Program No : 1

Aim : Perform all matrix operations using python (Using numpy)

PROGRAM

```
import numpy

array1 = []
n=int(input("Enter the array Size"))
for i in range(n):
    array1.append(int(input("Enter thr first array elements : ")))
array1=numpy.array(array1)
print(numpy.floor(array1))

array2=[]
for i in range(n):
    array2.append(int(input("Enter the second array elemensts : ")))
array2=numpy.array(array2)
print(numpy.floor(array2))

print("Array Addition ")
print(numpy.add(array1,array2))

print("Array Substraction ")
print(numpy.subtract(array1,array2))

print("Array Multiplication")
print(numpy.multiply(array1,array2))

print("Array Division")
print(numpy.divide(array1,array2))

print("Array Dot")
print(numpy.dot(array1,array2))

print("Array Squareroot")
print(numpy.sqrt(array1))

print("Array Summation of array1 ")
print(numpy.sum(array1))

print("Array Transpose of array1")
print(array1.T)
```

OUTPUT

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/Sam/numpymatrix.py
Enter the array Size 3
Enter thr first array elements : 4
Enter thr first array elements : 6
Enter thr first array elements : 10
[ 4.  6. 10.]
Enter the second array elemensts : 2
Enter the second array elemensts : 3
Enter the second array elemensts : 5
[2. 3. 5.]
Array Addition
[ 6  9 15]
Array Substraction
[2 3 5]
Array Multiplication
[ 8 18 50]
Array Division
[2. 2. 2.]
Array Dot
76
Array Squareroot
[2.         2.44948974 3.16227766]
Array Summation of array1
20
Array Transpose of array1
[ 4  6 10]
```

PROGRAM 2:

AIM: Perform SVD (Singular Value Decomposition) Using python.

PROGRAM:

```
from  numpy import array
from scipy.linalg import svd
A1= array([[2,13,3],[5,9,10],[8,7,3],[26,18,30],[22,31,45]])
print(A1)
a,b,c=svd(A1)
print(a)
print(b)
print(c)
```

OUTPUT:

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProject1/svd2.py
[[ 2 13  3]
 [ 5  9 10]
 [ 8  7  3]
 [26 18 30]
 [22 31 45]]
[[-0.12978816 -0.70009789  0.51828503 -0.21881071 -0.4201434 ]
 [-0.18827986 -0.19623861  0.00760666  0.95667704 -0.10369324]
 [-0.12501038  0.02998764  0.64353448  0.05797407  0.75231345]
 [-0.5697289   0.61976193  0.34606723 -0.03250256 -0.41289852]
 [-0.7794146  -0.2938524  -0.44432376 -0.18020408  0.27616395]]
[75.2800142  11.50945923  8.96949329]
[[-0.45378752 -0.51373277 -0.7281178 ]
 [ 0.65229561 -0.748186    0.12135957]
 [ 0.60711393  0.41987658 -0.6746231 ]]

Process finished with exit code 0
```

PROGRAM 3:

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

PROGRAM:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
irisData=load_iris()
a=irisData.data
b=irisData.target
a_train,a_test,b_train,b_test
=train_test_split(a,b,test_size=0.6,random_state=10)
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(a_train,b_train)
print(knn.predict(a_test))
x=knn.predict(a_test)
z=accuracy_score(b_test,x)
print(z)
```

OUTPUT :

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProject1/knn.py
[1 2 0 1 0 1 2 1 0 1 1 2 1 0 0 2 2 0 0 0 2 2 2 0 1 0 1 1 1 2 1 1 2 2 2 0 2
 2 2 2 0 0 1 0 2 0 1 2 2 2 1 2 1 1 1 0 0 1 0 2 0 0 1 1 2 0 2 0 1 2 0 2 2 2
 2 2 0 1 2 1 0 2 1 1 0 0 0 1 2 2]
0.9333333333333333

Process finished with exit code 0
```

PROGRAM 4

AIM : Program to implement K-NN classification using any random dataset without using inbuilt packages.

PROGRAM:

```
from math import sqrt
def euclidean_distance(row1,row2):
    distance = 0.0
    for i in range(len(row1) - 1):
        distance += (row1[i] - row2[i] )**2
        return sqrt(distance)

def get_neighbors(train,test_row, num_neighbors):
    distances = list()
    for train_row in train:
        dist =euclidean_distance(test_row, train_row)
        distances.append((train_row,dist))
        distances.sort(key=lambda  tup:tup[1])
        neighbors = list()
        for i in range(num_neighbors):
            neighbors.append(distances[i][0])
            return neighbors
def predict_classification(train,test_row, num_neighbors):
    neighbors = get_neighbors(train, test_row, num_neighbors)
    output_values = [row[-1] for row in neighbors]
    prediction = max(set(output_values), key=output_values.count)
    return prediction
dataset = [[2.7810836, 2.550537003,0],
           [1.465458936,2.64785645,0],
           [3.56789536,4.568555858,0],
           [1.468956556,3.1464756654,0],
           [5.135663212,2.621254545,0],
           [6.2545449552,5.1436870564,1],
           [8.4365631212,7.56655252636,1],
           [2.146589696,5.66655665555,1],
            [3.4664565252,5.46558866,1],
            [5.895525255,3.46565858,1]]

prediction = predict_classification(dataset,dataset[0], 5)
print('expected %d, Got %d. ' % (dataset[0][-1], prediction))
```

OUTPUT :

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProject1/knn_func.py
expected 0, Got 0.

