20MCA241 DATA SCIENCE LAB

Lab Report Submitted By

BILLAN JACOB JOHN

Reg. No.: AJC20MCA-2034

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 BILLAN JACOB JOHN (Reg.No: AJC20MCA-2034) in partial fulfillment of the requirements for the award of the Degree of Masterof Computer Applications under APJ Abdul Kalam TechnologicalUniversity during the year 2021-22.

Ms. Nimmy Francis

Lab In-Charge

CONTENT

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

AIM: Perform all matrix operation using python.

PROGRAM CODE

```
import numpy as np
import random
def PrintMatrix(matrix_in):
    for x in range(0, matrix_in.shape[0]):
        for y in range(0, matrix_in.shape[1]):
            print("%d \t" % (matrix_in[x][y]), end='')
            if (y % 3 > 1):
                print("\n")
def FillMatrix(matrix_in):
    for x in range(0, matrix_in.shape[0]):
        for y in range(0, matrix_in.shape[1]):
            matrix_in[x][y] = random.randrange(2, 10) + 2
matrix1 = np.ndarray((3,3))
matrix2 = np.ndarray((3,3))
FillMatrix(matrix1)
FillMatrix(matrix2)
add_results = np.add(matrix1,matrix2)
sub_results=np.subtract(matrix1,matrix2)
mult_results=np.multiply(matrix1,matrix2)
div_results=np.divide(matrix1,matrix2)
dot_results=np.dot(matrix1,matrix2)
sqrt1_results=np.sqrt(matrix1)
sqrt2_results=np.sqrt(matrix2)
trans_results=add_results.T
print("Matrix1:")
PrintMatrix(matrix1)
print("Matrix2:")
PrintMatrix(matrix2)
print("Adding")
```

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

OUTPUT

```
Matrix1:

7      8      5
9      5      4
9      7      4
Matrix2:
8      10     6
9      6      6
10     4      6
Adding
15     18     11
18     11     10
19     11     10


Subtraction

-1     -2     -1
0      -1     -2
1      0      -5
```

```
Multiplication
56      80      30
81      30      24
90      28      24
Dot Operation
178     138     120
157     136     108
175     148     120

Squareroot Operation
matrix 1
2       2       3
3       2       2
3       2       2
matrix 2
2       3       3
3       2       2
3       2       2

Transpose
15      18      19
18      11      11
11      10      10

Process finished with exit code
```

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

AIM: Program to perform SVD (Singular value Decomposition) using Python.

PROGRAM CODE

```
from scipy. linalg import svd
from numpy import array
A= ([[3, 2, 4], [7, 3, 5]])
print(A)
X, B, T = svd(A)
print("decomposition")
print(X)
print("inverse")
print(B)
print("transpose")
print(T)
```

OUTPUT

```
[[3, 2, 4], [7, 3, 5]]
decomposition
[[-0.50093689 -0.86548381]
 [-0.86548381  0.50093689]]
inverse
[10.49777695  1.34040258]
transpose
[[-0.72026653 -0.34277021 -0.60309594]
 [ 0.6789802  -0.17021524 -0.71415171]
 [ 0.14213381 -0.92386977  0.35533453]]
Process finished with exit code
```

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

AIM: Program to implement k-NN Classification using any standard dataset available in the public domain and find the accuracy of the algorithm using in built 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
iris = load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train,y_train)
print(knn.predict(x_test))
V=knn.predict(x_test)
result=accuracy_score (y_test, V)
print ("accuracy:", result)
```

OUTPUT

```
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
accuracy: 0.9666666666666667
Process finished with exit code 0
```

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

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

PROGRAM CODE

```
from math import sqrt
def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1) - 1):
        distance += (row1[i] - row2[i]) ** 2
    return sqrt(distance)
# Locate the most similar neighbors
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
# Make a classification prediction with neighbors
def predict_classification(train, test_row, num_neighbors):
    neighbors = get_neighbors(train, test_row, num_neighbors)
    output_values = [row[-1] for row in neighbors]
    prediction = max(set(output_values), key=output_values.count)
    return prediction

# Test distance function
dataset = [[2.781, 2.550,0],
           [1.465, 2.326,3],
           [3.398, 4.429,5],
           [1.388, 1.857,11],
           [3.064, 3.393,3],
           [7.624, 2.235,4],
           [5.338, 2.775,8]]
prediction = predict_classification(dataset, dataset[0], 3)

print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
```

OUTPUT

```
Expected 2, Got 3.
Process finished with exit code 0
```

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

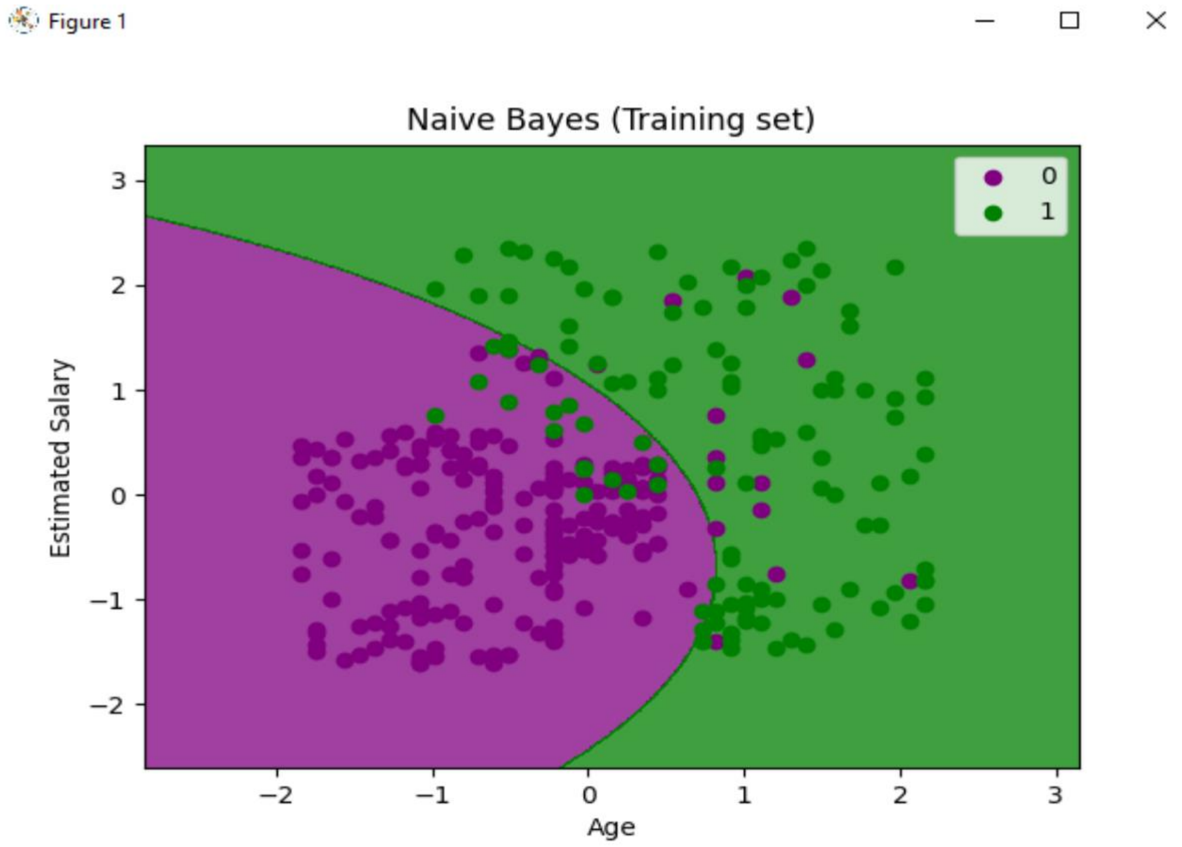
AIM: Program to implement Naïve Bayes Algorithm using any standard dataset available inthe public domain and find the accuracy of the algorithm.

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



```
[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
```

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

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

PROGRAM CODE

