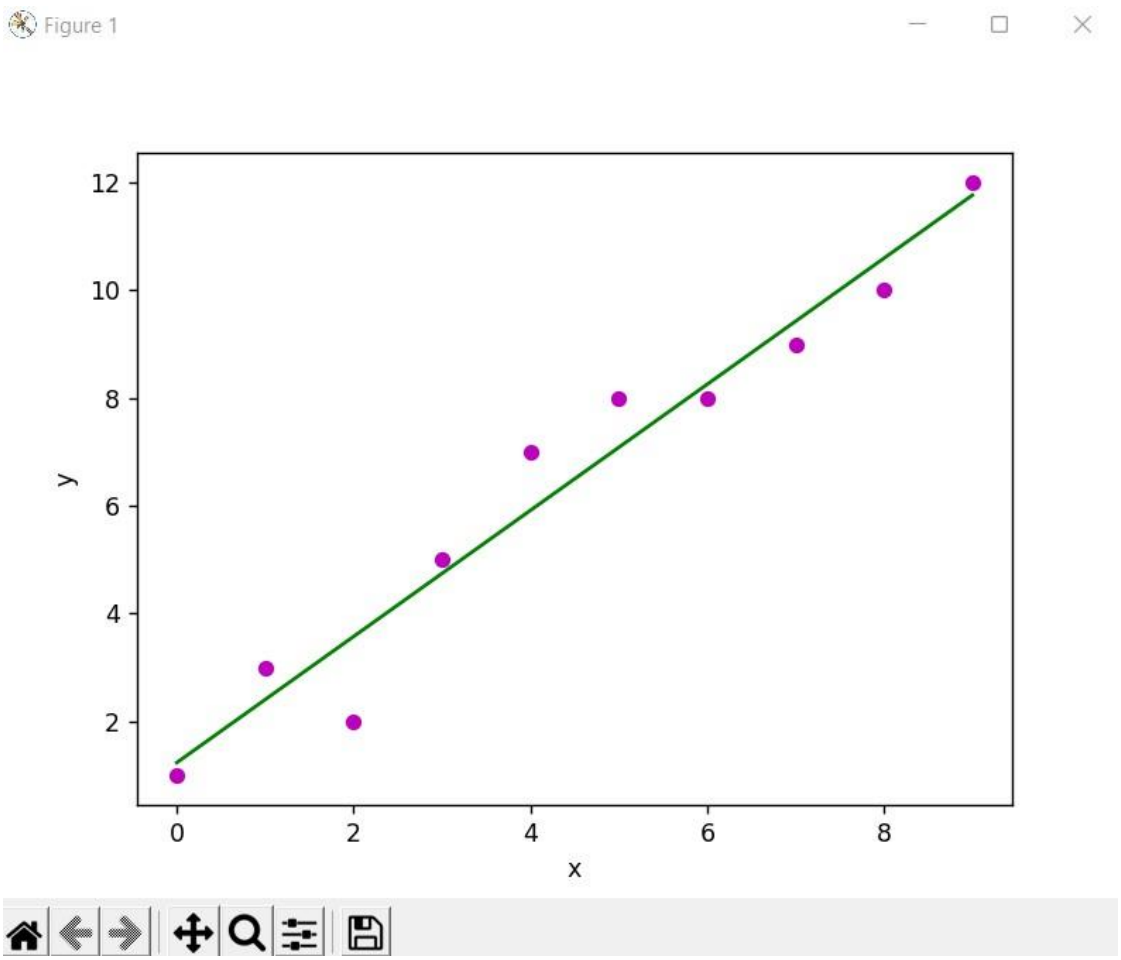
Program:

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

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

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

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

 # calculating regression
coefficients   b_1 = SS_xy / SS_xx
b_0 = m_y - b_1*m_x

 return (b_0, b_1)

def plot_regression_line(x, y, b):
```

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

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

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

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

  # function to show plot
plt.show()

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

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

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

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

Output:

```
C:\Users\AMAL\PycharmProjects\pythonProject2\venv\Scripts\python.exe
Estimated coefficients:
b_0 = 1.2363636363636363
b_1 = 1.1696969696969697
```



Date:15/01/2022

Program no: 8

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

Program:

```
import pandas
df=pandas.read_csv("cars.csv")
x=df[['Weight','Volume']] y=df['CO2']
from sklearn import linear_model
regr=linear_model.LinearRegression()
regr.fit(x,y) predictedco2=regr.predict([[2300,1300]])
print(predictedco2)
```

OUTPUT

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

Date:15/01/2022

Program - 9

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

Program:

```
import matplotlib.pyplot as plt from sklearn import

datasets,linear_model,metrics boston=datasets.load_boston() x=boston.data

y=boston.target from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split( x,y,test_size=0.4,random_state=1)

reg=linear_model.LinearRegression() reg.fit(x_train,y_train)

pre=reg.predict(x_test) print("Prediction : ",pre) print('Coefficients: ',reg.coef_)