Process finished with exit code 0
```

PROGRAM NO : 5

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

PROGRAM

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

#importing the dataset

dataset = pd.read_csv('Social_Network_Ads.csv')
x = dataset.iloc[:, [2,3]].values
y = dataset.iloc[: , -1].values

#Splitting the dataset into Training set and Test set

from sklearn.model_selection import train_test_split
x_train, x_test, y_train ,y_test = train_test_split(x,y,test_size =
0.20,random_state = 20)

#feature scaling

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

print(x_train)
print(x_test)

#training the naive bayes model on the training set

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train,y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)
print(y_pred)

# making the confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
ac = accuracy_score(y_test,y_pred)
cm = confusion_matrix(y_test,y_pred)

print(ac)
print(cm)
```

OUTPUT

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProject1/naivebayers.py
[[-0.80276277  0.44295604]
 [-0.70800656  1.43671337]
 [-0.23422551 -0.5508013 ]
 [ 0.90284902  1.16568865]
 [-1.0870314   0.4730699 ]
 [-0.89751898 -1.09285075]
 [-0.51849414  0.95489164]
 [ 1.47138628  0.41284218]
 [-1.46605624  0.38272832]
 [-1.75032487 -1.48433091]
 [-0.42373793 -1.12296461]
 [ 0.99760523 -1.00250917]
 [-0.23422551 -1.24342004]
 [ 0.90284902 -1.36387548]
 [ 0.52382418  1.82819354]
 [-1.65556866 -0.97239532]
 [ 0.14479933  0.20204517]
 [ 0.05004312 -0.5508013 ]
 [-0.61325035  0.17193131]
 [-0.32898172 -0.76159831]
 [-0.1394693   1.49694109]
 [ 1.37663007 -1.42410318]
 [ 0.52382418  1.31625794]
 [-1.84508108  0.53329761]
 [ 1.94516733  2.27990141]
 [-0.1394693  -1.06273689]
 [-1.37130003 -1.45421705]
 [ 2.03992354  1.85830739]
 [-0.80276277 -0.76159831]]
[0 1 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0
 0 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1
 0 1 0 0 1 0]
0.8875
[[44  1]
 [ 8 27]]

Process finished with exit code 0
```

PROGRAM NO : 6

AIM : Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance (Using Builtin Function)

PROGRAM

```
import numpy as np
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("Coeffient of determination : ",r_sq)
print("Intercept : ",model.intercept_)
print("Slope : ",model.coef_)

y_pred=model.predict(x)

print("Predicting Responce : ",y_pred)
```

OUTPUT

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProjec
[[ 5]
 [15]
 [25]
 [35]
 [45]
 [55]]
[ 5 20 14 32 22 38]
Coeffient of determination :  0.7158756137479542
Intercept :  5.633333333333329
Slope :  [0.54]
Predicting Responce :  [ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]

Process finished with exit code 0
```

PROGRAM NO : 7

AIM : Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance (Using Builtin Function)

and Plot

PROGRAM

```
import numpy as np
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("Coeffient of determination : ",r_sq)
print("Intercept : ",model.intercept_)
print("Slope : ",model.coef_)

y_pred=model.predict(x)

print("Predicting Responce : ",y_pred)


plt.scatter(x,y,color="m",marker="o",s=30)
plt.plot(x,y_pred,color="g")

plt.xlabel('x')
plt.ylabel('y')

plt.show()
```

OUTPUT

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProjec
[[ 5]
 [15]
 [25]
 [35]
 [45]
 [55]]
[ 5 20 14 32 22 38]
Coeffient of determination :  0.7158756137479542
Intercept :  5.633333333333329
Slope :  [0.54]
Predicting Responce :  [ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]