```
import numpy as np
from sklearn.linear_model import LinearRegression
x = np.array([2,6,7,8]).reshape((-1,1))
y = np.array([16,7,8,9])
model = LinearRegression()
model.fit(x,y)
r_sq = model.score(x,y)
print("Score: ",r_sq)
print("Intercept: ",model.intercept_)
print("Slope: ",model.coef_)
y_pred = model.predict(x)
print("Y-prediction : ",y_pred)
```

OUTPUT

```
Score:  0.7556626506024098
Intercept:  17.759036144578314
Slope:  [-1.34939759]
Y-prediction :  [15.06024096  9.6626506  8.31325301  6.96385542]
Process finished with exit code
```

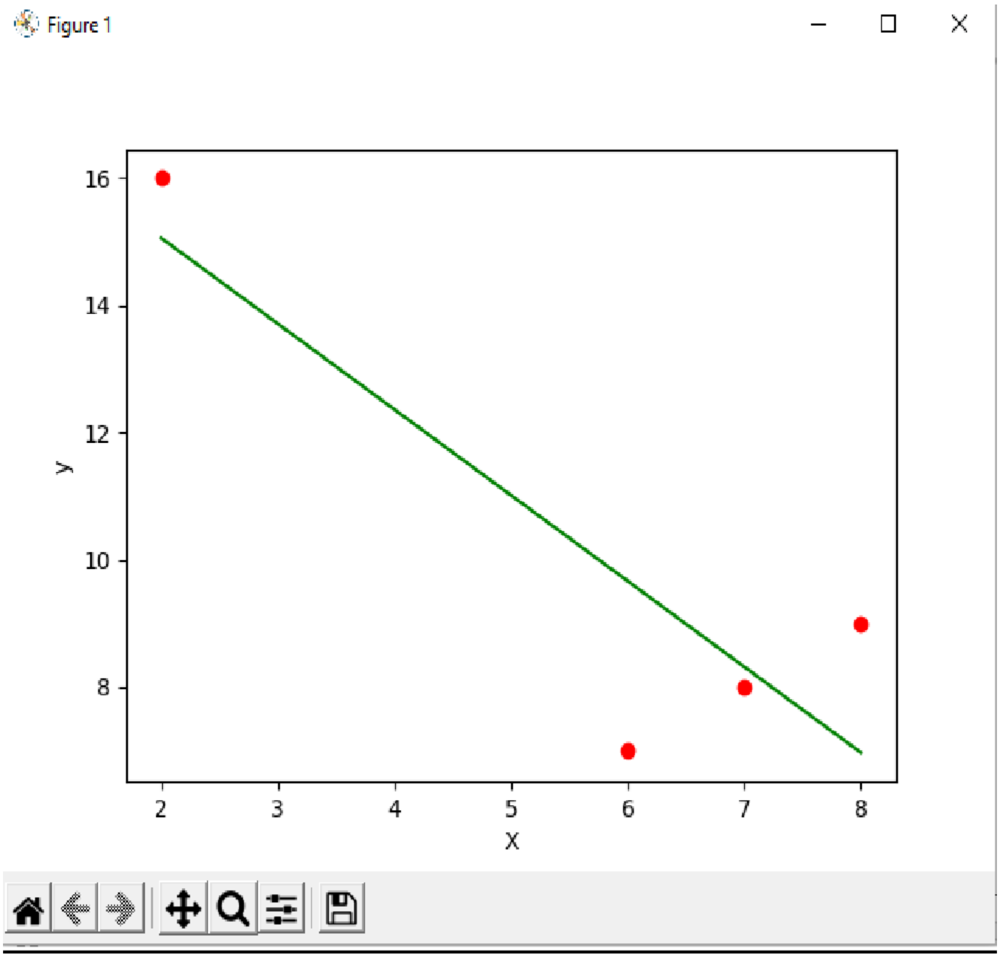
PROGRAM NO: 07 **Date :08/12/2021**

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

PROGRAM CODE

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([2,6,7,8])
y = np.array([16,7,8,9])
n = np.size(x)
n_x = np.mean(x)
n_y = np.mean(y)
SS_xy  =  np.sum(y*x)-n*  n_y*n_x
SS_xx  =  np.sum(x*x)-n*  n_x*n_x
b_1 = SS_xy/SS_xx
b_0 = n_y - b_1*n_x
y_pred = b_1 * x + b_0
print(y_pred)
plt.scatter(x, y, color='red')
plt.plot(x, y_pred, color='green')
plt.xlabel('X')
plt.ylabel('y')
plt.show()
```

OUTPUT



```
[15.06024096  9.6626506  8.31325301  6.96385542]
```

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

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

PROGRAM CODE

```
import pandas
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)
#predict the CO2
predictedCO2 = regr.predict([[2300, 1300]])
print(predictedCO2)
```

OUTPUT

```
 [107.2087328]

Process finished with exit code
```

PROGRAM NO: 09 Date :15/12/2021

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

PROGRAM CODE

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics
from sklearn.metrics import r2_score
boston = datasets.load_boston(return_X_y=False)

X = boston.data
y = boston.target
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,random_state=1)
reg = linear_model.LinearRegression()
reg.fit(X_train, y_train)
V=reg.predict(X_test)
result=r2_score(y_test, V)
print("accuracy :", result)
print('Coefficients: ', reg.coef_)
print('Variance score:{}'.format(reg.score(X_test, y_test)))
```

OUTPUT

```
accuracy : 0.7209056672661767

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

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

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

 -5.04008320e-01]

