20MCA241 DATA SCIENCE LAB

Lab Report Submitted By

FREDDY JENSON

Reg. No.: AJC20MCA-2038

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

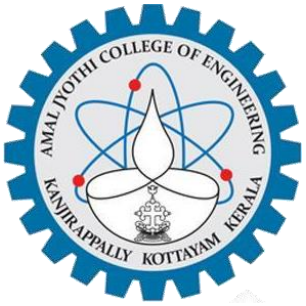
[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

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 **FREDDY JENSON (Reg.No:AJC20MCA-2038)** 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. Nimmy Francis

Lab In-Charge

CONTENT

Date:24/11/2021

PROGRAM NO : 1

AIM :

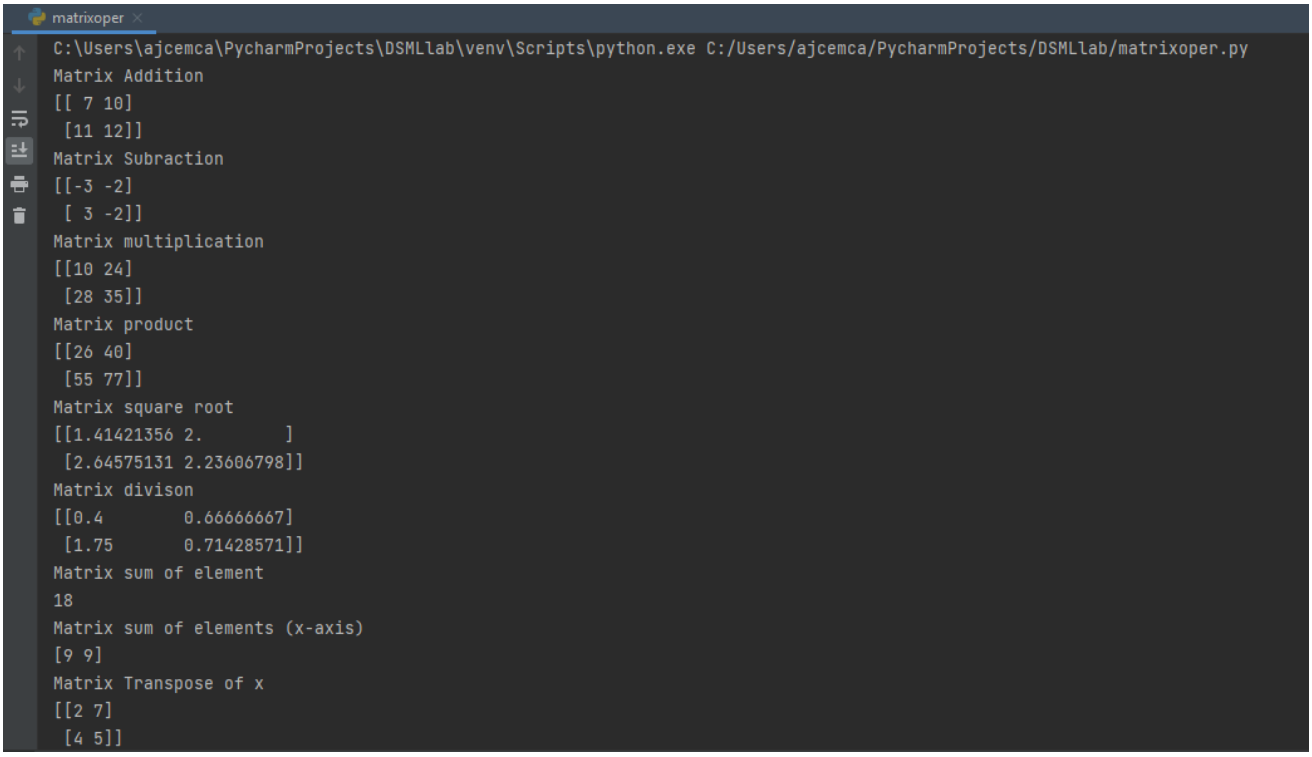
Perform all matrix operations using python (using numpy).

PROGRAM CODE :

```
import numpy

x=numpy.array([[2,4],[7,5]])

y=numpy.array([[5,6],[4,7]])

print("Matrix Addition")

print(numpy.add(x,y))

print("Matrix Subraction")

print(numpy.subtract(x,y))

print("Matrix multiplication")

print(numpy.multiply(x,y))

print("Matrix product")

print(numpy.dot(x,y))

print("Matrix square root")

print(numpy.sqrt(x))

print("Matrix divison")

print(numpy.divide(x,y))

print("Matrix sum of element")

print(numpy.sum(x))

print("Matrix sum of elements (x-axis)")
```

```
print(numpy.sum(x,axis=0))

print("Matrix Transpose of x")

print(x.T)
```

OUTPUT



Date:01/12/2021

PROGRAM NO : 2

AIM :

Perform SVD(Singular Value Decomposition)

PROGRAM CODE:

```
from  numpy import array

from scipy.linalg import svd

a=array([[1,2,3,4],[7,8,3,5],[4,6,9,10]])

print(a)

u,s,vt=svd(a)

print("Decomposed Matrix\n",u)

print("Inverse Matrix\n",s)

print("Transpose matrix\n",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
```

Date:01/12/2021

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 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

idata=load_iris()

x=idata.data

y=idata.target

x_train,x_test,y_train,y_test=train_test_split( x,y,test_size=0.3,random_state=55)

knn=KNeighborsClassifier(n_neighbors=3)

knn.fit(x_train,y_train)

y_p=knn.predict(x_test)

print(knn.predict(x_test))

print("Accuracy score : ",accuracy_score(y_test,y_p))
```

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
```

DATE:01/12/2021

PROGRAM NO : 4

AIM :

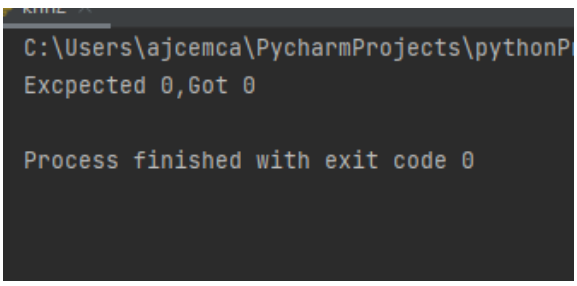
Program to implement k-NN classification using any random data set without using inbuilt packages.