Process finished with exit code 0
```

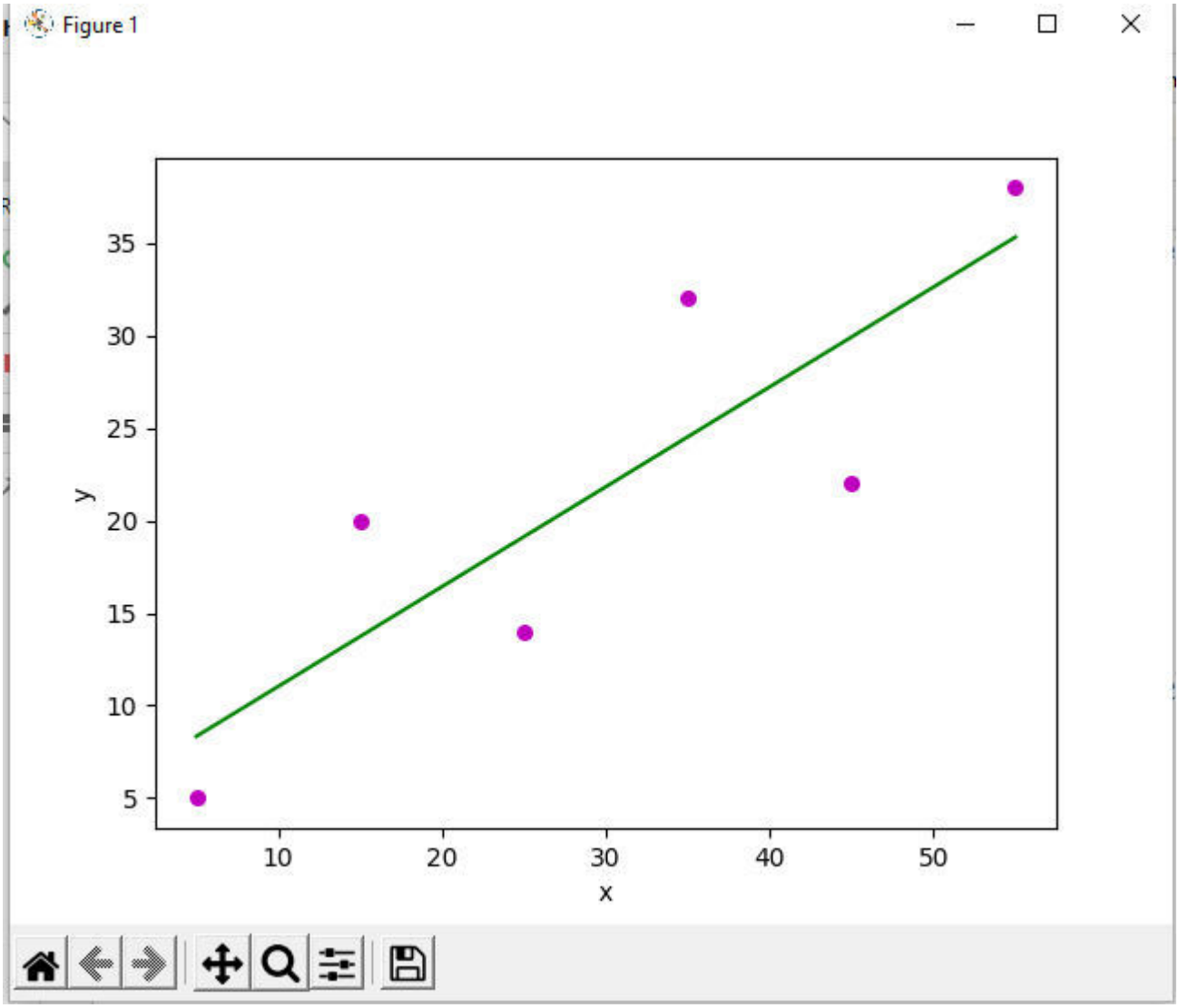


Figure 1
35
30
25
20
15
10
5
y
10
20
30
40
50
x

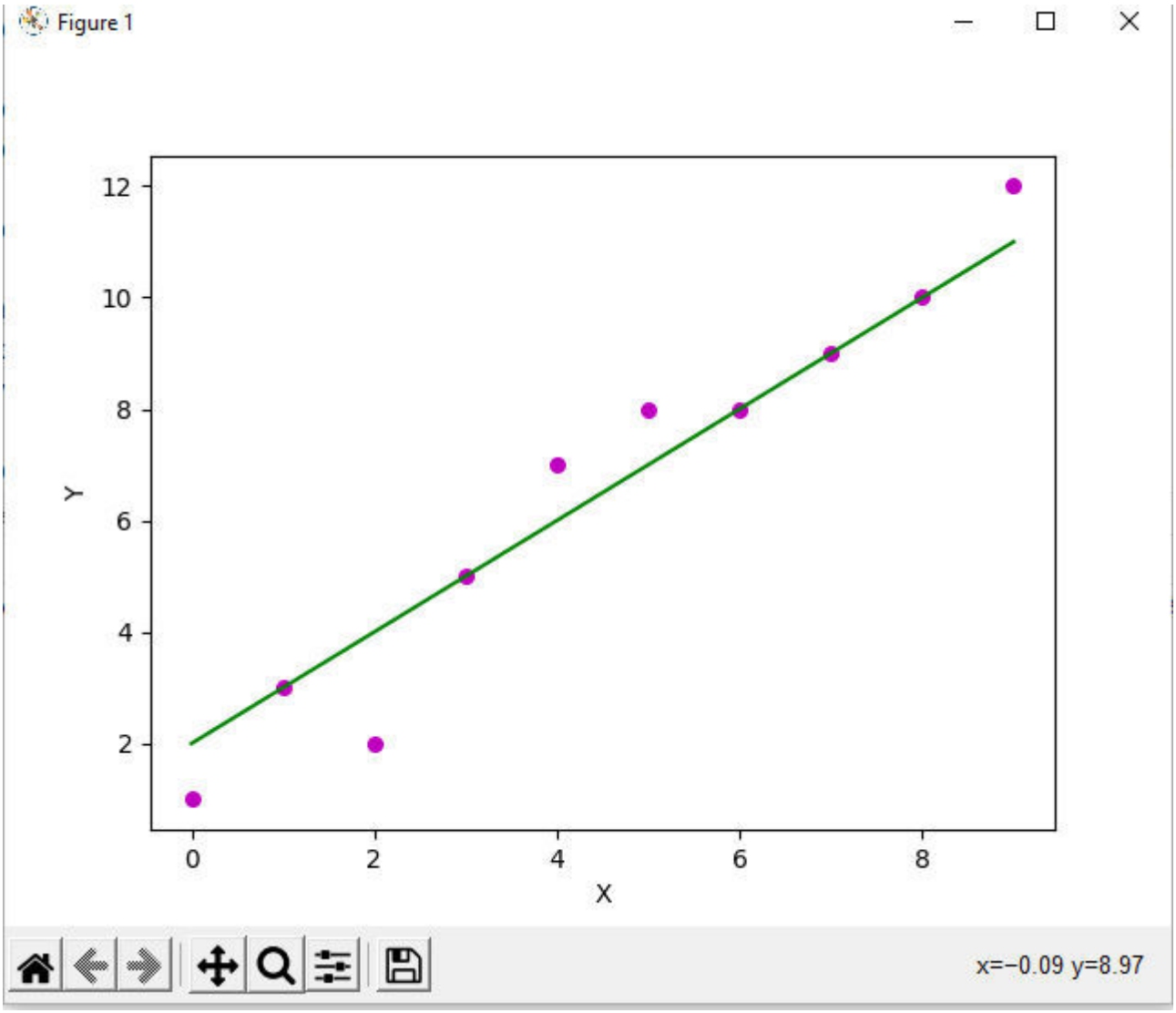
PROGRAM NO : 8

AIM : Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance (Using Without Builtin Function)

PROGRAM

```
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_y * m_x

    b_1 = SS_xx / SS_xx
    b_0 = m_y - b_1 * m_x
    #plot_regression_line()
    return  (b_0,b_1)

def plot_regression_line(x, y, b):

    plt.scatter(x, y, color = "m", marker= "o",s=30)

    y_pred = b[0] + b[1] * x

    plt.plot(x, y_pred, color="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 Coeffiecnts : \n b_0 ={} \n  b_1 ={}
".format(b[0],b[1]))

    plot_regression_line(x,y,b)

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

OUTPUT




```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe
Estimated Coeffiecnts :
 b_0 =2.0
  b_1 =1.0
```

PROGRAM NO : 9

AIM : Program to implement multiple regression techniques using any standard dataset available in the public domain and evaluate its performance (Using Without Builtin Function)

PROGRAM

```
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.values,y)

pridictedC02=regr.predict([[2300,1300]])
print(pridictedC02)
```

OUTPUT

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users
[107.2087328]

Process finished with exit code 0
```

PROGRAM NO : 10

AIM : Program to implement multiple regression techniques using any standard dataset available in the public domain and evaluate its performance (Using Without Builtin Function)

PROGRAM

```
import matplotlib.pyplot as plt
from sklearn import datasets,linear_model, metrics