Variance score:0.7209056672661767

Process finished with exit code 0
```

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

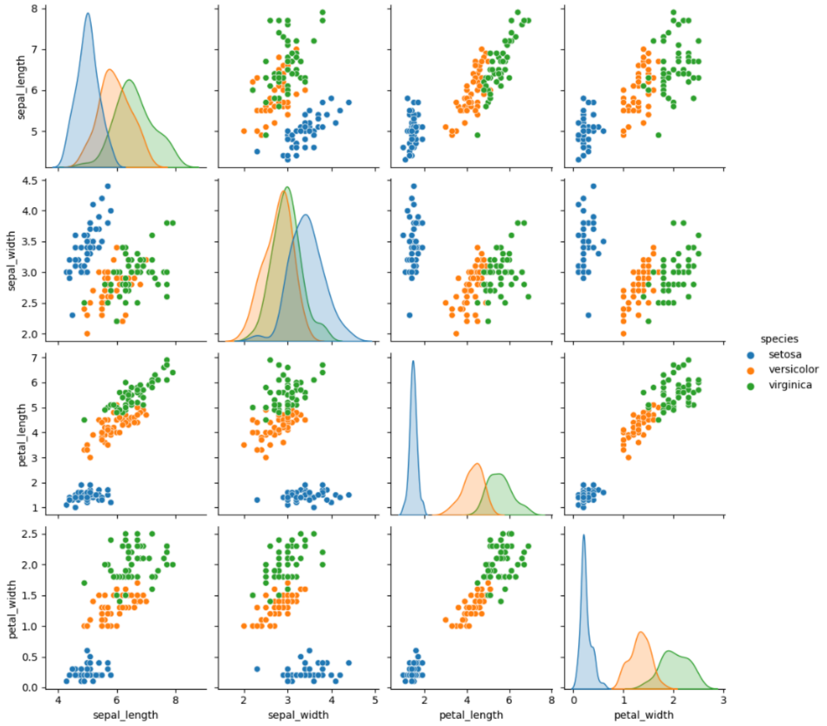
AIM: Program to implement decision tree using any standard dataset available in the public domain and find the accuracy of the algorithm

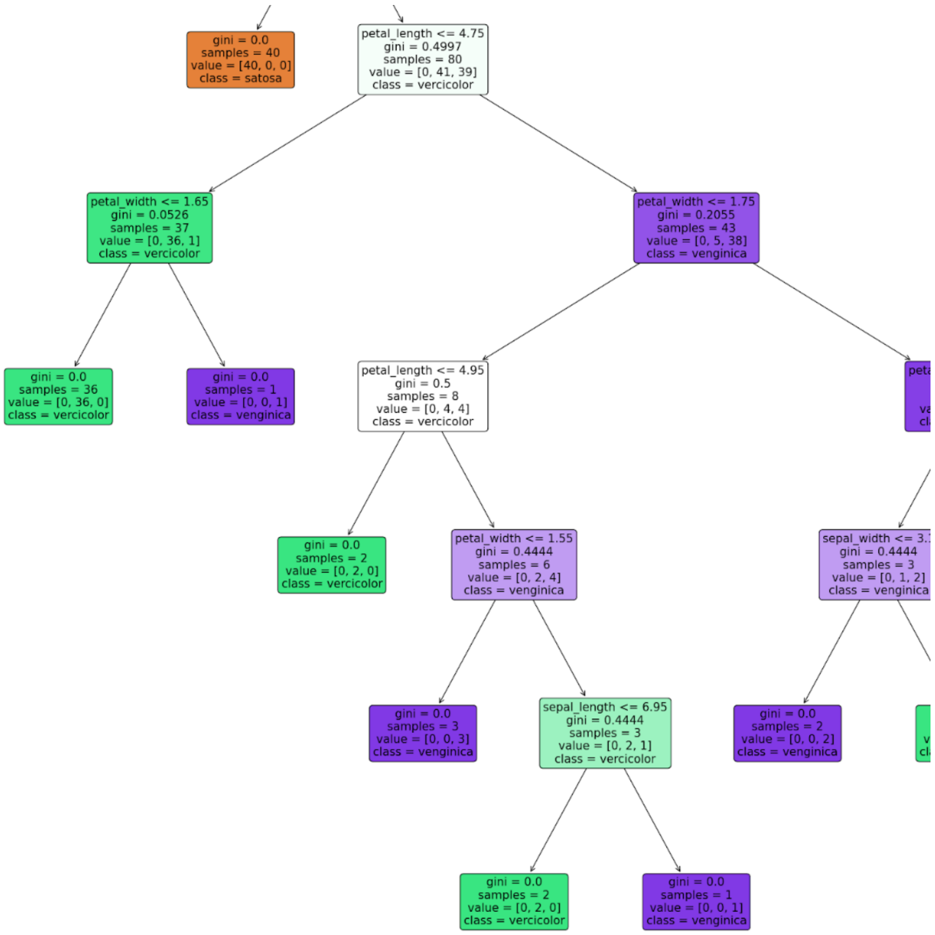
PROGRAM CODE

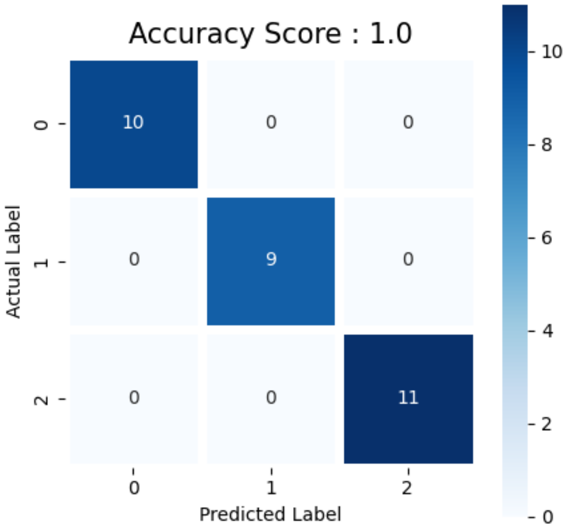
```
Import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from  sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix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

```
print(target)
y= target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state= 42)
print("training split input" , x_train.shape)
print("test split input",x_test.shape)
dtree=DecisionTreeClassifier()
dtree.fit(x_train, y_train)
print("decision tree classifer created")
y_pred = dtree.predict(x_test)
print("classification report-\n",classification_report(y_test,y_pred))
cm = confusion_matrix(y_test,y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5,annot=True,square=True,cmap='Blues')
plt.ylabel('Actual label')
plt.xlabel('predicted label')
all_sample_title = 'Accuracy Score:{0}'.format(dtree.score(x_test,y_test))
plt.title(all_sample_title,size=12)
plt.savefig("two.png")
plt.figure(figsize=(20,20))
dec_tree=plot_tree(decision_tree=dtree,feature_names=df1.columns,class_names=["setosa","vercic
olor","verginica"],filled=True ,precision=4,rounded=True)
plt.savefig("three.png")
```

OUTPUT


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica


gini = 0.0
samples = 40
value = [40, 0, 0]
class = satosa
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 = venginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = venginica
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 = venginica
sepal_width <= 3.
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = venginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = venginica
sepal_length <= 6.95
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = vercicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = venginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = venginica


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

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

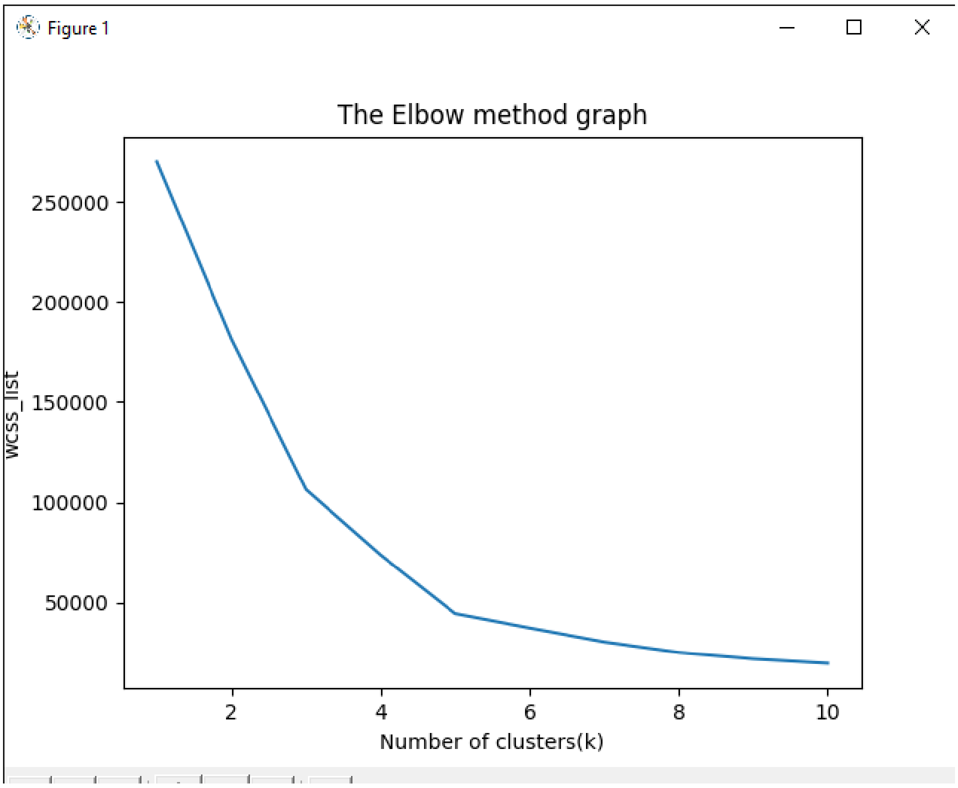
AIM: Program to implement K-Means clustering technique using any standard dataset available in the public domain

PROGRAM CODE

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

dataset=pd.read_csv('Mall_Customers.csv') x=dataset.iloc[:,[3,4]].values
print(x)

from sklearn.cluster import KMeans wcss_list=[]
for i in range(1,11):
kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42) kmeans.fit(x)
wcss_list.append(kmeans.inertia_) mtp.plot(range(1,11), wcss_list) mtp.title('The Elbow Method Graph')
mtp.xlabel('Number of clusters(k)') mtp.ylabel('wcss_list')
mtp.show()