PROGRAM CODE :

```
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



Date:08/12/2021

PROGRAM NO : 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 CODE :

```
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

Date:08/12/2021

PROGRAM NO : 6

AIM :

Program to implement linear regression techniques using any standard dataset available in the public domain and evaluate its performance (with inbuilt).

PROGRAM CODE :

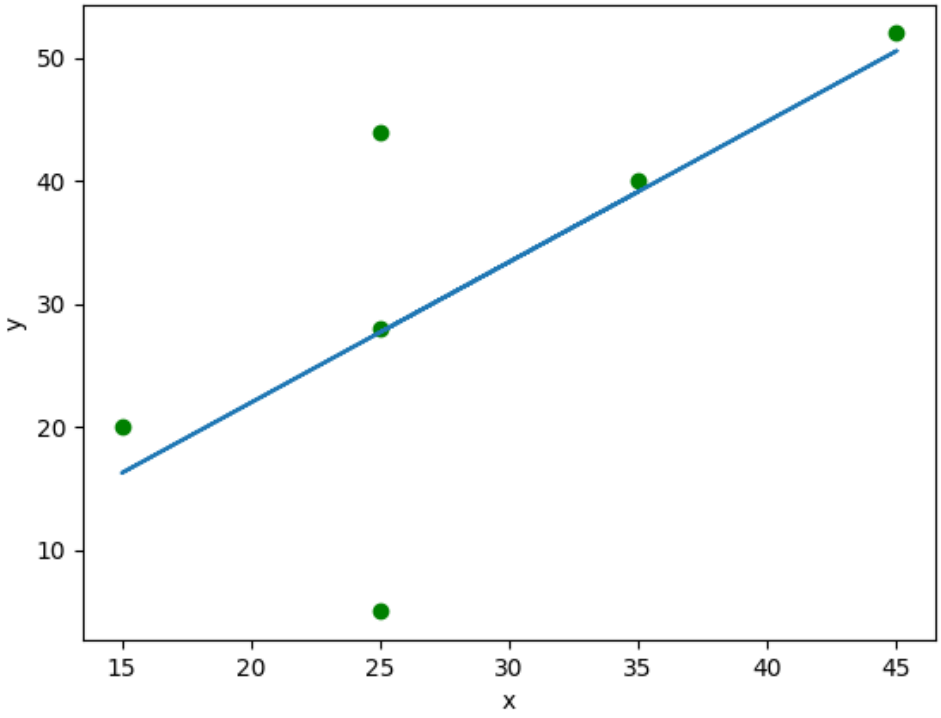
```
import  numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

x=np.array([5,15,25,35,45,55]).reshape((-1,1))

y=np.array([5,20,14,32,22,38])

print(x)

print(y)

model=LinearRegression()

model.fit(x,y)

r_sq=model.score(x,y)

print('coefficent of determination: ',r_sq)

print('intercept: ',model.intercept_)

print('slope : ',model.coef_)

y_pred=model.predict(x)

print('Predicted response: ',y_pred)

plt.scatter(x,y,color="g")

plt.plot(x,y_pred)

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 : 7

AIM :

Program to implement linear regression techniques using any standard dataset available in the public domain and evaluate its performance (without inbuilt).

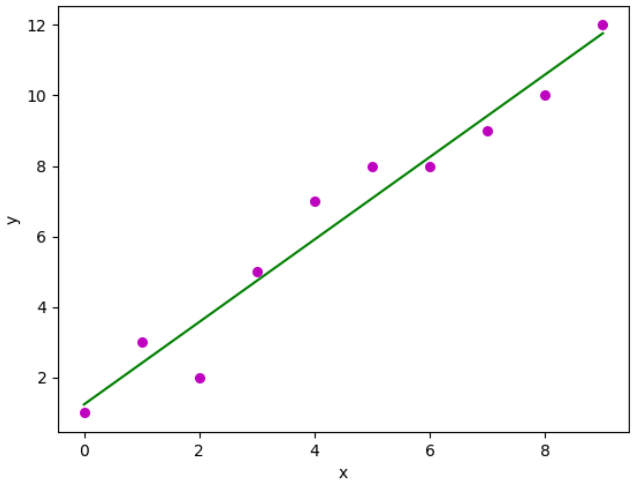
PROGRAM CODE :

```
import numpy as np

import matplotlib.pyplot as plt

def estimate_coef(x,y):

  n=np.size(x)

  m_x=np.mean(x)

  m_y=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_regr_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="g")

  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:\nb_0 = {}  \
          \nb_1 = {}".format(b[0], b[1]))
    plot_regr_line(x, y, b)

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

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\py
Estimated coefficients:
b_0 = 1.2363636363636363
b_1 = 1.1696969696969697
```



Date:15-12-2021

PROGRAM NO : 8

AIM :

Program to implement multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

PROGRAM CODE :