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

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
```

Date:15/01/2022

Program no:10

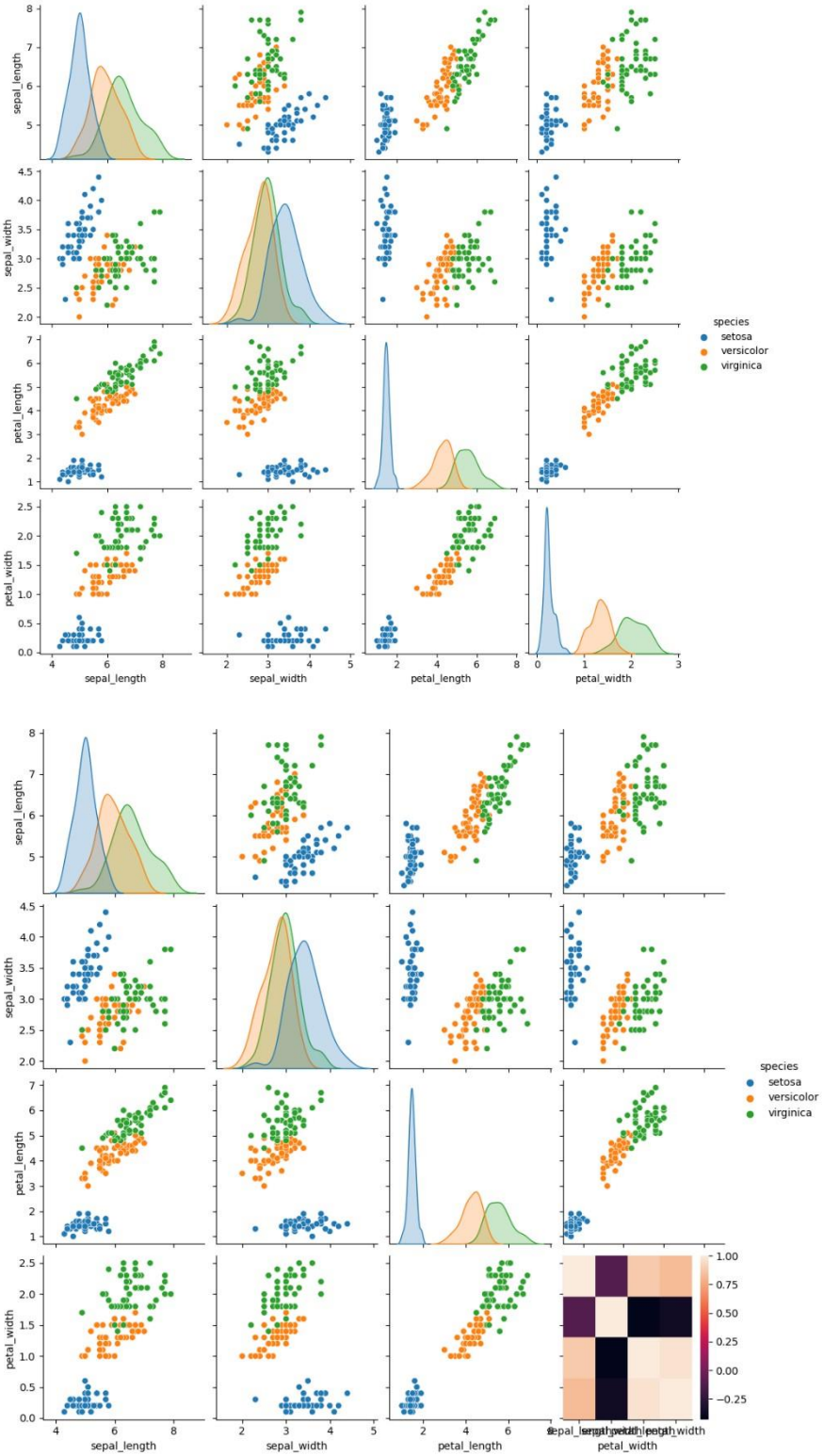
Aim: Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.

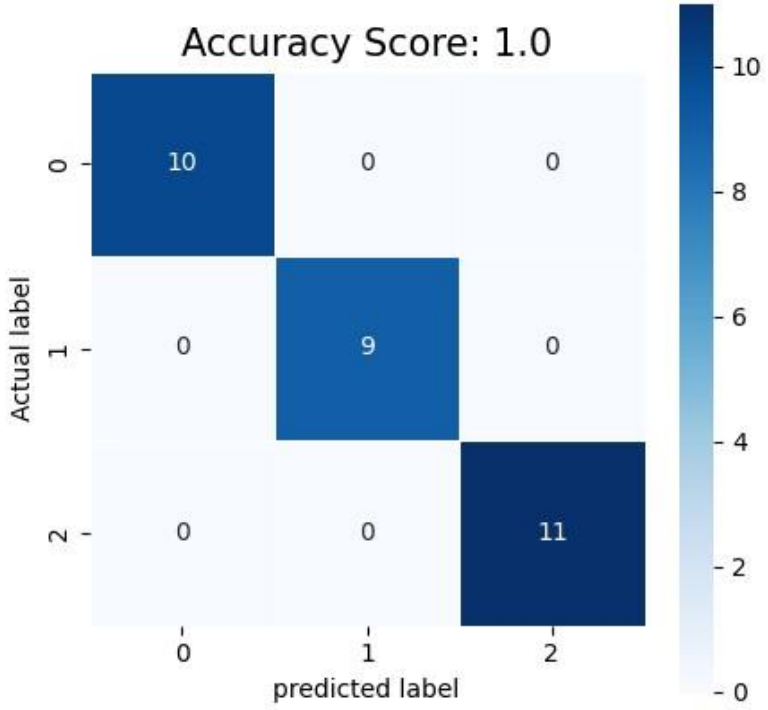
Program:

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

from sklearn.preprocessing import LabelEncoder from
sklearn.model_selection import train_test_split from sklearn.tree
import DecisionTreeClassifier from sklearn.metrics import
classification_report, confusion_matrix from sklearn.tree import
plot_tree

df = sns.load_dataset('iris')
print(df.head())
print(df.info())
df.isnull().any() #return value true if any fields are null otherwise false.boolean value
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("Training
split input- ",X_test.shape)


dtree=DecisionTreeClassifier() dtree.fit(X_train,Y_train)

print('Decision Tree Classifier Created') y_pred
=dtree.predict(X_test) print("classification report - \n",
classification_report(Y_test,y_pred)) cm = confusion_matrix(Y_test,
y_pred) plt.figure(figsize=(5,5))

sns.heatmap(data=cm,linewidths=.5, annot = True,square = True, cmap = 'Blues')
plt.ylabel('Actual label') plt.xlabel('predicted label')

all_sample_title = 'Accuracy Score: {0}'.format(dtree.score(X_test, Y_test))
plt.title(all_sample_title, size = 15)

plt.savefig("two.png")

plt.figure(figsize = (20,20))
dec_tree = plot_tree(decision_tree=dtree, feature_names = df1.columns,class_names =
["setosa", "verginica"],filled = True, precision =4 ,rounded =True) plt.savefig("three.png")
```

Output:

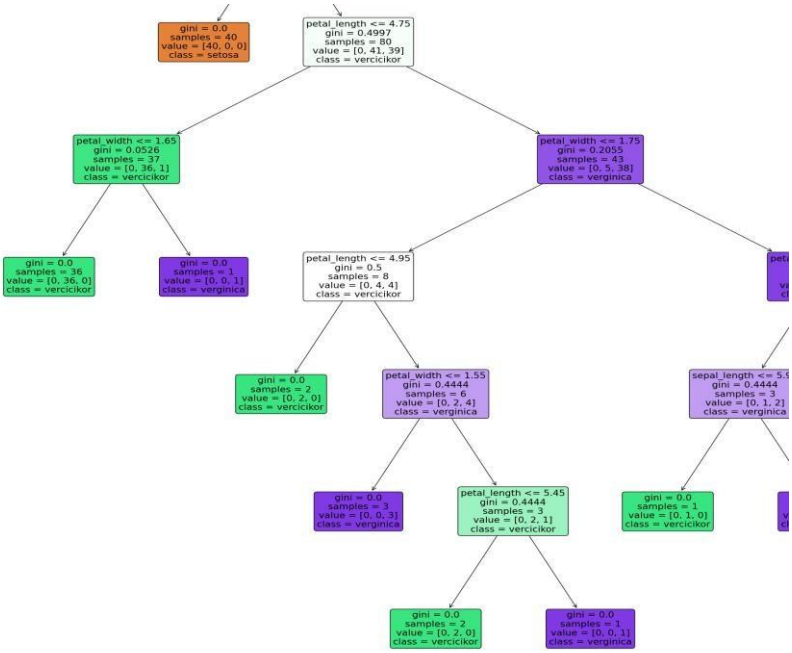

sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica




```
C:\Users\AMAL\PycharmProjects\pythonProject2\venv\Scripts\python.exe
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
(150, 5)
(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 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]
Training split input-  (120, 4)
Training 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
```


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

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:

```
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)
#finding optimal number of clusters using elbow method

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('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) #visualising the cluster
mtp.scatter(x[y_predict == 0,0], x[y_predict == 0,1], s =100, c= 'blue', label =
'cluster1')
mtp.scatter(x[y_predict == 1,0], x[y_predict == 1,1], s =100, c= 'green', label =
'cluster2') mtp.scatter(x[y_predict == 2,0], x[y_predict == 2,1], s =100, c= 'red', label =
'cluster3') mtp.scatter(x[y_predict == 3,0], x[y_predict == 3,1], s =100, c= 'cyan', label
=
```

'cluster4') mtp.scatter(x[y_predict == 4,0], x[y_predict == 4,1], s =100, c=
'yellow', label =
'cluster5') mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s=100, c='black', label='Centroids') mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)') mtp.ylabel('Spending Score (1-100)')
mtp.legend() mtp.show()

Output

```
C:\Users\mca\PycharmProjects\svd\venv\Scripts\python.exe
[[ 15  39]
 [ 15  81]
 [ 16   6]
 [ 16  77]
 [ 17  40]
 [ 17  76]
 [ 18   6]
 [ 18  94]
 [ 19   3]
 [ 19  72]
 [ 19  14]
 [ 19  99]
 [ 20  15]
 [ 20  77]
 [ 20  13]
 [ 20  79]
 [ 21  35]
```

```
[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 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 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]
```

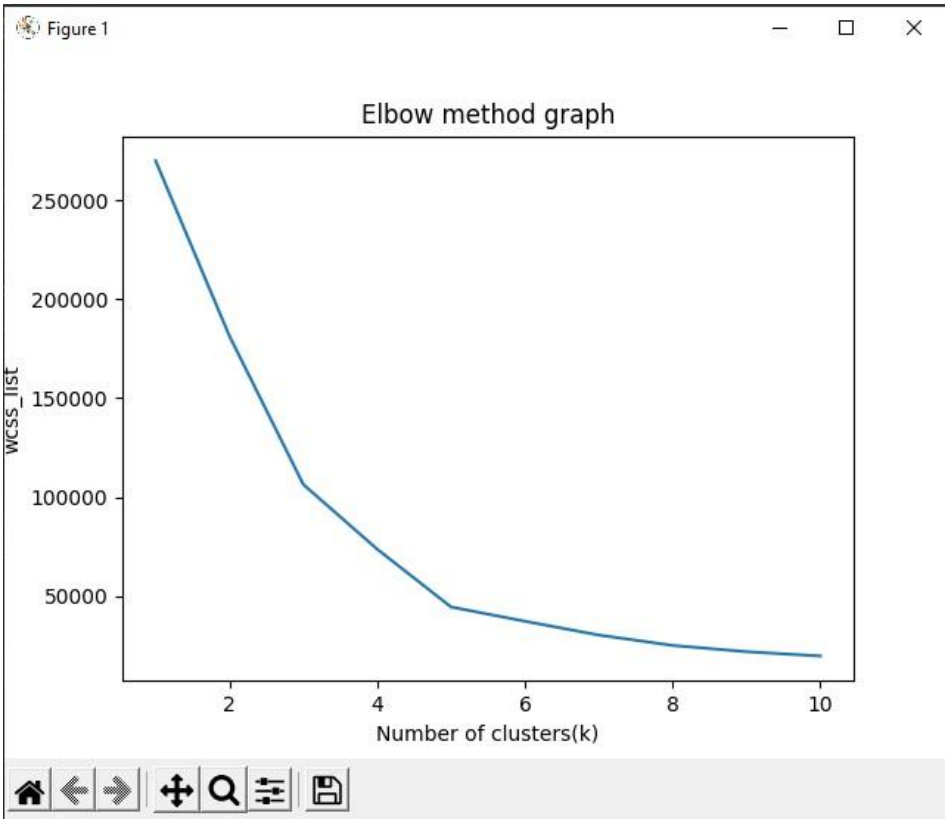


Figure 1
Elbow method graph
250000
200000
150000
100000
50000
wcss_list
2
4
6
8
10
Number of clusters(k)