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

dataset=pd.read_csv('Mall_Customers.csv') x=dataset.iloc[:,[3,4]].values
print(x)
from sklearn.cluster import KMeans wcss_list=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42) kmeans.fit(x)
    wcss_list.append(kmeans.inertia_) mtp.plot(range(1,11), wcss_list)
   mtp.title('The Elbow Method Graph') mtp.xlabel('Number of clusters(k)')
    mtp.ylabel('wcss_list')
mtp.show()
kmeans=KMeans(n_clusters=5,init='k-means++',random_state=42) y_predict=kmeans.fit_predict(x)
print('predict=',y_predict) mtp.scatter(x[y_predict==0,0],x[y_predict==0,1],s=100,c='blue',label='Cluster
1')
 mtp.scatter(x[y_predict==1,0],x[y_predict==1,1],s=100,c='red',label='Cluster 2')
 mtp.scatter(x[y_predict==2,0],x[y_predict==2,1],s=100,c='green',label='Cluster 3')
 mtp.scatter(x[y_predict==3,0],x[y_predict==3,1],s=100,c='yellow',label='Cluster 4')
 mtp.scatter(x[y_predict==4,0],x[y_predict==4,1],s=100,c='magenta',label='Cluster 5')

 mtp.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='black')
 mtp.title('Clusters of Customer')
 mtp.xlabel('Annual Income(k$)') mtp.ylabel('Spending Score (1-100)') mtp.legend();
 mtp.show()
```

OUTPUT

```
[[ 15  39]….

 [137  18]

 [137  83]]

predict= [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 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 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 1 4]
```



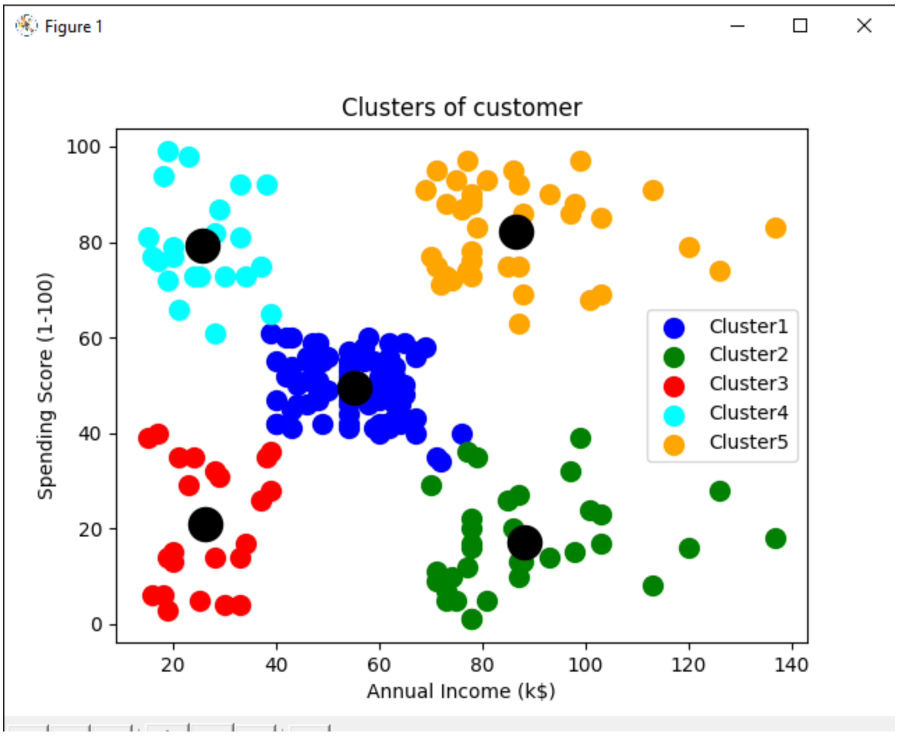


Figure 1
Clusters of customer
100
80
60
40
20
0
Spending Score (1-100)
20
40
60
80
100
120
140
Annual Income (k$)
● Cluster1
● Cluster2
● Cluster3
● Cluster4
● Cluster5

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

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

PROGRAM CODE

```
import numpy as nm
import matplotlib.pyplot as
mtpimport 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
KMeanswcss_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.inerti
a_)mtp.plot(range(1,11),
wcss_list) mtp.title('The Elbow
Method Graph')
mtp.xlabel('Number of
clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()

kmeans=KMeans(n_clusters=3,init='k-
means++',random_state=42)y_predict=kmeans.fit_predict(x)
print('predict=',y_predict)
```

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

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

mtp.scatter(x[y_predict==2,0],x[y_predict==2,1],s=100,c='green',label='Cluster 3')
mtp.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=300,c='black')
mtp.title('Clusters of world Country')
mtp.xlabel('latitude')
mtp.ylabel('longitud
e')mtp.legend();
mtp.show()
```

OUTPUT

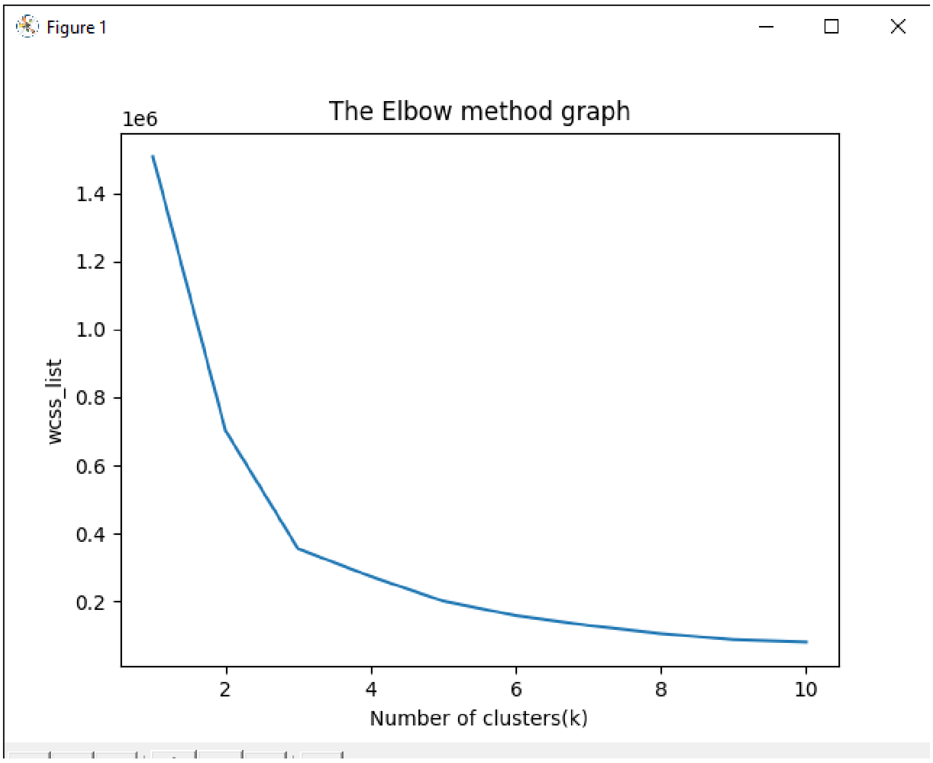


Figure 1
The Elbow method graph
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
wcss_list
2
4
6
8
10
Number of clusters(k)