```
import pandas

df=pandas.read_csv("cars.csv")

x=df[['Weight','Volume']]

y=df['CO2']

from sklearn import linear_model

regr=linear_model.LinearRegression()

regr.fit(x,y)

predictedco2=regr.predict([[2300,1300]])

print(predictedco2)
```

OUTPUT

```
warnings.warn(
[107.2087328]
[0.00755095 0.00780526]
```

Date:15-12-2021

PROGRAM NO : 9

AIM :

Program to implement multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

PROGRAM CODE :

```
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:22-12-2021

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 CODE :

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


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)
```

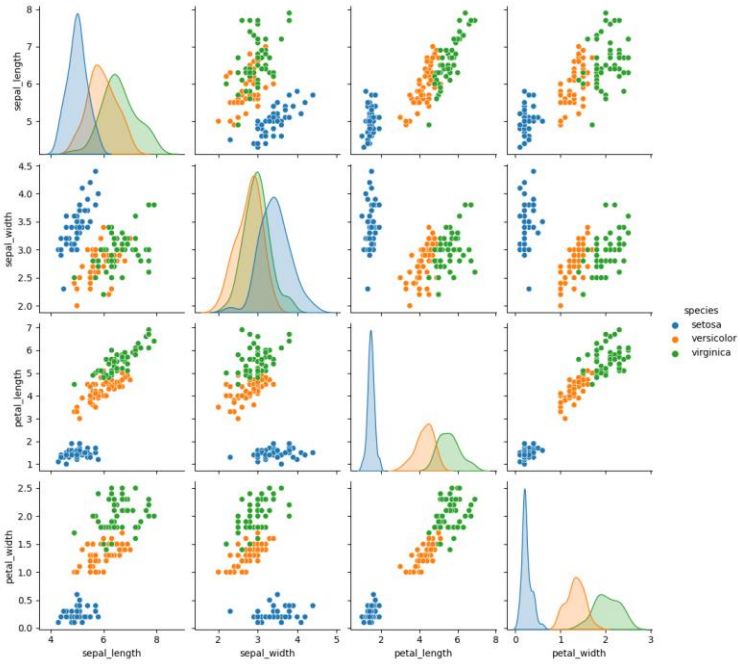
```
dtree=DecisionTreeClassifier()
dtree.fit(x_train,y_train)

print("Decision tree classifier created")

y_pred=dtree.predict(x_test)
print("classsification 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('Predictd label')

all_sample_title='Accuracy Score:{0}'.format(dtree.score(x_test,y_test))

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

OUTPUT

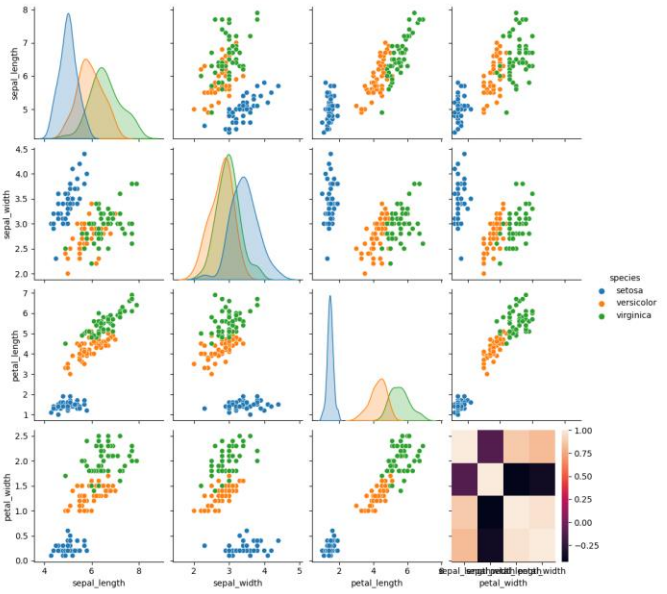
```
C:\Users\ashis\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/ashis/PycharmProjects/pythonProject1/venv/d
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          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
```

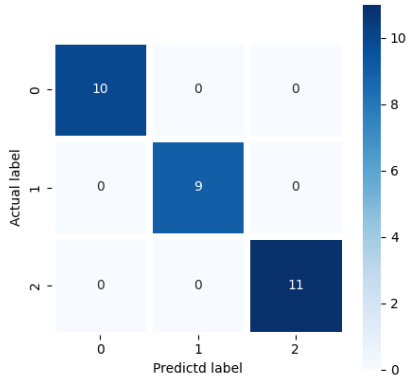
Pne.png



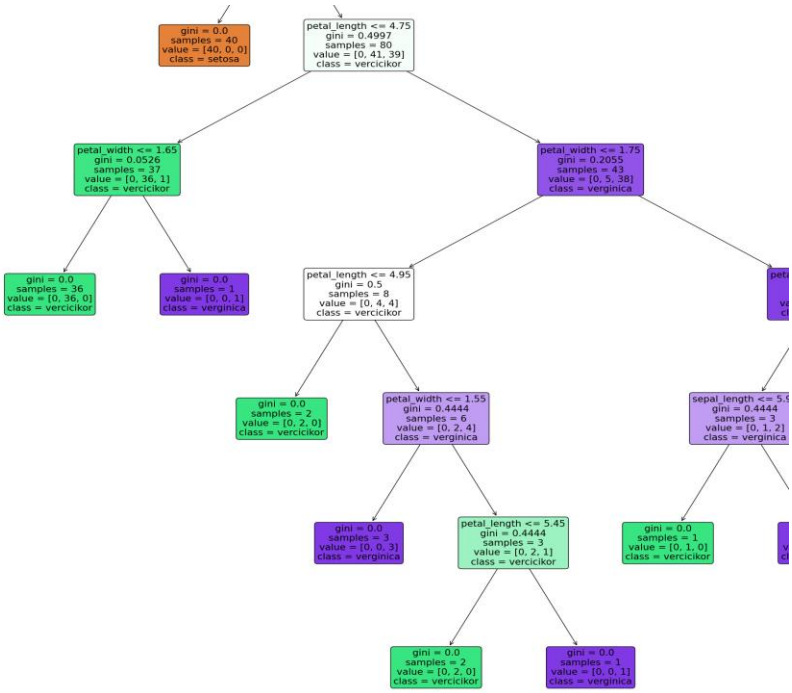
One.png



Two.png



Three.png



Date:05-01-2022

PROGRAM NO : 11

AIM :