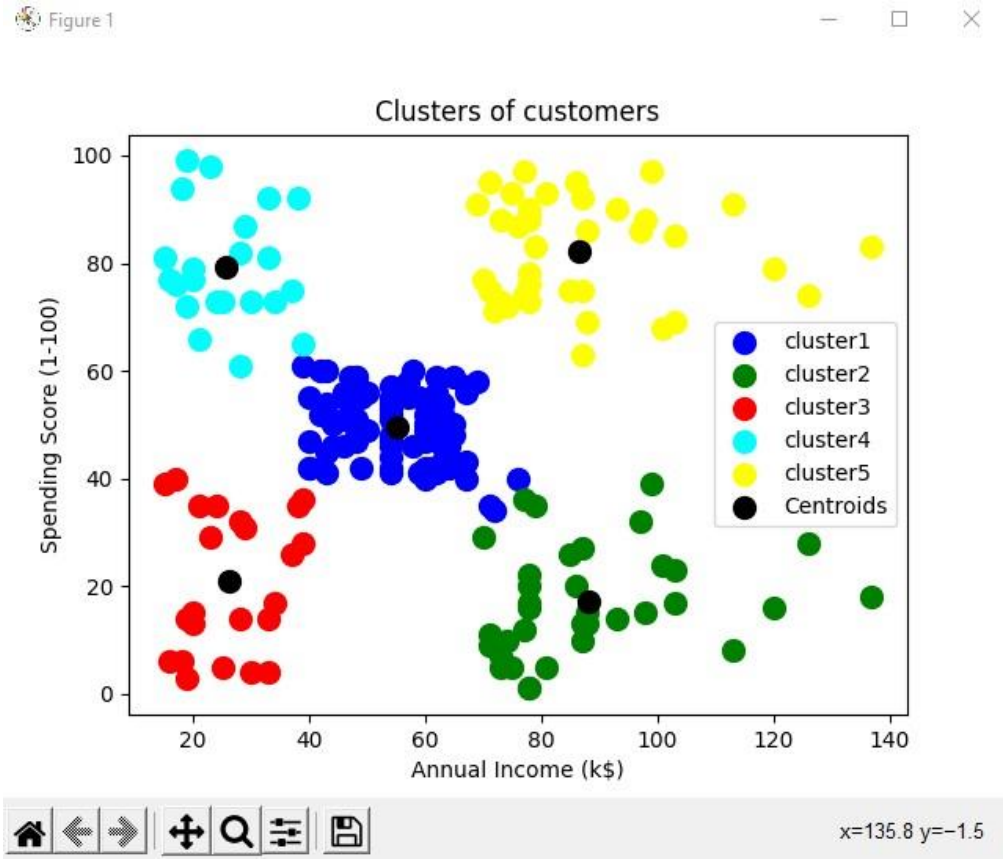


Figure 1
Clusters of customers
100
80
60
40
20
0
Spending Score (1-100)
20
40
60
80
100
120
140
Annual Income (k$)
cluster1
cluster2
cluster3
cluster4
cluster5
Centroids
x=135.8 y=−1.5

Date:05/01/2022

Program no:12

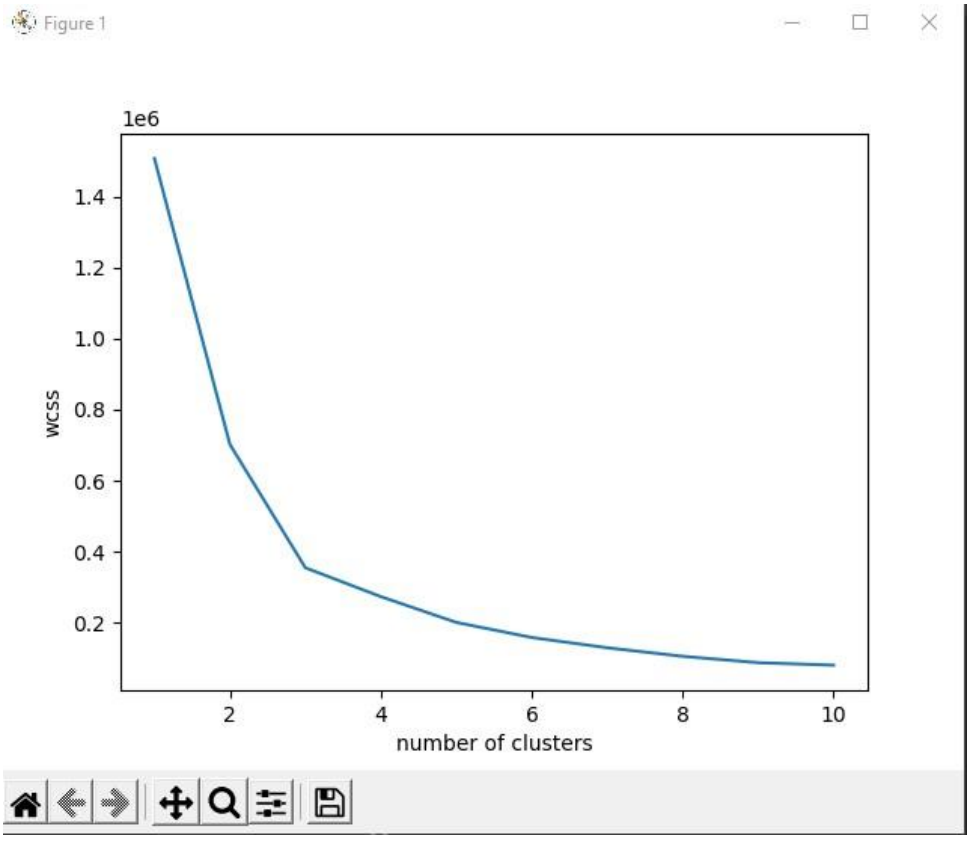
Aim: Program to implement K-Means clustering technique using any standard dataset available in the public domain.

Program:

```
import numpy as nm import
matplotlib.pyplot as mtp import
pandas as pd import sklearn

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 = []  # empty array for
i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
kmeans.fit(x)     wcss.append(kmeans.inertia_) mtp.plot(range(1, 11),
wcss) mtp.xlabel('number of clusters') mtp.ylabel('wcss') mtp.show()

kmeans = KMeans(n_clusters=5, init="k-means++", random_state=42) y_kmeans
= kmeans.fit_predict(x)

mtp.scatter(x[y_kmeans==0,0], x[y_kmeans==0,1], s=80, c='red', label = 'Cluster 1')
mtp.scatter(x[y_kmeans==1,0], x[y_kmeans==1,1], s=80, c='blue', label = 'Cluster 2')
mtp.scatter(x[y_kmeans==2,0], x[y_kmeans==2,1], s=80, c='green', label = 'Cluster 3')
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c =
'black', label = 'Centroids')
mtp.legend() mtp.show()
```

output

```
C:\Users\mca\PycharmProjects\svd\venv\Scripts\python.exe
[[ 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]
```