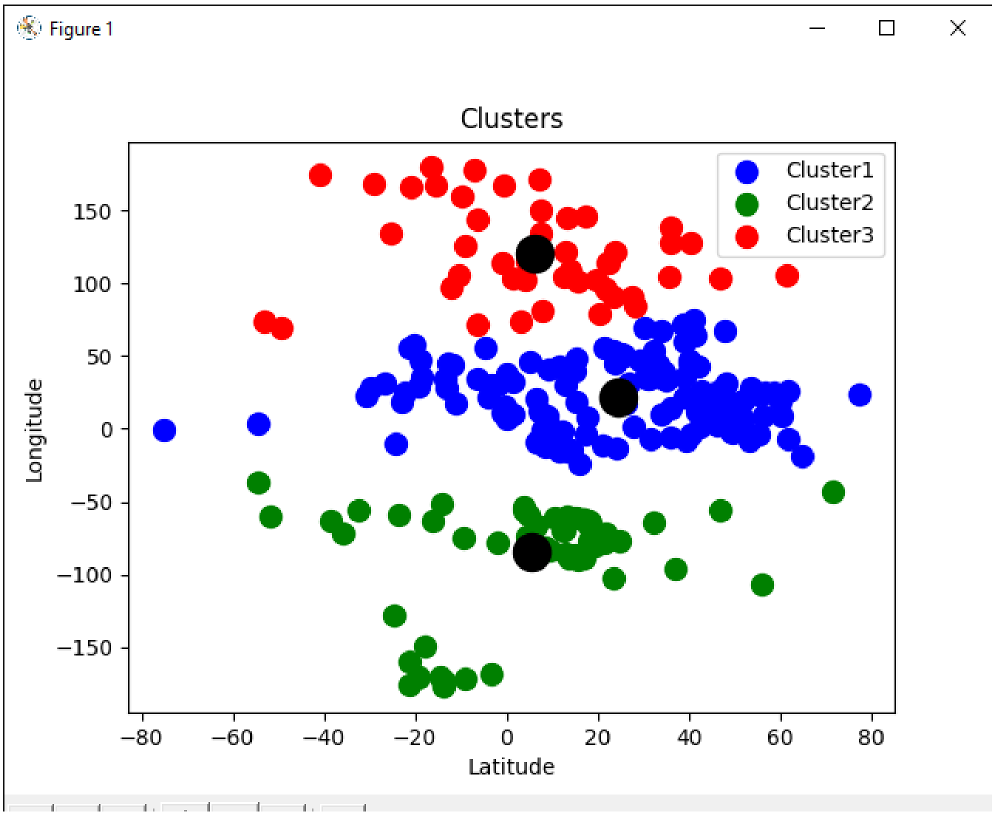


Figure 1
Clusters
Cluster1
Cluster2
Cluster3
150
100
50
0
−50
−100
−150
Longitude
−80
−60
−40
−20
0
20
40
60
80
Latitude

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

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

PROGRAM CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
np.random.seed(42)
# tf.set.random. seed(42)
fashion_mnist = keras.datasets.fashion_mnist

(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
print(X_train.shape, X_test.shape)
X_train = X_train / 255.0
X_test = X_test / 255.0
plt.imshow(X_train[1], cmap='binary')
plt.show()
np.unique(y_test)
class_names = ['T-Shirt/Top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker','8ag',
'Ankle Boot']

n_rows = 5
n_cols = 10
plt.figure(figsize=(n_cols * 1.4, n_rows * 1.6))
```

```
for row in range(n_rows): for
    col in range(n_cols):
        index = n_cols * row + col
        plt.subplot(n_rows, n_cols, index + 1)
        plt.imshow(X_train[index], cmap='binary', interpolation='nearest')
        plt.axis('off')
        plt.title(class_names[y_train[index]])
plt.show()
model_CNN = keras.models.Sequential()
model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size=7, padding='same',
activation='relu', input_shape=[28, 28, 1]))

model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))
model_CNN.add(keras.layers.Conv2D(filters=64, kernel_size=3, padding='same',
activation='relu'))

model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))
model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size=3, padding='same',
activation='relu'))

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

model_CNN.summary()
model_CNN.add(keras.layers.Flatten())
model_CNN.add(keras.layers.Dense(units=128, activation='relu'))
model_CNN.add(keras.layers.Dense(units=64, activation='relu'))
model_CNN.add(keras.layers.Dense(units=10, activation='softmax'))


model_CNN.summary()
model_CNN.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

 X_train = X_train[..., np.newaxis]
X_test = X_test[..., np.newaxis]
history_CNN = model_CNN.fit(X_train, y_train, epochs=2, validation_split=0.1)
pd.DataFrame(history_CNN.history).plot()
plt.grid(True)
plt.xlabel('epochs')
plt.ylabel('loss/accuracy')
plt.title('Training and validation plot')
plt.show()
test_loss, test_accuracy = model_CNN.evaluate(X_test, y_test)