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

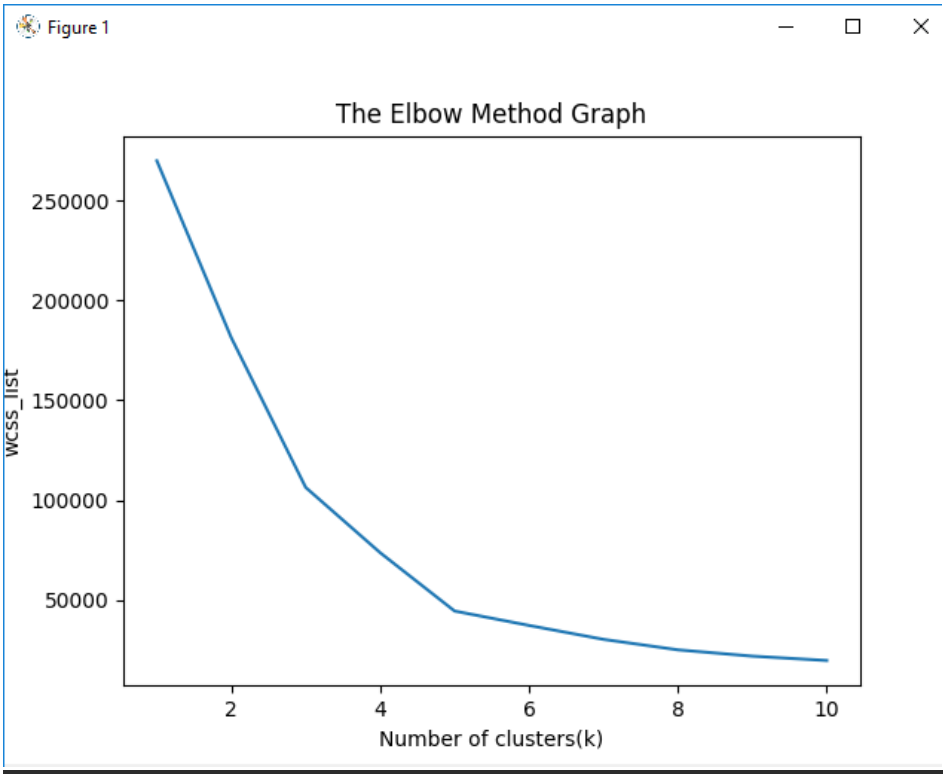
PROGRAM CODE :

```
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=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='cluster')

mtp.title('Clusters of customers')

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

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

mtp.legend()

mtp.show()
```

OUTPUT

```
C:\Users\ajcemca\PycharmProje
[[ 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]
```



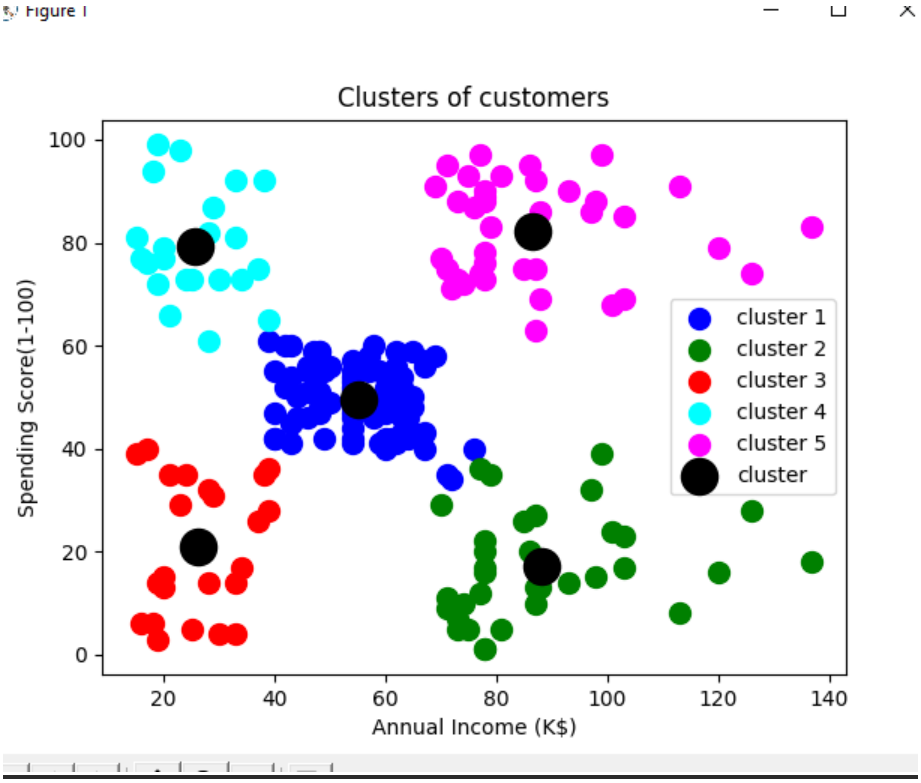


Figure 1
Clusters of customers
Spending Score(1-100)
Annual Income (K$)
0
20
40
60
80
100
20
40
60
80
100
120
140
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster

Date:05-01-2022

PROGRAM NO : 12

AIM :

Program to implement k-means clustering technique using any standard dataset available in the public domain (Using world_country_and_usa_states_latitude_and_longitude_values.csv)

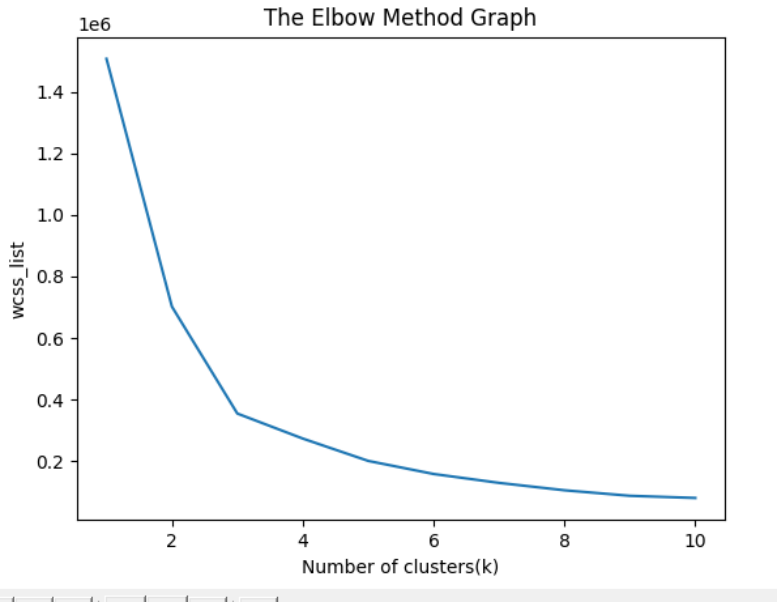
PROGRAM CODE :

```
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='clu
ster')
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]
```



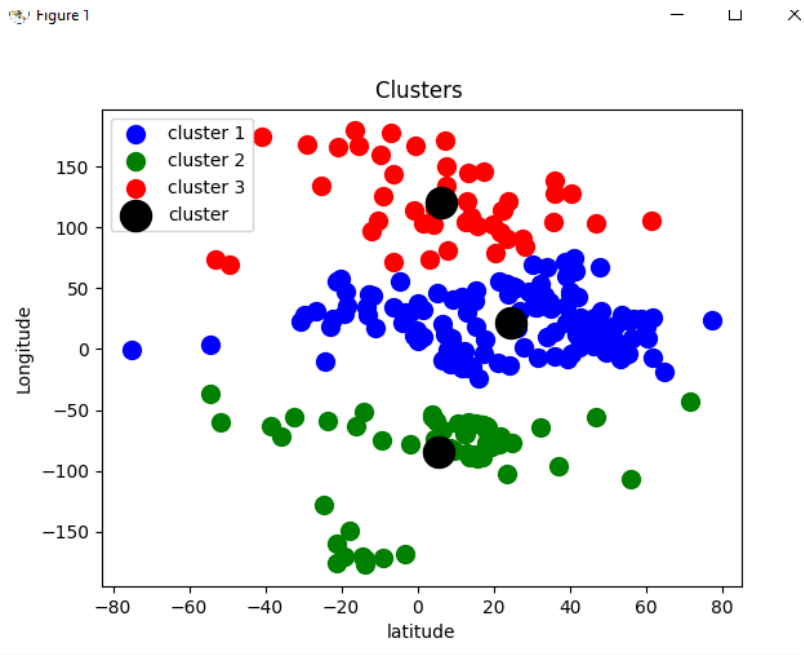


Figure 1
Clusters
cluster 1
cluster 2
cluster 3
cluster
Longitude
latitude
150
100
50
0
−50
−100
−150
−80
−60
−40
−20
0
20
40
60
80

Date:02/02/2022

PROGRAM NO : 13

AIM :