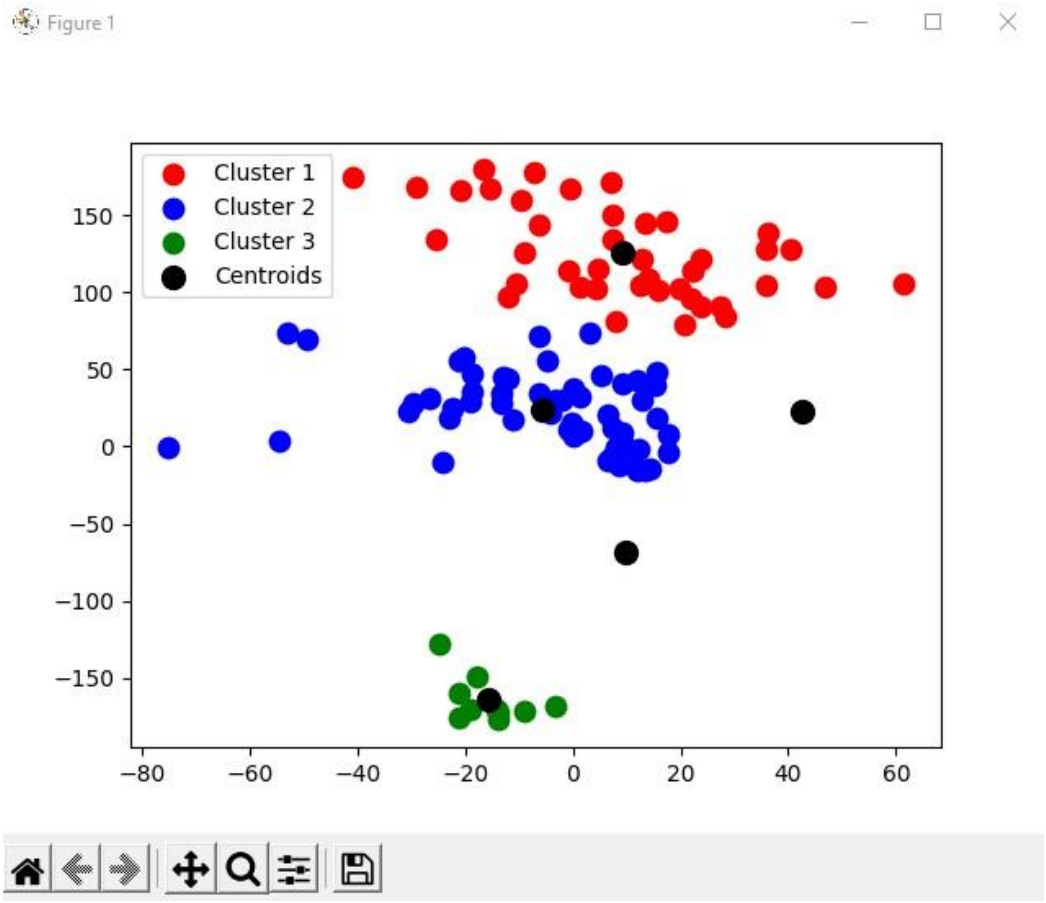


Figure 1
Cluster 1
Cluster 2
Cluster 3
Centroids
150
100
50
0
−50
−100
−150
−80
−60
−40
−20
0
20
40
60

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** 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()

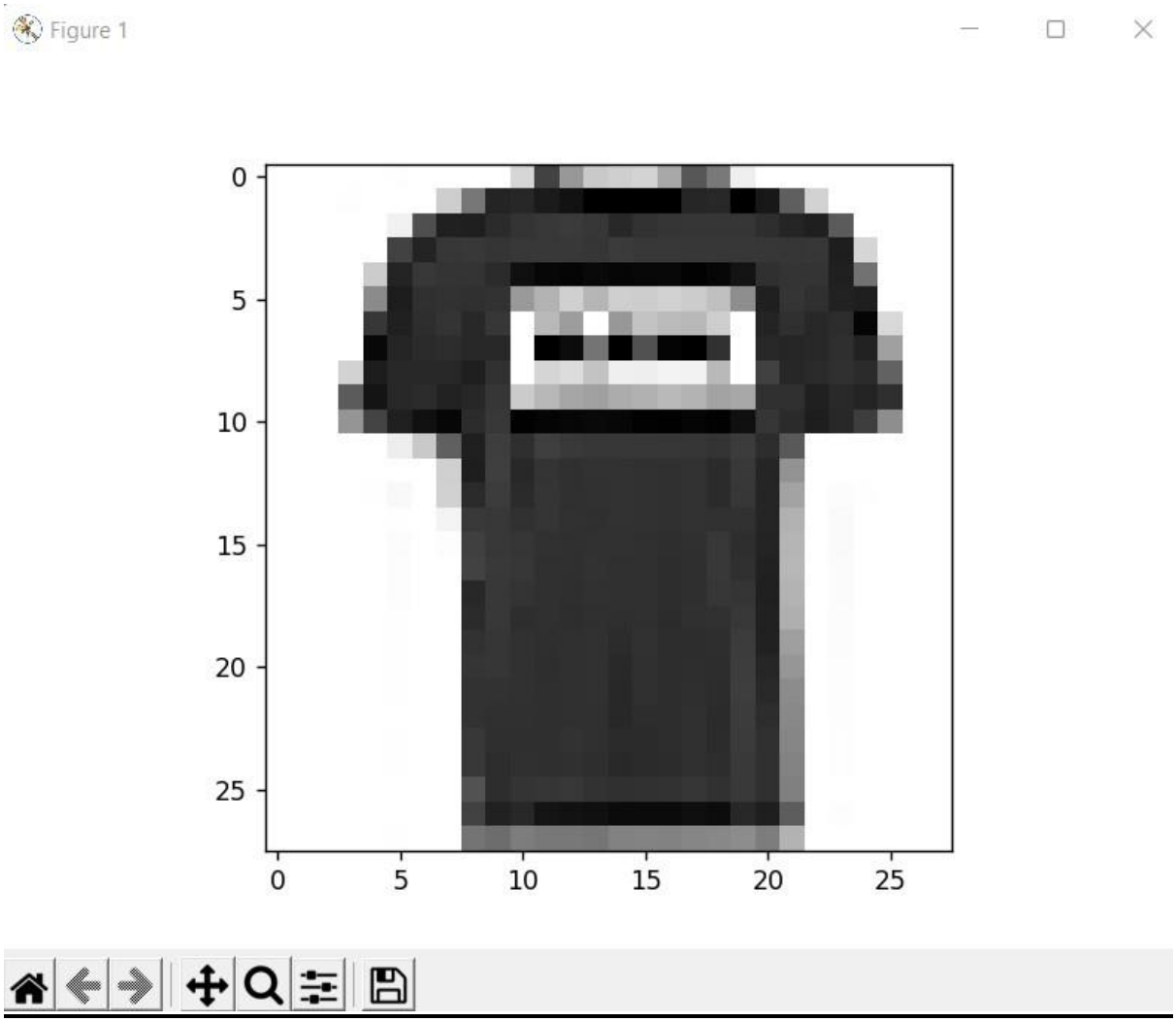
model_CNN = keras.models.Sequential()