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)

reg=linear_model.LinearRegression()

reg.fit(x_train,y_train)

y_pred=reg.predict(x_test)
print('Prediction',y_pred)

print('Coefficeint : ',reg.coef_)

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

OUTPUT

```
Prediction [23.67134888 21.74030058 18.99758813 25.48956674 33.81448    31.89278927
 10.78199818 11.76948128 41.56280351 25.3473233  24.372034   14.11450302
 12.40701286 27.20234921 23.43737274 32.84466914 23.60842764 32.59259513
 23.85004238 19.7314527  20.45440867 27.35630269 12.76780542 25.66660897
 29.72807157 22.15357852 30.19079303 16.04994832 28.39033945 14.35377707
 15.2366864  38.21896987 13.04273791 27.6805403  26.50783112 25.62615724
 25.38748819  6.18021821 23.83088365 39.44655193 17.18940017 35.30999805
 24.69974917  8.64460222 19.30224429 24.60568703 27.33660161  8.98033395
 23.69649295 19.01882711 29.95786303 24.28410874 25.0816296  21.47294993
 28.49968875  8.70205993 24.15032296 20.61644917 14.02028429 24.78978901
 28.82338698 25.38068056 23.46418288 22.310027   18.46758923 31.59220845
 15.36426251 20.47409342 30.44748162 36.1446885  32.21339274 16.96687201
 24.11087371 19.05568856 22.82152205 10.98287887  6.17771783  9.24239709
 15.64598693 20.63845804 33.63461858 24.08390253 32.35812922 26.61566412
 26.31571754 21.9784763  32.83918789 13.11679982 22.05714618 26.07450531
 20.03929014 30.62756585 31.36584349 27.00634194 31.2587809  10.26099249
 32.51975394 14.547433   29.30997208 24.76376686 20.66108026 21.37869093
 22.68461713 34.77810512 43.99628487 15.70849322 17.52320688 27.80934545
 20.75472888 17.49201269 18.03966614 35.70078381 12.76119622 38.97422809
  7.68925202 19.4931839  41.32259643 28.48332245 39.14616263 13.84700912
 23.11796205 21.25534719 26.62699789 37.98747407 23.72747822 30.68819329
```

```
 22.4086301  14.21437607 31.65769069 15.32061751 17.74804611  9.69417308
 24.26986084 22.15086613 10.01444493 39.73186967 22.97980241 11.67661997
 21.06713213  1.50437144 19.73119136 17.42220953 20.10684724 27.64309473
 35.94944206 18.39398149 13.3431956  22.96380391 16.99239185]
Coefficeint :  [-1.18523476e-01  3.79842025e-02  5.01458926e-03  3.46020042e+00
 -1.38245151e+01  3.77221309e+00 -2.63157836e-02 -1.40793915e+00
  2.64788312e-01 -1.08549788e-02 -7.87130516e-01  1.00651827e-02
 -4.66957118e-01]
Variance scores : 0.7465776075198429

Process finished with exit code 0
```

PROGRAM NO : 11

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

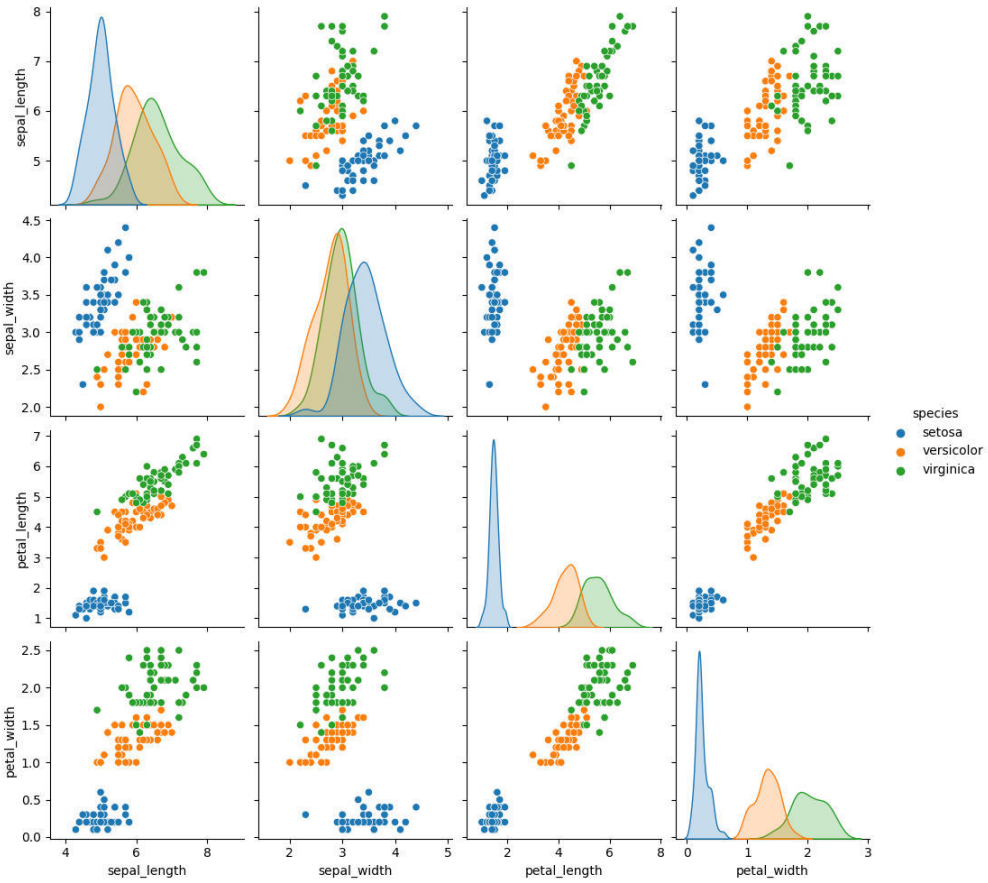
PROGRAM

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

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("decison_tree.png")

#correlation matrix
sns.heatmap(df.corr())
plt.savefig("one.png")

target=df['species']

df1=df.copy()
df1=df1.drop('species',axis=1)

print(df1.shape)
print(df1.head())

#defining the attribute
x=df1;
print(target)