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

OUTPUT

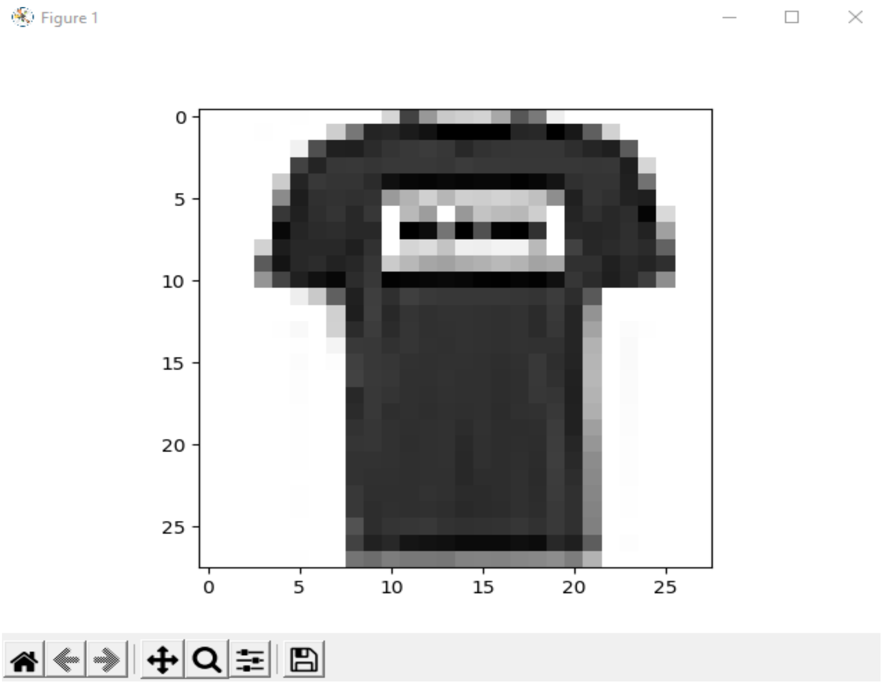


Figure 1
0
5
10
15
20
25
0
5
10
15
20
25

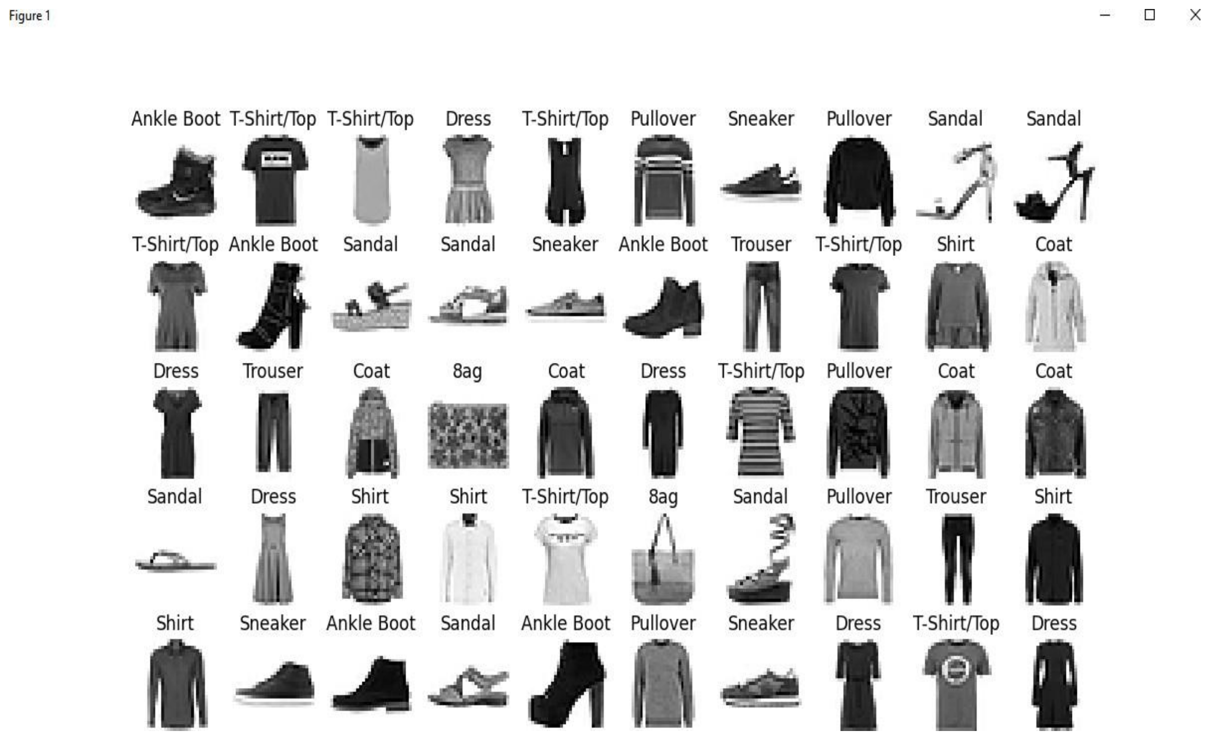
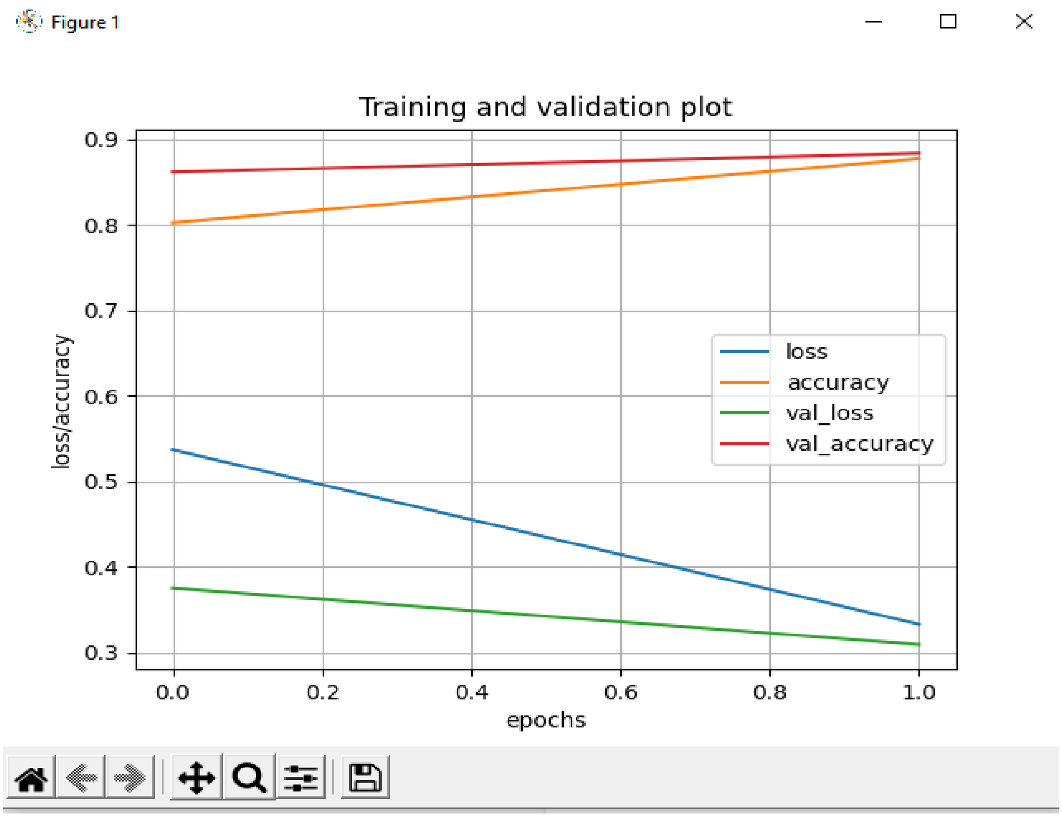
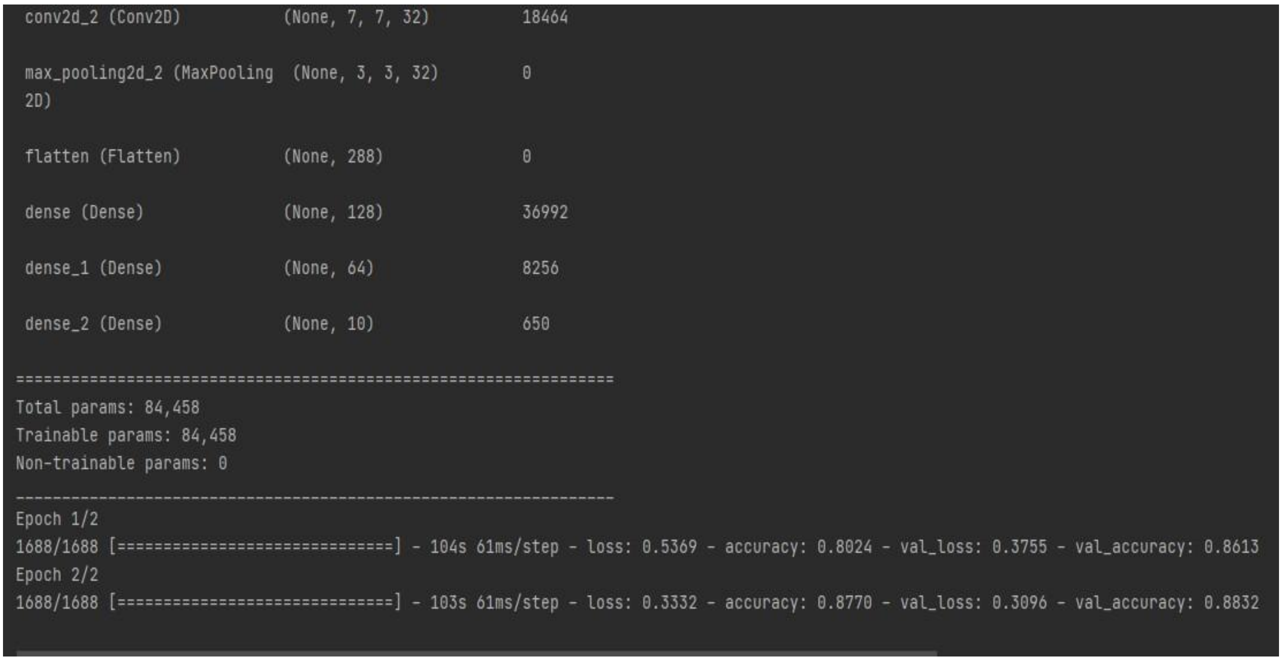


Figure 1
Ankle Boot T-Shirt/Top T-Shirt/Top Dress T-Shirt/Top Pullover Sneaker Pullover Sandal Sandal
T-Shirt/Top Ankle Boot Sandal Sandal Sneaker Ankle Boot Trouser T-Shirt/Top Shirt Coat
Dress Trouser Coat 8ag Coat Dress T-Shirt/Top Pullover Coat Coat
Sandal Dress Shirt Shirt T-Shirt/Top 8ag Sandal Pullover Trouser Shirt
Shirt Sneaker Ankle Boot Sandal Ankle Boot Pullover Sneaker Dress T-Shirt/Top Dress





```
 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 [==============================] - 104s 61ms/step - loss: 0.5369 - accuracy: 0.8024 - val_loss: 0.3755 - val_accuracy: 0.8613
Epoch 2/2
1688/1688 [==============================] - 103s 61ms/step - loss: 0.3332 - accuracy: 0.8770 - val_loss: 0.3096 - val_accuracy: 0.8832
```