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

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

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

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

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

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

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

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

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

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

model_CNN.compile(loss='sparse_categorical_crossentropy', optimizer='adam',

metrics=['accuracy']) x_train = x_train[..., np.newaxis] x_test = x_test[...,

np.newaxis]

history_CNN = model_CNN.fit(x_train, y_train, epochs=2, validation_split=0.1)

pd.DataFrame(history_CNN.history).plot() plt.grid(True) plt.xlabel('epochs')

plt.ylabel('loss/accuracy') plt.title('Training and validation plot')

plt.show()

test_loss, test_accuracy = model_CNN.evaluate(x_test, y_test) print('Test

Loss:{}' , 'Test Accuracy:{}'.format(test_loss, test_accuracy))
```

Output



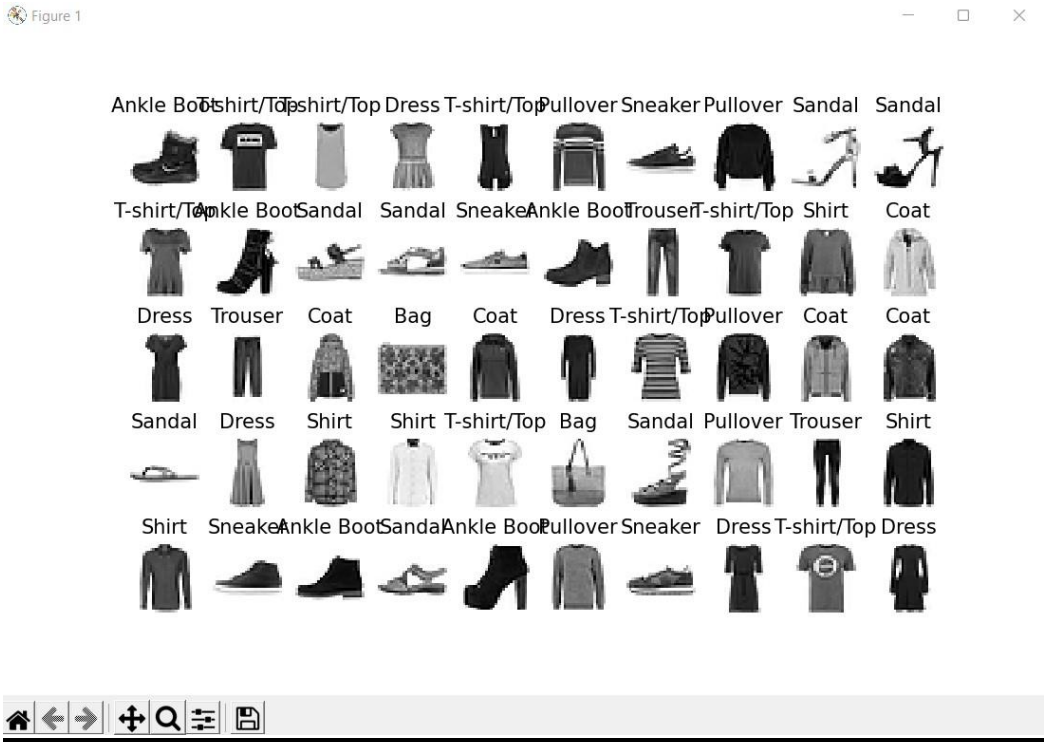


Figure 1
Ankle Boot T-shirt/Top T-shirt/Top Dress T-shirt/Top Pullover Sneaker Pullover Sandal Sandal
T-shirt/Top Ankle Boot Sandal Sandal Sneaker Ankle Boot Trouser T-shirt/Top Shirt Coat
Dress Trouser Coat Bag Coat Dress T-shirt/Top Pullover Coat Coat
Sandal Dress Shirt Shirt T-shirt/Top Bag Sandal Pullover Trouser Shirt
Shirt Sneaker Ankle Boot Sandal Ankle Boot Pullover Sneaker Dress T-shirt/Top Dress

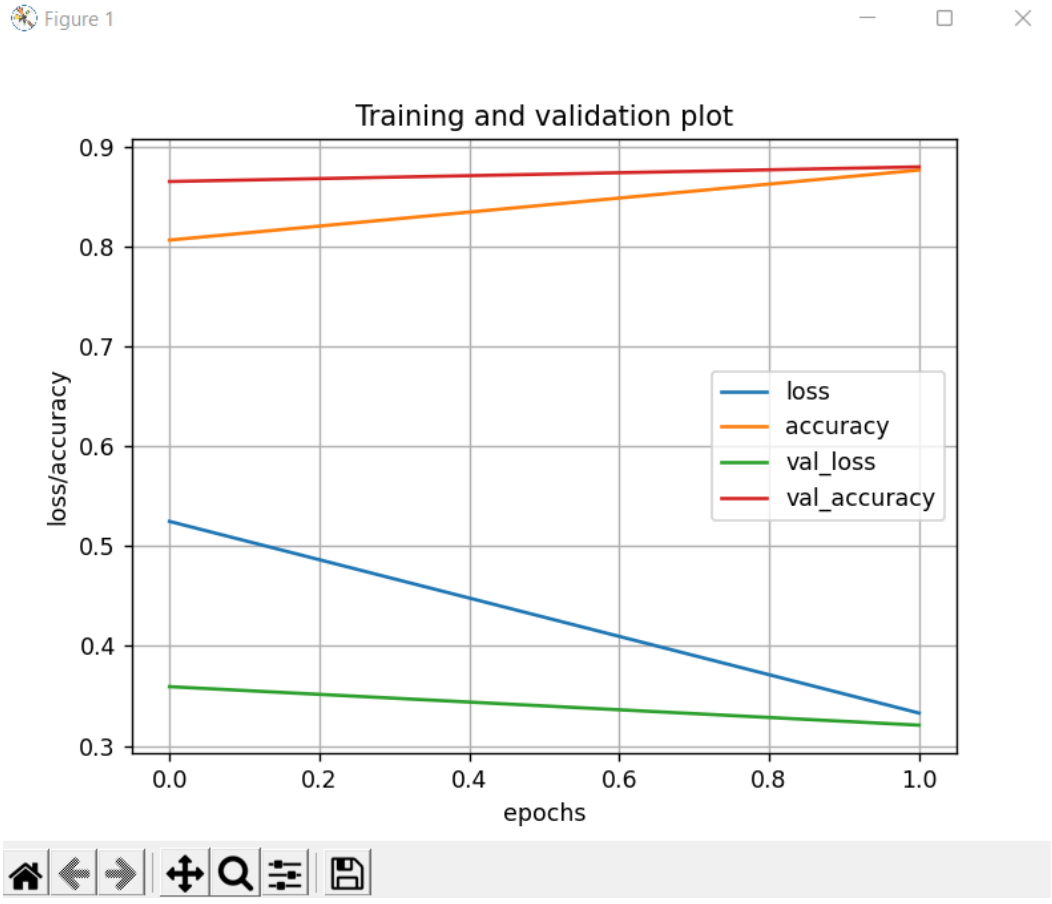


Figure 1
Training and validation plot
loss/accuracy
epochs
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0
loss
accuracy
val_loss
val_accuracy

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

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

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

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

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

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

=================================================================
Total params: 38,560
Trainable params: 38,560
Non-trainable params: 0
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        1600

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