#label encoding
le=LabelEncoder()
target=le.fit_transform(target)
print(target)

y=target
```

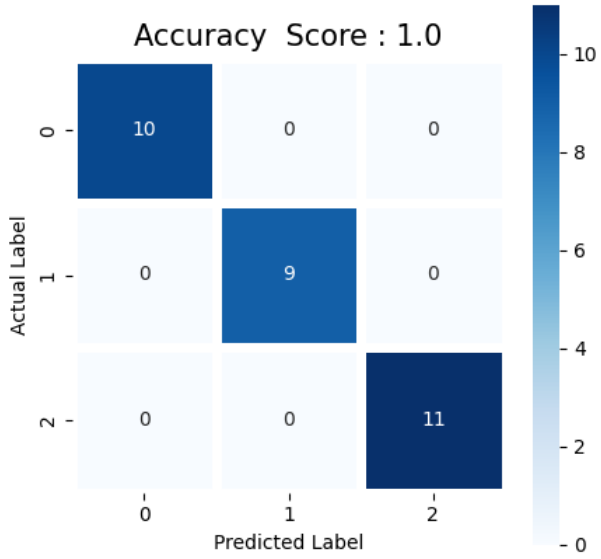
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)

print('Training split input- ',x_train.shape)
print('testing split input- ',x_test.shape)

#Defing the Decision tree algorithm
dtree=DecisionTreeClassifier()
dtree.fit(x_train,y_train)

y_pred=dtree.predict(x_test)

print('Classification Report - \n',classification_report(y_test,y_pred))

cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(5,5))

sns.heatmap(data=cm,linewidth=5,annot=True,square=True,cmap="Blues")

plt.ylabel("Actual Label")
plt.xlabel("Predicted Label")

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

plt.savefig("2.png")

#visualizong the graph without the use of graphics



plt.figure(figsize=(20,20))
dec_tre=plot_tree(decision_tree=dtree,feature_names=df1.columns,class_names=["satosa","vercicolor","v
enginica"],filled=True,precision=4,rounded=True)

plt.savefig("3.png")
```

OUTPUT

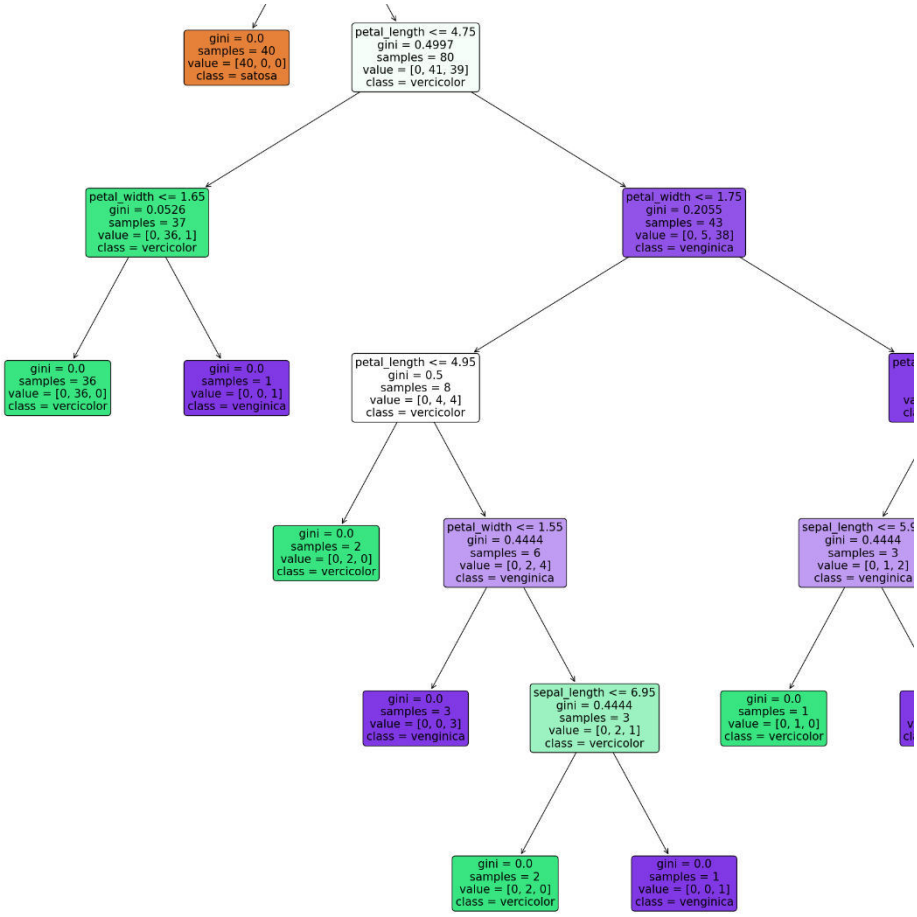
1.png



2.png



3.png


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_length <= 5.9
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 = 1
value = [0, 1, 0]
class = vercicolor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = vercicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = venginica

decison_tree

```
C:\Programming\Python39\python.exe C:/Users/asifk/PycharmProjects/ML/venv/22-21-2021/decison_tree.py
   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
```

Run TODO Problems Debug Python Packages Python Console Terminal

decison_tree

```
2         setosa
3         setosa
4         setosa
         ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
Training split input-  (120, 4)
testing split input-  (30, 4)
Classification Report -
              precision    recall  f1-score   support

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

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


Process finished with exit code 0
```