PROGRAM NO: 14 Date :16/02/2022

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

PROGRAM CODE

```
import requests
import lxml
from bs4 import BeautifulSoup
#import beautifulsoup4
url = "https://www.rottentomatoes.com/top/bestofrt/"

headers = { 'User-Agents' : 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 QIHU 36OSE'}

f = requests.get(url, headers = headers)
movies_list = []
soup = BeautifulSoup(f.content, 'html.parser')
movies = soup.find('table', {'class' : 'table'}) .find_all('a')
print(movies)
num = 0

for anchor in movies:
            urls = 'https://www.rottentomatoes.com' + anchor['href']
    movies_list.append(urls)

print(movies_list)
num +=1
movie_url=urls
#movie_url=movies_lst
movie_f=requests.get(movie_url,headers=headers)
```

```
movie_soup=BeautifulSoup(movie_f.content,'lxml')
movie_content=movie_soup.find('div',{
   'class':'movie_synopsis clamp clamp-6 js-clamp'
})

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

OUTPUT

```
[<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">

['https://www.rottentomatoes.com/m/it_happened_one_night',
'https://www.rottentomatoes.com/m/citizen_kane',
'https://www.rottentomatoes.com/m/the_wizard_of_oz_1939',
'https://www.rottentomatoes.com/m/modern_times',
'https://www.rottentomatoes.com/m/black_panther_2018',
'https://www.rottentomatoes.com/m/parasite_2019',
'https://www.rottentomatoes.com/m/avengers_endgame', 'https://www.rottentomatoes.com/m/1003707-
casablanca', 'https://www.rottentomatoes.com/m/knives_out',
'https://www.rottentomatoes.com/m/us_2019', 'https://www.rottentomatoes.com/m/toy_story_4',
'https://www.rottentomatoes.com/m/lady_bird',
'https://www.rottentomatoes.com/m/mission_impossible_fallout',
'https://www.rottentomatoes.com/m/blackkklansman', 'https://www.rottentomatoes.com/m/get_out',
'https://www.rottentomatoes.com/m/the_irishman',

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 efforts to squelch the uprisings of the Algerian
War. There he faces Ali la Pointe (Brahim Haggiag), a former petty criminal who, as the leader of the
Algerian Front de Liberation Nationale, directs terror strategies against the colonial French government
occupation. As each side resorts to ever-increasing brutality, no violent act is too unthinkable.
```

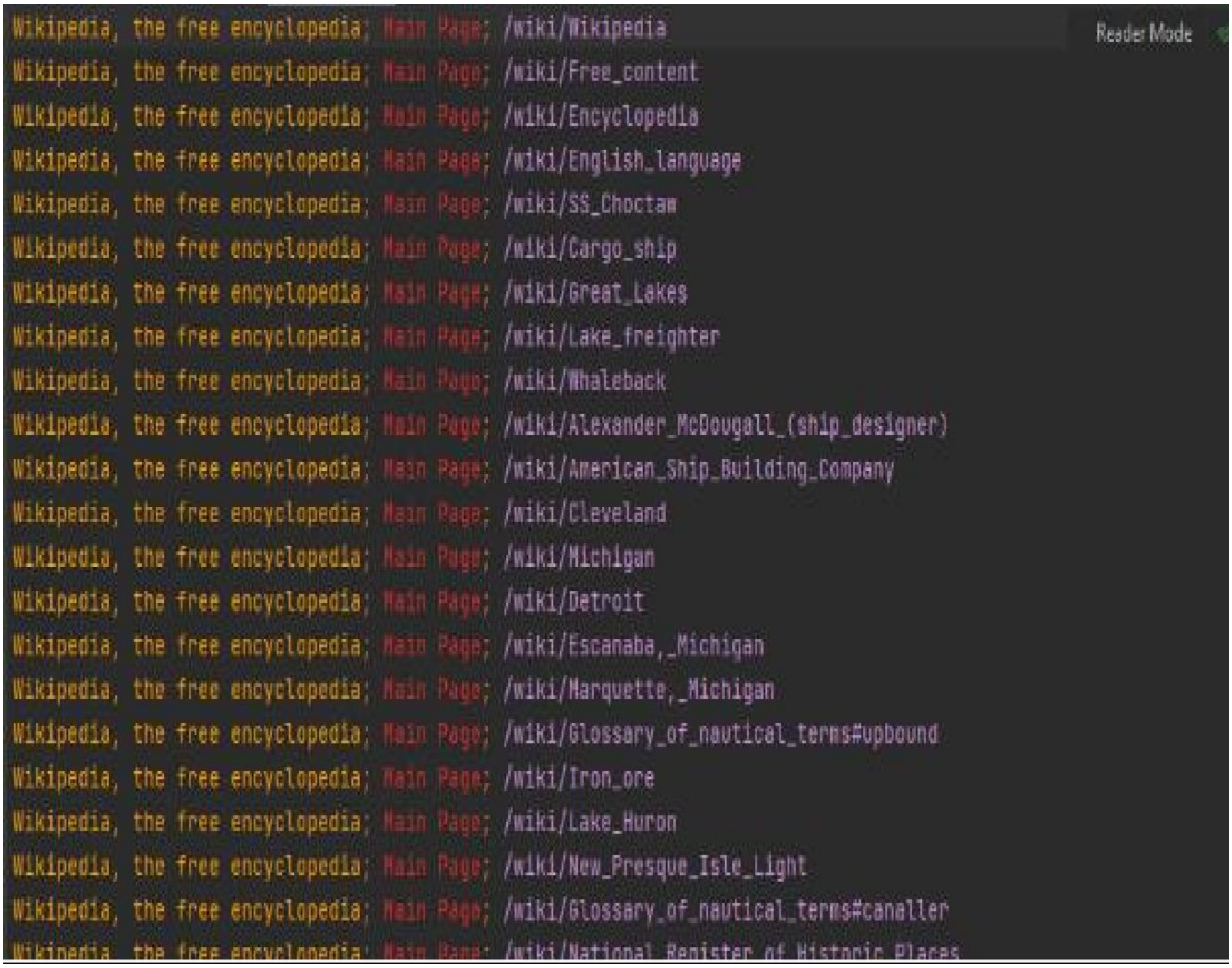
PROGRAM NO: 15 **Date :16/02/2022**

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

PROGRAM CODE

```
from bs4 import
BeautifulSoupimport
requests
pages_crawled =[
]def crawler(url):
        page =requests.get(url)
   soup=BeautifulSoup(page.text,'html.parser')
   links=soup.find_all('a')
        for link in links:

        if 'href' in link.attrs:
        if link['href'].startswith('/wiki') and ':' not in
        link['href']:if link['href'] not in pages_crawled:
            new_link =
            f"https://en.wikipedia.org{link['href']}"
            pages_crawled.append(link['href'])
            try:
               with open('data.csv','a') as file:
                    file.write(f'{soup.title.text}:{link["href"]}\
                    n')
                 crawler(new_lin
            k)except:
               continue
crawler('https://en.wikipedia.org')
```

OUTPUT



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

PROGRAM NO: 16 Date :16/02/2022

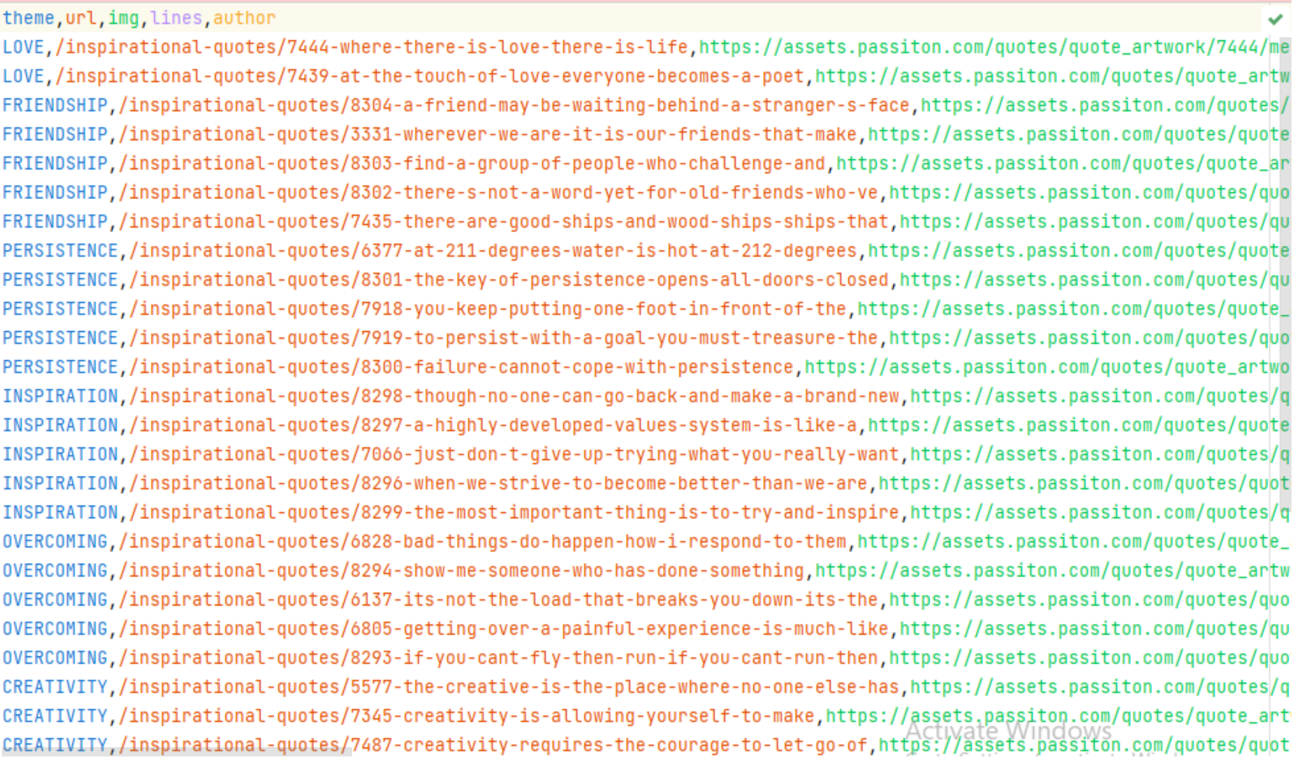
AIM: Program to implement scrap of any website.

PROGRAM CODE

```
import requests

from bs4 import BeautifulSoup
import csv
URL = "http://www.values.com/inspirational-quotes"
r = requests.get(URL)
print(r.content)

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

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

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

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

OUTPUT



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

Activate Windows

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

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

PROGRAM CODE

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

OUTPUT

```
[['this', 'is', 'a'], ['is', 'a', 'very'], ['a', 'very', 'good'], ['very', 'good', 'book'], ['good', 'book', 'to'],['book', 'to',
'study']]

Process finished with exit code 0
```

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

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

PROGRAM CODE

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

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

OUTPUT

```
('this', 'is')

('is', 'a')

('a', 'very')

('very', 'good')

('good', 'book')

('book', 'to')

('to', 'study')

Process finished with exit code 0
```

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

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

PROGRAM CODE

```
import nltk


from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize,sent_tokenize
#nltk.download('stopwords')
#nltk.download('averaged_perceptron_tagger')
stop_words=set(stopwords.words('english'))
txt="Ammu,How are you."\
        "Archana,i am fine.How are you"\
        "Sukanya is getting married next year"\
        "Marriage is a big step in ones life"\
        "yes it is a big event"\
        "okey bye Ammu."
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

```
[('Ammu', 'NNP'), (',', ','), ('How', 'NNP'), ('you.Archana', 'NNP'), (',', ','), ('fine.How', 'NN'),
('youSukanya', 'RB'), ('getting', 'VBG'), ('married', 'VBN'), ('next', 'JJ'), ('yearMarriage', 'NN'), ('big',

'JJ'), ('step', 'NN'), ('ones', 'NNS'), ('lifeyes', 'RB'), ('big', 'JJ'), ('eventokey', 'NN'), ('bye', 'NN'),
('Ammu', 'NNP'), ('.', '.')]

Process finished with exit code 0
```

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

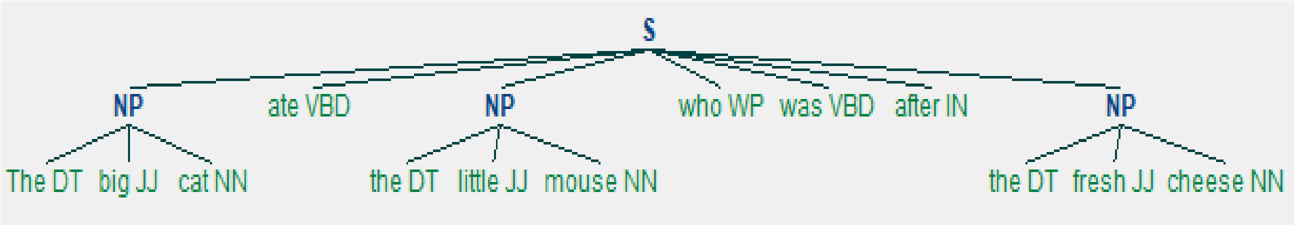
AIM: Program for Natural Language Processing which performs Chunking.

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

```
['The', 'big', 'cat', 'ate', 'the', 'little', 'mouse', 'who', 'was', 'after', 'the', 'fresh', 'cheese']
[('The', 'DT'), ('big', 'JJ'), ('cat', 'NN'), ('ate', 'VBD'), ('the', 'DT'), ('little', 'JJ'), ('mouse', 'NN'), ('who',
'WP'), ('was', 'VBD'), ('after', 'IN'), ('the', 'DT'), ('fresh', 'JJ'), ('cheese', 'NN')]

(S
(NP The/DT big/JJ cat/NN) ate/VBD
(NP the/DT little/JJ mouse/NN)
who/WPwas/VBD after/IN
(NP the/DT fresh/JJ cheese/NN))
```



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

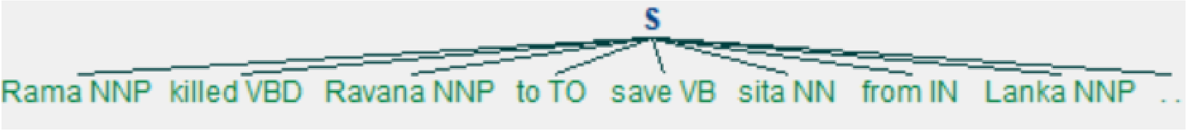
AIM: Write a python program for natural program language processing with chunking.

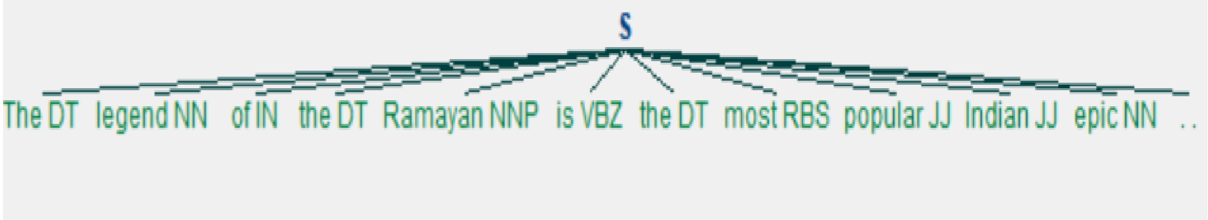
PROGRAM CODE

```
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
alreadybeen 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

```
(S
 Rama/NNP
 killed/VBD
Ravana/NNP
to/TO
save/VB
sita/NN
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
Rama NNP
killed VBD
Ravana NNP
to TO
save VB
sita NN
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
. .