```
=================================================================
Total params: 84,458
Trainable params: 84,458
Non-trainable params: 0
_________________________________________________________________
Epoch 1/2
1688/1688 [==============================] - 57s 33ms/step - loss: 0.5248 - accuracy: 0.8064 - val_loss: 0.3593 - val_accuracy: 0.8650
Epoch 2/2
1688/1688 [==============================] - 52s 31ms/step - loss: 0.3329 - accuracy: 0.8765 - val_loss: 0.3207 - val_accuracy: 0.8797
313/313 [==============================] - 4s 12ms/step - loss: 0.3302 - accuracy: 0.8794
Test Loss:{} Test Accuracy:0.3301725685596466

Process finished with exit code 0
```

Date:16/02/2022

Program no:14

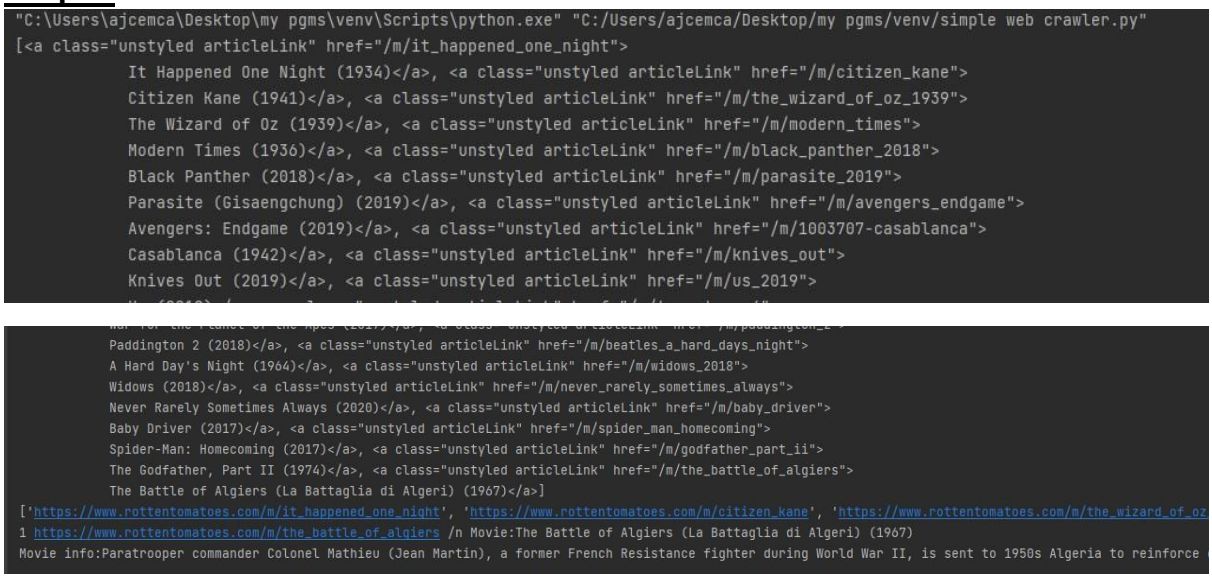
Aim: Program to implement a simple web crawler using python.

Program:

```
import requests import
lxml
from bs4 import BeautifulSoup
url = "https://www.rottentomatoes.com/top/bestofrt/" headers
= {
  'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 OPR/50.0.2762.58 (Edition Yx 01)'
}
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:



Date:16/02/2022

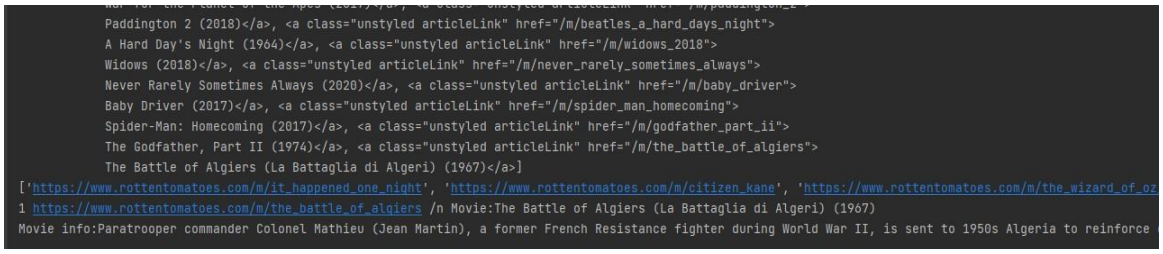
Program no:15

Aim: Program to implement a simple web crawler using python.

Program:

```
import requests import
lxml
from bs4 import BeautifulSoup
url = "https://www.rottentomatoes.com/top/bestofrt/" headers
= {
  'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 OPR/50.0.2762.58 (Edition
Yx 01)'
}
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:



Program no:16

Aim: Program to implement scrap of any website.

Program: import csv import
requests from bs4 import
BeautifulSoup

```
url="http://www.values.com/inspirational-quotes"
r=requests.get(url) print("Content:")
print(r.content) print("Prettify:")
soup=BeautifulSoup(r.content,'lxml')
print(soup.prettify())

quotes=[]
table=soup.find('div',attrs={'id':'all_quotes'})
for row in table.find_all('div',attrs={'class':'col-6 col-lg-3 text-center margin-
30pxbottom 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='insipration_quotation.csv'
with open(filename,'w',newline='')as f:

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

Output:

```
"C:\Users\ajcemca\Desktop\my pgms\venv\Scripts\python.exe" "C:/Users/ajcemca/Desktop/my pgms/transpose.py"
Content:
b'<!DOCTYPE html>\n<html class="no-js" dir="ltr" lang="en-US">\n    <head>\n        <title>Inspirational Quotes - Motivational Quotes - Leadership Quotes | Pas
Prettify:
<!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"/>
```

```
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/quo
```