PROGRAM NO : 12

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

PROGRAM

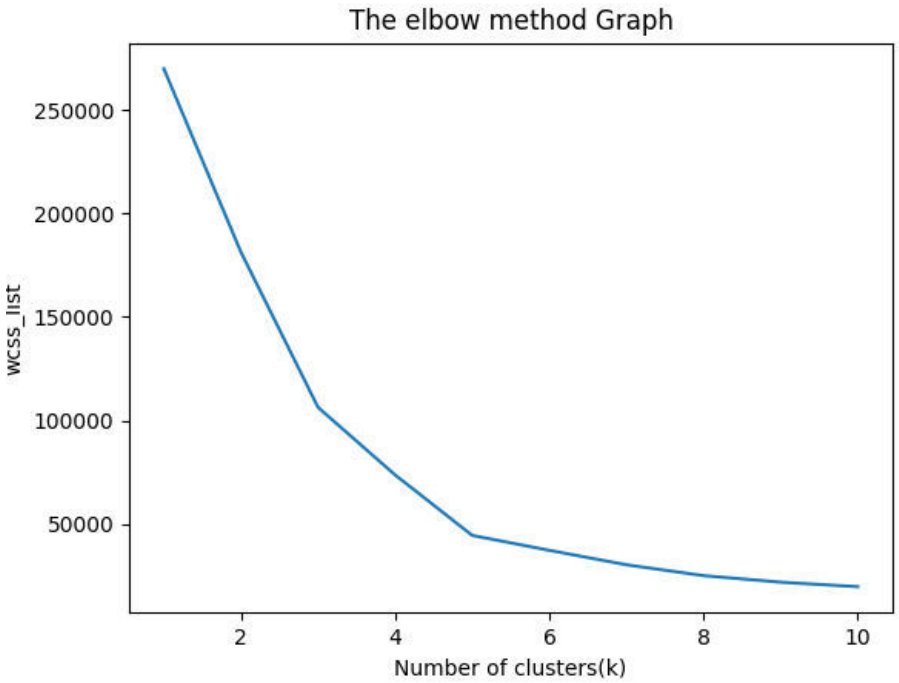
```
import matplotlib.pyplot as mtp
import pandas as pd

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

#find elbow
from sklearn.cluster import KMeans

wcss_list=[]  #initializing the list for the values of WCSS (sum of squared distance b/w each value)

#using the loop for iteration from 1 to 10
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()

#traning thr  K-Means model pm a dataset

kmeans=KMeans(n_clusters=5,init='k-means++',random_state=42)
y_predict=kmeans.fit_predict(x)
print(y_predict)

#Visualizing the Clusters
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')

mtp.title("Clusters of Customers")
```

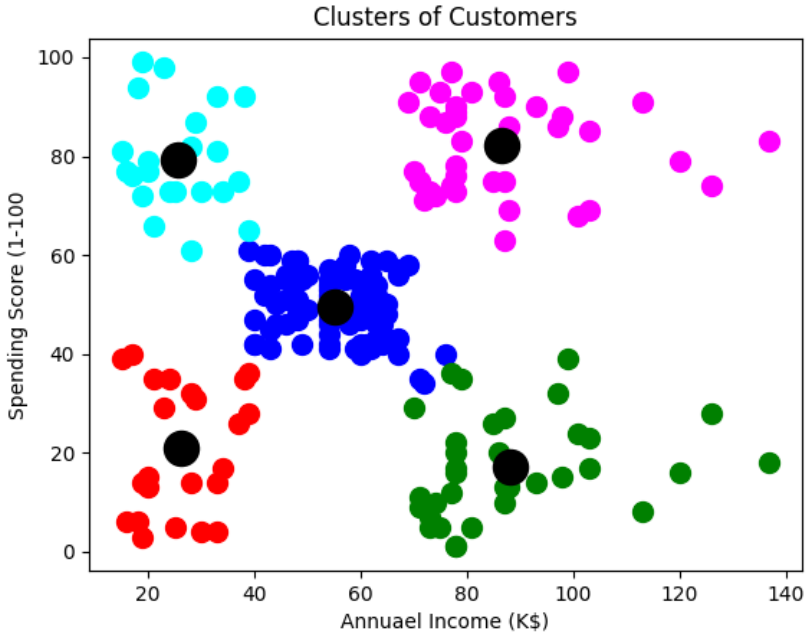
```
mtp.xlabel("Annuael Income (K$)")
mtp.ylabel("Spending Score (1-100")
mtp.legend()
mtp.show()
```

OUTPUT

```
Run:  k-means_clustering
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/05-01-2022/k-means_clustering.py
[[ 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]
 [ 21  66]
 [ 23  29]
```

```
 [120  16]
 [120  79]
 [126  28]
 [126  74]
 [137  18]
 [137  83]]
[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]
```


The elbow method Graph
wcss_list
250000
200000
150000
100000
50000
2
4
6
8
10
Number of clusters(k)


Clusters of Customers
Spending Score (1-100
100
80
60
40
20
0
20
40
60
80
100
120
140
Annuael Income (K$)

PROGRAM NO : 13

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

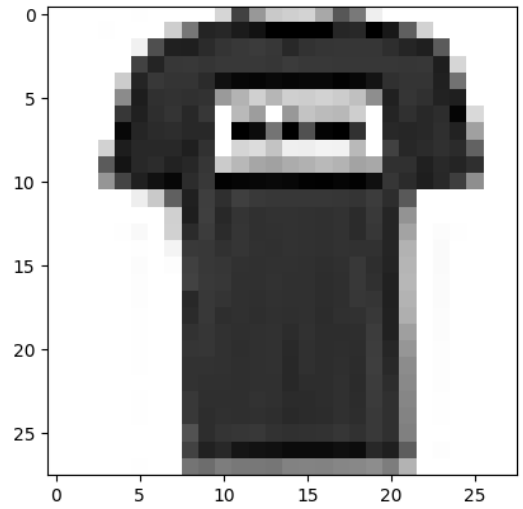
PROGRAM

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

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_name = ['T-
shirt/Top','Trouser','Pullover','Dress','Cost','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_name[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'))
```

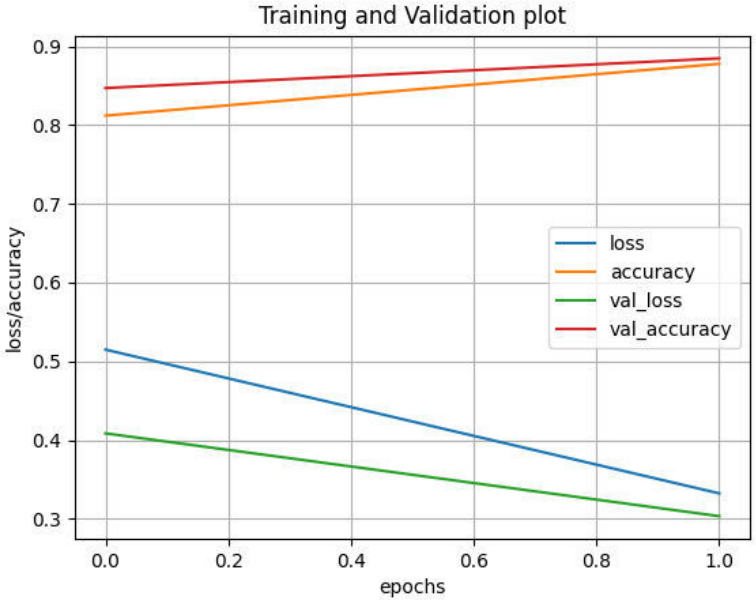
```
model_CNN.summary()