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

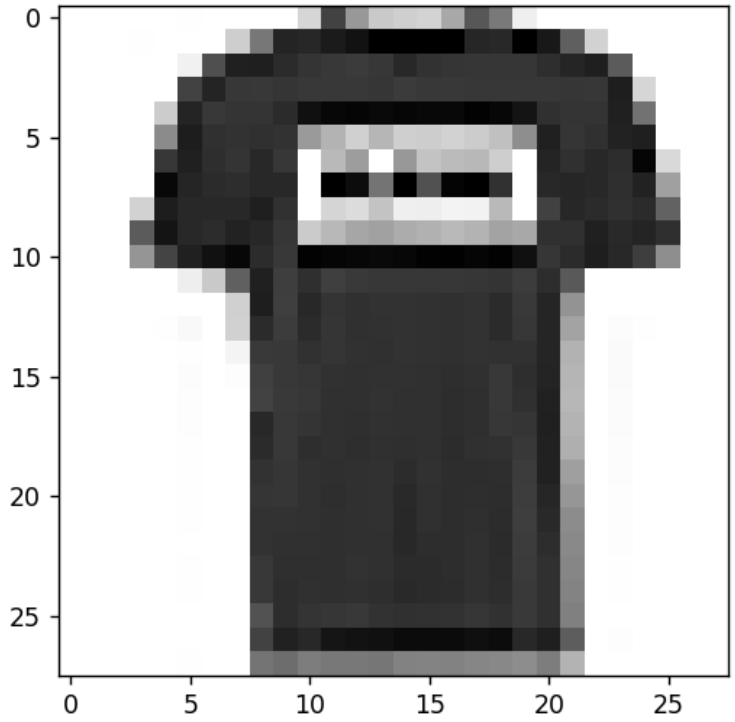
PROGRAM CODE :

```
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','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=['accurac
y'])

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

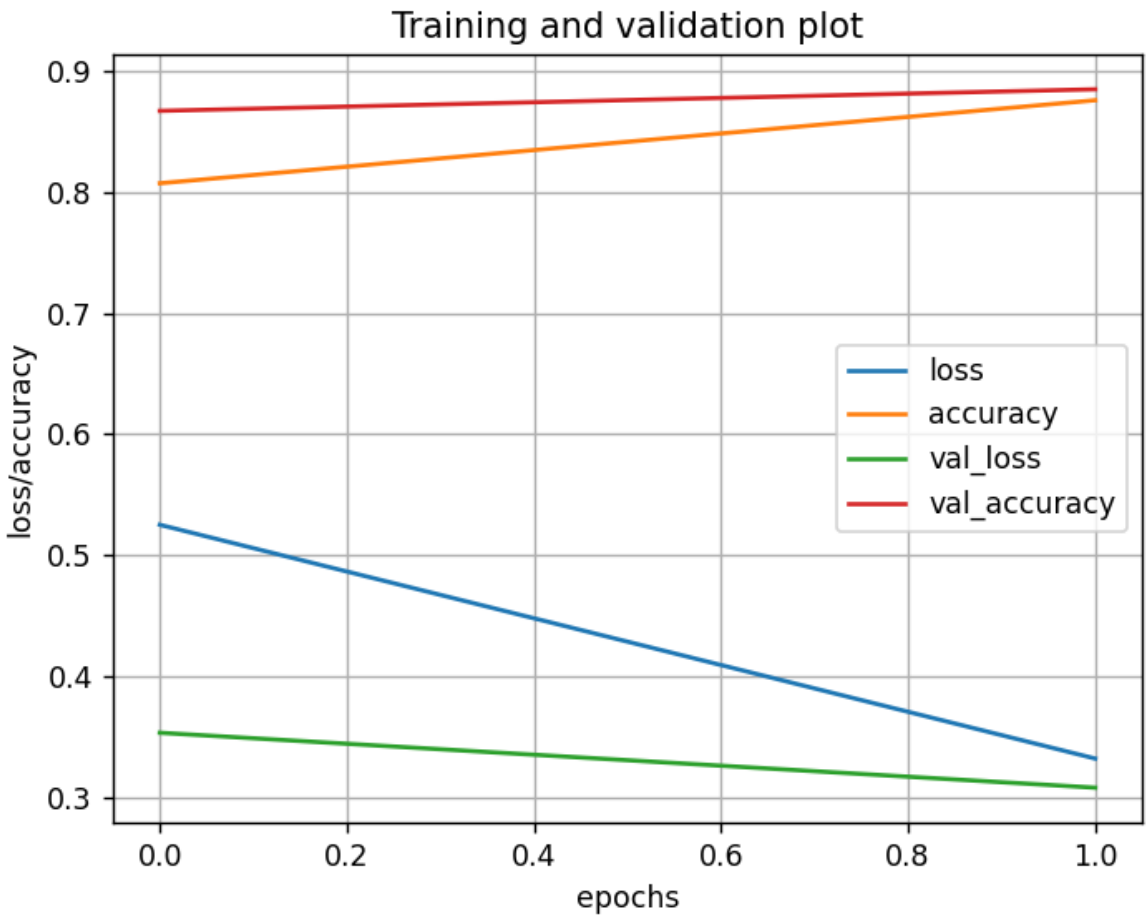

0
5
10
15
20
25
0
5
10
15
20
25


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

```
(60000, 28, 28) (10000, 28, 28)
2022-02-02 12:03:16.761271: W tensorflow/stream_executor/platform/default/dso_l
2022-02-02 12:03:16.763256: W tensorflow/core/common_runtime/gpu/gpu_device.cc:
Skipping registering GPU devices...
2022-02-02 12:03:16.773939: I tensorflow/core/platform/cpu_feature_guard.cc:151
To enable them in other operations, rebuild TensorFlow with the appropriate comp
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
```

```
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
 )

 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 [==============================] - 74s 43ms/step - loss: 0.5097 - accuracy: 0.8133 - val_loss: 0.3481 - val_accuracy: 0.8688
Epoch 2/2
1688/1688 [==============================] - 73s 43ms/step - loss: 0.3272 - accuracy: 0.8795 - val_loss: 0.3289 - val_accuracy: 0.8763
313/313 [==============================] - 4s 13ms/step - loss: 0.3441 - accuracy: 0.8721
 Test Loss :0.34412604570388794, Test Accuracy : 0.8720999956130981