Date:16/02/2022

Program no:17

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

Program:

```
def generate_ngrams(text,WordsToCombine):
  words=text.split()  output=[]  for i in
range(len(words)-WordsToCombine+1):
output.append(words[i:i + WordsToCombine])
return output

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

Output:

```
"C:\Users\ajcemca\Desktop\my pgms\venv\Scripts\python.exe" "C:/Users/ajcemca/Desktop/my pgms/venv/ngrams.py"
[['this', 'is', 'a'], ['is', 'a', 'good'], ['a', 'good', 'book'], ['good', 'book', 'to'], ['book', 'to', 'study']]

Process finished with exit code 0
```

Date:16/02/2022

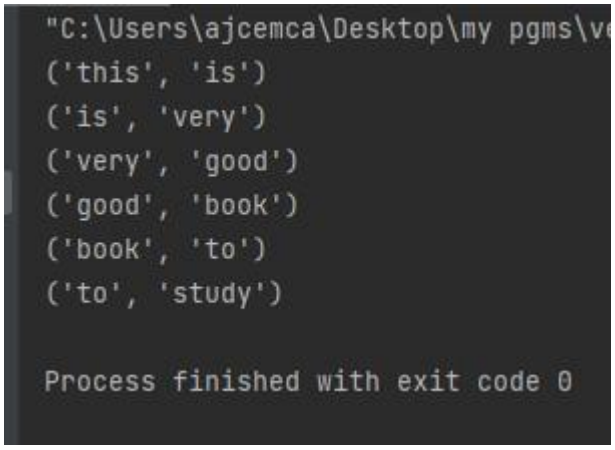
Program no:18

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

Program:

```
import nltk

from nltk.util import ngrams


samplText="this is very good book to study"

Ngrams=ngrams(sequence=nltk.wordpunct_tokenize(samplText),n=2) for

grams in Ngrams:

  print(grams)
```

Output:



Date:16/02/2022

Program no:19

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

Program:

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords') nltk.download('punkt')
nltk.download('averaged_perceptron_tagger') 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:

```
[('Sukanya', 'NNP'), (',', ','), ('Rajib', 'NNP'), ('Naba', 'NNP'), ('good', 'JJ'), ('friends', 'NNS'), ('.', '.')]
[('Sukanya', 'NNP'), ('getting', 'VBG'), ('married', 'VBN'), ('next', 'JJ'), ('year', 'NN'), ('.', '.')]
[('Marriage', 'NN'), ('big', 'JJ'), ('step', 'NN'), ('one', 'CD'), ("'", 'NN'), ('life.It', 'NN'), ('exciting', 'VBG'), ('frightening', 'NN'), ('.', '.')]
[('But', 'CC'), ('friendship', 'NN'), ('sacred', 'VBD'), ('bond', 'NN'), ('people.It', 'NN'), ('special', 'JJ'), ('kind', 'NN'), ('love', 'VB'), ('us', 'PRP'), ('.', '.')]
[('Many', 'JJ'), ('must', 'MD'), ('tried', 'VB'), ('searching', 'VBG'), ('friend', 'NN'), ('never', 'RB'), ('found', 'VBD'), ('right', 'JJ'), ('one', 'CD'), ('.', '.')]

Process finished with exit code 0
```

Date:16/02/2022

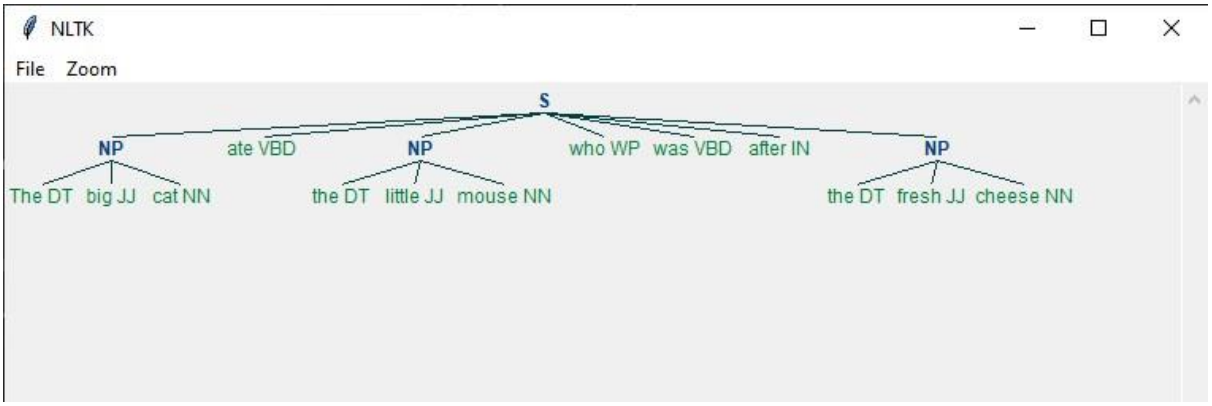
Program no:20

Aim: Program to perform chunking.

Program:

```
import nltk nltk.download('punkt') 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:


NLTK
File Zoom
S
NP
ate VBD
NP
who WP
was VBD
after IN
NP
The DT
big JJ
cat NN
the DT
little JJ
mouse NN
the DT
fresh JJ
cheese NN

Program no:19

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

Program:

```
import nltk

sample_text="""Rama killed Ravana to save sita from Lanka.The legend of Ramayan is thr most

popular Indian epic.A lot of movies and serials have already been shot in several languages here

in Indaia based on ramayana."""

tokenized=nltk.sent_tokenize(sample_text)

for i in tokenized:

 words=nltk.word_tokenize(i)

 tagged_words=nltk.pos_tag(words)

 chunkgram=r"""VB:{<DT>*<NN>?<JJ>}"""

 chunkParser=nltk.RegexpParser(chunkgram)

 chunked=chunkParser.parse(tagged_words)

 print(chunked)  chunked.draw()
```

Output:

```
C:\Users\ajcemca\PycharmProjects\mypython\ve
(S
  Rama/NNP
  killed/VBD
  Ravana/NNP
  to/TO
  save/VB
  sita/NN
  from/IN
  Lanka.The/NNP
  legend/NN
  of/IN
  Ramayan/NNP
  is/VBZ
  (VB thr/JJ)
  most/RBS
  (VB popular/JJ)
  (VB Indian/JJ)
  epic.A/NN
  lot/NN
  of/IN
  movies/NNS
  and/CC
  serials/NNS
  have/VBP
  already/RB
```