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

```
 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
_________________________________________________________________
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 [==============================] - 38s 22ms/step - loss: 0.5148 - accuracy: 0.8119 - val_loss: 0.4085 - val_accuracy: 0.8470
Epoch 2/2
1688/1688 [==============================] - 38s 22ms/step - loss: 0.3323 - accuracy: 0.8777 - val_loss: 0.3034 - val_accuracy: 0.8847
313/313 [==============================] - 2s 6ms/step - loss: 0.3215 - accuracy: 0.8789
Test Loss : 0.32149529457092285,Test Accuracy : 0.8788999915122986
```




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
Cost
Dress
Trouser
Cost
Bag
Cost
Dress
T-shirt/Top
Pullover
Cost
Cost
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


Training and Validation plot
loss/accuracy
epochs
loss
accuracy
val_loss
val_accuracy

Program No 14

Program to implement a webcrawler

Code

```
import requests
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,'lxml')
movies = soup.find('table', {
   'class': 'table'
}).find_all('a')
print(movies)
num = 0
for anchor in movies:
   urls = 'https://www.rottentomatoes.com' + anchor['href']
   movies_lst.append(urls)
print(movies_lst)
num += 1
movie_url = urls
movie_f = requests.get(movie_url, headers=headers)
movie_soup = BeautifulSoup(movie_f.content, 'lxml')
movie_content = movie_soup.find('div', {'class': 'movie_synopsis clamp clamp-6 js-clamp'})
print(num, urls, '\n', 'Movie:' + anchor.string.strip())

print('Movie info:' + movie_content.string.strip()
```

Output

```
C:\Programming\Python39\python.exe C:/Users/asifk/PycharmProjects/ML/16-02-2022/webcrawler.py
[<a class="unstyled articleLink" href="/m/it_happened_one_night">
            It Happened One Night (1934)</a>, <a class="unstyled articleLink" href="/m/citizen_kane">
            Citizen Kane (1941)</a>, <a class="unstyled articleLink" href="/m/the_wizard_of_oz_1939">
            The Wizard of Oz (1939)</a>, <a class="unstyled articleLink" href="/m/modern_times">
            Modern Times (1936)</a>, <a class="unstyled articleLink" href="/m/black_panther_2018">
            Black Panther (2018)</a>, <a class="unstyled articleLink" href="/m/parasite_2019">
            Parasite (Gisaengchung) (2019)</a>, <a class="unstyled articleLink" href="/m/avengers_endgame">

             Baby Driver (2017)</a>, <a class="unstyled articleLink" href="/m/spider_man_homecoming">
             Spider-Man: Homecoming (2017)</a>, <a class="unstyled articleLink" href="/m/godfather_part_ii">
             The Godfather, Part II (1974)</a>, <a class="unstyled articleLink" href="/m/the_battle_of_algiers">
             The Battle of Algiers (La Battaglia di Algeri) (1967)</a>]
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentoma

Process finished with exit code 0
```

Program No 15

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

Code:

```
from bs4 import BeautifulSoup
import requests

pages_crawled = []

def crawler(url):
    page = requests.get(url)
    soup = BeautifulSoup(page.text, 'html.parser')
    links = soup.find_all('a')

    for link in links:
        if 'href' in link.attrs:
            if link['href'].startswith('/wiki') and ':' not in link['href']:
                if link['href'] not in pages_crawled:
                    new_link = f"https:en.wikipedia.org{link['href']}"
                    pages_crawled.append(link['href'])
                    try:
                        with open('data.csv', 'a') as file:
                            file.write(f'{soup.title.text};{soup.h1.text};{link["href"]}\n')
                        crawler(new_link)
                    except:
                        continue
crawler('https://en.wikipedia.org')
```

Output

```
Wikipedia, the free encyclopedia;Main Page;/wiki/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/Wonderful_Parliament
Wikipedia, the free encyclopedia;Main Page;/wiki/Legislative_session
Wikipedia, the free encyclopedia;Main Page;/wiki/Parliament_of_England
Wikipedia, the free encyclopedia;Main Page;/wiki/Westminster_Abbey
Wikipedia, the free encyclopedia;Main Page;/wiki/Richard_II_of_England
Wikipedia, the free encyclopedia;Main Page;/wiki/Favourite
Wikipedia, the free encyclopedia;Main Page;/wiki/Hundred_Years%27_War
Wikipedia, the free encyclopedia;Main Page;/wiki/Lord_Chancellor
Wikipedia, the free encyclopedia;Main Page;/wiki/Michael_de_la_Pole,_1st_Earl_of_Suffolk
Wikipedia, the free encyclopedia;Main Page;/wiki/Impeachment
```