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

```
[<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">
```

```
['https://www.rottentomatoes.com/m/it_happened_one_night']
1 https://www.rottentomatoes.com/m/it_happened_one_night /n MovieIt Happened One Night (1934)
Movie info:In Frank Capra's acclaimed romantic comedy, spoiled heiress Ellie Andrews (Claudette Colbert) impetuously marries the scheming King Westley, leadi
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane']
2 https://www.rottentomatoes.com/m/citizen_kane /n MovieCitizen Kane (1941)
Movie info:When a reporter is assigned to decipher newspaper magnate Charles Foster Kane's (Orson Welles) dying words, his investigation gradually reveals th
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wizard_of_o
3 https://www.rottentomatoes.com/m/the_wizard_of_oz_1939 /n MovieThe Wizard of Oz (1939)
Movie info:When a tornado rips through Kansas, Dorothy (Judy Garland) and her dog, Toto, are whisked away in their house to the magical land of Oz. They foll
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wizard_of_o
4 https://www.rottentomatoes.com/m/modern_times /n MovieModern Times (1936)
Movie info:This comedic masterpiece finds the iconic Little Tramp (Charlie Chaplin) employed at a state-of-the-art factory where the inescapable machinery co
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wizard_of_o
5 https://www.rottentomatoes.com/m/black_panther_2018 /n MovieBlack Panther (2018)
Movie info:After the death of his father, T'Challa returns home to the African nation of Wakanda to take his rightful place as king. When a powerful enemy su
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wizard_of_o
6 https://www.rottentomatoes.com/m/parasite_2019 /n MovieParasite (Gisaengchung) (2019)
Movie info:Greed and class discrimination threaten the newly formed symbiotic relationship between the wealthy Park family and the destitute Kim clan.
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wizard_of_o
7 https://www.rottentomatoes.com/m/avengers_endgame /n MovieAvengers: Endgame (2019)
Movie info:Adrift in space with no food or water, Tony Stark sends a message to Pepper Potts as his oxygen supply starts to dwindle. Meanwhile, the remaining
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wizard_of_o
```

Date:16/02/2022

PROGRAM NO : 15

AIM :

Implement a simple web crawler.

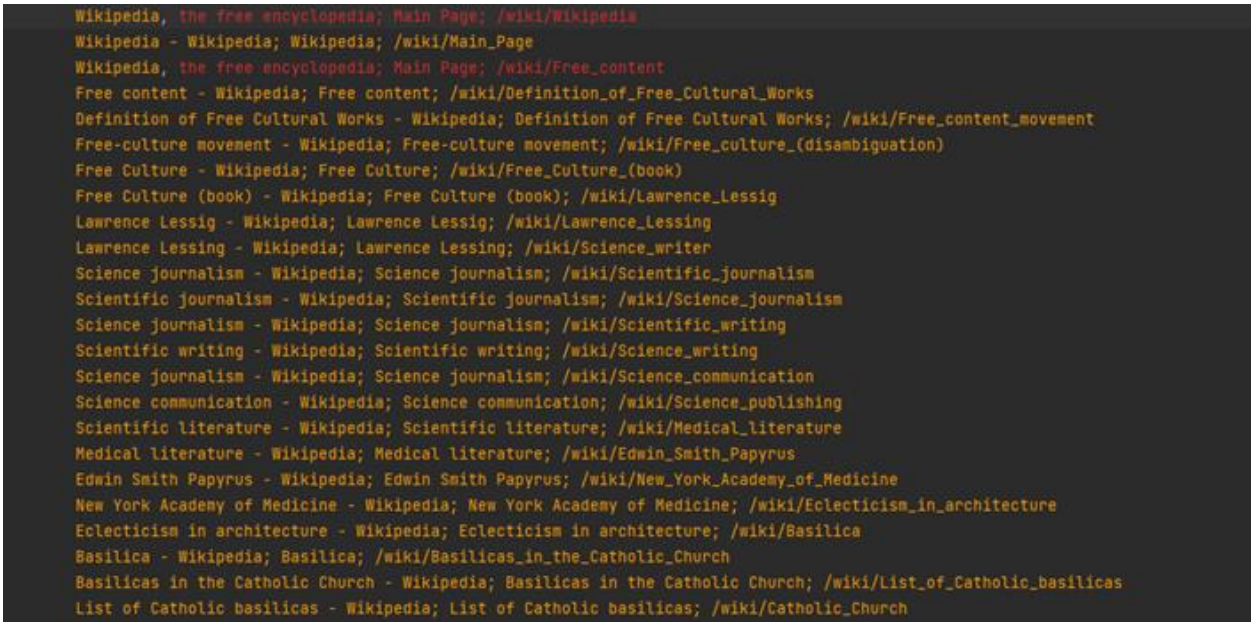
PROGRAM CODE :

```
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



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/7444/
LOVE,/inspirational-quotes/7439-at-the-touch-of-love-everyone-becomes-a-poet,https://assets.passiton.com/quotes/quote_ar
FRIENDSHIP,/inspirational-quotes/8304-a-friend-may-be-waiting-behind-a-stranger-s-face,https://assets.passiton.com/quote
FRIENDSHIP,/inspirational-quotes/3331-wherever-we-are-it-is-our-friends-that-make,https://assets.passiton.com/quotes/quo
FRIENDSHIP,/inspirational-quotes/8303-find-a-group-of-people-who-challenge-and,https://assets.passiton.com/quotes/quote_
FRIENDSHIP,/inspirational-quotes/8302-there-s-not-a-word-yet-for-old-friends-who-ve,https://assets.passiton.com/quotes/q
FRIENDSHIP,/inspirational-quotes/7435-there-are-good-ships-and-wood-ships-ships-that,https://assets.passiton.com/quotes/
PERSISTENCE,/inspirational-quotes/6377-at-211-degrees-water-is-hot-at-212-degrees,https://assets.passiton.com/quotes/quo
PERSISTENCE,/inspirational-quotes/8301-the-key-of-persistence-opens-all-doors-closed,https://assets.passiton.com/quotes/
PERSISTENCE,/inspirational-quotes/7918-you-keep-putting-one-foot-in-front-of-the,https://assets.passiton.com/quotes/quot
PERSISTENCE,/inspirational-quotes/7919-to-persist-with-a-goal-you-must-treasure-the,https://assets.passiton.com/quotes/q
PERSISTENCE,/inspirational-quotes/8300-failure-cannot-cope-with-persistence,https://assets.passiton.com/quotes/quote_art
INSPIRATION,/inspirational-quotes/8298-though-no-one-can-go-back-and-make-a-brand-new,https://assets.passiton.com/quotes
INSPIRATION,/inspirational-quotes/8297-a-highly-developed-values-system-is-like-a,https://assets.passiton.com/quotes/quo
INSPIRATION,/inspirational-quotes/7066-just-don-t-give-up-trying-what-you-really-want,https://assets.passiton.com/quotes
INSPIRATION,/inspirational-quotes/8296-when-we-strive-to-become-better-than-we-are,https://assets.passiton.com/quotes/qu
INSPIRATION,/inspirational-quotes/8299-the-most-important-thing-is-to-try-and-inspire,https://assets.passiton.com/quotes
OVERCOMING,/inspirational-quotes/6828-bad-things-do-happen-how-i-respond-to-them,https://assets.passiton.com/quotes/quot
OVERCOMING,/inspirational-quotes/8294-show-me-someone-who-has-done-something,https://assets.passiton.com/quotes/quote_ar
OVERCOMING,/inspirational-quotes/6137-its-not-the-load-that-breaks-you-down-its-the,https://assets.passiton.com/quotes/q
OVERCOMING,/inspirational-quotes/6885-getting-over-a-painful-experience-is-much-like,https://assets.passiton.com/quotes/
OVERCOMING,/inspirational-quotes/8293-if-you-cant-fly-then-run-if-you-cant-run-then,https://assets.passiton.com/quotes/q
```

Date:16/02/2022

PROGRAM NO : 17

AIM :

Python program for natural language processing - Ngram (without using inbuilt function).

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 a very good book to study', WordsToCombine=3)
print(x)
```

OUTPUT

```
"C:\python\PyCharm Community Edition 2021.2.3\FJ\venv\Scripts\python.exe" "C:/python/PyCharm Community Edition 2021.2.3/Ngram.py"
[['this', 'is', 'a'], ['is', 'a', 'very'], ['a', '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 :

Python program for natural language processing - Ngram (using inbuilt function).

PROGRAM CODE :

```
import nltk

nltk.download()

from nltk.util import ngrams

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

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
('this', 'is')
('is', 'a')
('a', 'very')
('very', 'good')
('good', 'book')
('book', 'to')
('to', 'study')

Process finished with exit code 0
```

Date:16/02/2022

PROGRAM NO : 19

AIM :

Python program for natural language processing - Speechtagging.

PROGRAM CODE :

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

```
"C:\python\PyCharm Community Edition 2021.2.3\FJ\venv\Scripts\python.exe" "C:/python/PyCharm Community Edition 2021.2.3/ngram02.py"
[('Sukanya', 'NNP'), (',', ','), ('Rajib', 'NNP'), ('Naba', 'NNP'), ('good', 'JJ'), ('friends', 'NNS'), ('.', '.')]
[('Sukanya', 'NNP'), ('getting', 'VBG'), ('married', 'VBN'), ('next', 'JJ'), ('year', 'NN'), ('.', '.')]
[('Marriage', 'NN'), ('big', 'JJ'), ('step', 'NN'), ('one', 'CD'), ("'s", 'POS'), ('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', 'VBN'), ('right', 'RB'), ('one', 'CD'), ('.', '.')]

Process finished with exit code 0
```

Date:23/02/2022

PROGRAM NO : 20

AIM :

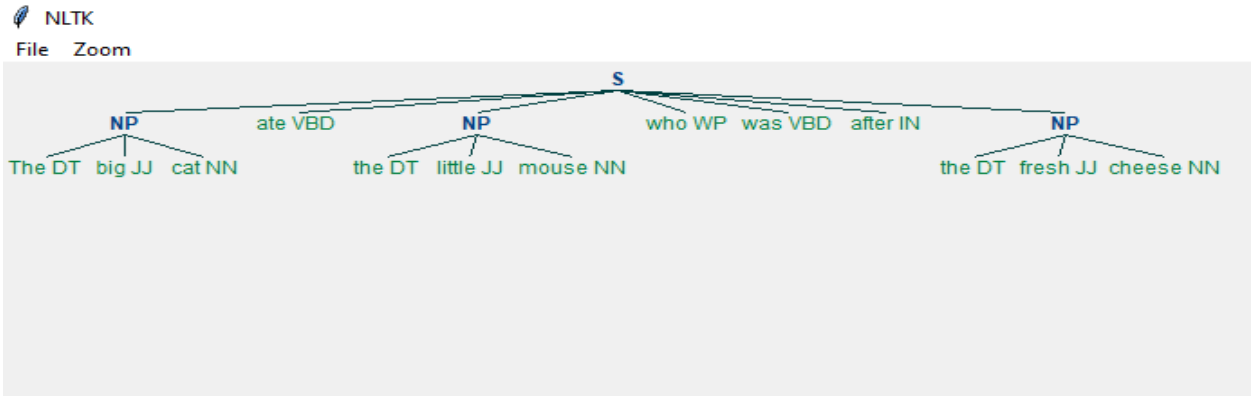
Python program for natural language processing which performs chunking.

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:\python\PyCharm Community Edition 2021.2.3\FJ\venv\Scripts\python.exe" "C:/python/PyCharm Community Edition 2021.2.3/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'),
(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))

Process finished with exit code 0
```



Date:23/02/2022

PROGRAM NO : 21

AIM :

Python program for natural language processing which performs 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)

tagged_words = nltk.pos_tag(words)

chunkGram = r"""VB: {}"""

chunkParser = nltk.RegexpParser(chunkGram)

chunked = chunkParser.parse(tagged_words)

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

OUTPUT

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\ajcemca\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
(S
  Rama/NNP
  killed/VBD
  Ravana/NNP
  to/TO
  save/VB
  sita/NN
  from/IN
  Lanka/NNP
  ./.)
```