Program No 16

Implement a program to scrap the webpage of any popular website

Code

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

url = "https://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

```
C:\Programming\Python39\python.exe C:/Users/asifk/PycharmProjects/ML/16-02-2022/scrap.py
b'<!DOCTYPE html>\n<html class="no-js" dir="ltr" lang="en-US">\n    <head>\n        <title>Inspirational Quotes - Motivational Quotes - Leadership Quotes | PassItOn.com</title>\n
<!DOCTYPE html>
<html class="no-js" dir="ltr" lang="en-US">
 <head>
  <title>
   Inspirational Quotes - Motivational Quotes - Leadership Quotes | PassItOn.com
  </title>
  <meta charset="utf-8"/>
  <meta content="text/html; charset=utf-8" http-equiv="content-type"/>
  <meta content="IE=edge" http-equiv="X-UA-Compatible"/>
  <meta content="width=device-width,initial-scale=1.0" name="viewport"/>
  <meta content="The Foundation for a Better Life | Pass It On.com" name="description"/>
  <link href="/apple-touch-icon.png" rel="apple-touch-icon" sizes="180x180"/>
  <link href="/favicon-32x32.png" rel="icon" sizes="32x32" type="image/png"/>

    </div>
   </div>
  </div>
  </footer>
  <a class="scroll-top-arrow" href="javascript:void(0);">
   <i class="ti-arrow-up">
   </i>
  </a>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.js">
  </script>
  <script crossorigin="anonymous" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/u
  </script>
  <script crossorigin="anonymous" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/boot
  </script>
  <script src="/assets/pofo-1a7dc0d92519266568dcfcc8a6e53534.js">
  </script>
 </body>
</html>


Process finished with exit code 0
```

Program No 17

Python program for natural language processing-Ngram(without using in-built functions)

Code

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

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

Output

```
C:\Programming\Python39\python.exe C:/Users/asifk/PycharmProjects/ML/16-02-2022/ngram.py
[['this', 'is', 'a', 'very']]

Process finished with exit code 0
```

Program No 18

Python program for natural language processing-Ngram(with using in-built functions)

Code

```
import nltk
nltk.download()
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

```
C:\Programming\Python39\python.exe "C:/Users/asifk/PycharmProjects/ML/16-02-2022/ngram withbuiltin funct.py"
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
```

Program No 19

Python program for natural language processing- part of speech tagging

Code

```
from cgitb import text

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

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

text
txt = "Sukanya, Rajib and Naba are my good friends. " \
     "Sukanya is getting married next year. " \
     "Marriage is a big step in ones'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

```
C:\Programming\Python39\python.exe C:/Users/asifk/PycharmProjects/ML/16-02-2022/nlp.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'), ('ones', 'NNS'), ("'s", 'POS'), ('lifeIt', '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
```

Program no:20

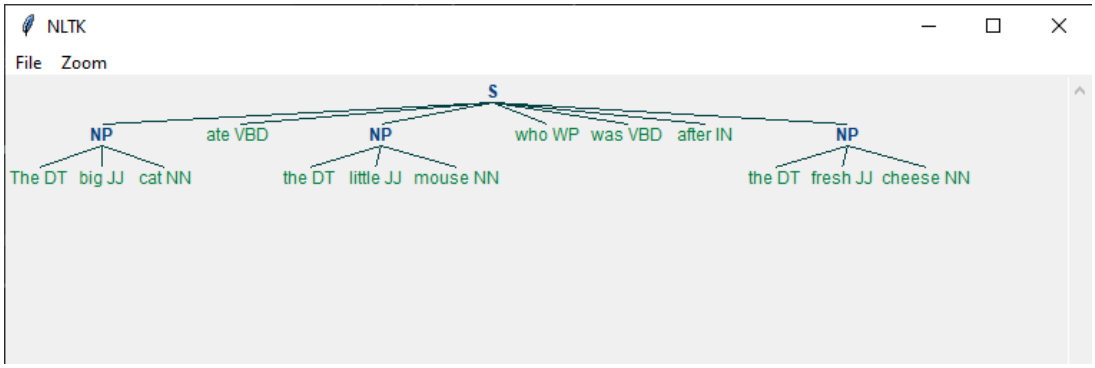
Aim: Program for Natural Language Processing which performs Chunking.

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

```
C:\Users\mca\PycharmProjects\pythonProject123\venv\Scripts\python.exe "C:/Users/mca/PycharmProjects/pythonProject123/Ml programs/jk.py"
['The', 'big', 'cat', 'ate', 'the', 'little', 'mouse', 'who', 'was', 'after', 'the', 'fresh', 'cheese']
[('The', 'DT'), ('big', 'JJ'), ('cat', 'NN'), ('ate', 'VBD'), ('the', 'DT'), ('little', 'JJ'), ('mouse', 'NN'), ('who', 'WP'), ('was', 'VBD'), ('after', 'IN'
(S
  (NP The/DT big/JJ cat/NN)
  ate/VBD
  (NP the/DT little/JJ mouse/NN)
  who/WP
  was/VBD
  after/IN
  (NP the/DT fresh/JJ cheese/NN))
```



PROGRAM NO : 21

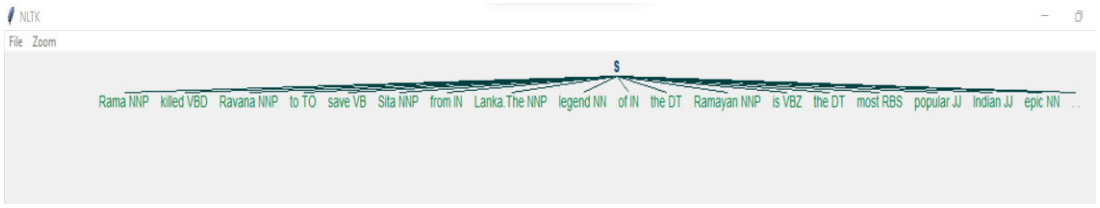
Aim: Write a python program for natural program language processing with chunking.

Program:

```
import nltk
#nltk.download('averaged_perception_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 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

```
C:\Programming\Python39\python.exe "C:/Users/asifk/PycharmProjects/ML/23-02-2022/chunking 2.py"
(S
  Rama/NNP
  killed/VBD
  Ravana/NNP
  to/TO
  save/VB
  Sita/NNP
  from/IN
  Lanka.The/NNP
  legend/NN
  of/IN
  the/DT
  Ramayan/NNP
  is/VBZ
  the/DT
  most/RBS
  popular/JJ
  Indian/JJ
  epic/NN
  ./.)
```


NLTK
File Zoom
S
Rama NNP
killed VBD
Ravana NNP
to TO
save VB
Sita NNP
from IN
Lanka.The NNP
legend NN
of IN
the DT
Ramayan NNP
is VBZ
the DT
most RBS
popular JJ
Indian JJ
epic NN
. .