AIM:1

Program to implement Naive Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

PROGRAM

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

# 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 the 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 = 0)

# 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_train)

# 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, acc
```

OUTPUT

```
[ 7.56421121e-01 -8.38107706e-01]
 [ 2.70367388e-01 -2.87638347e-01]
 [ 3.67578135e-01 -1.71750061e-01]
 [-1.18475597e-01  2.20395980e+00]
 [-1.47942605e+00 -6.35303205e-01]
 [-1.28500455e+00 -1.06988428e+00]
 [-1.38221530e+00  4.07691369e-01]
 [-1.09058306e+00  7.55356227e-01]
 [-1.47942605e+00 -2.00722133e-01]
 [ 9.50842613e-01 -1.06988428e+00]
 [ 9.50842613e-01  5.81523798e-01]
 [ 3.67578135e-01  9.87132798e-01]
 [ 5.61999628e-01 -8.96051849e-01]
 [-6.04529329e-01  1.45068594e+00]
 [-2.12648508e-02 -5.77359062e-01]
```

```
C:\Users\ajcemca\PycharmProjects\file\venv\Scripts\python.exe C
[[ 1.92295008e+00  2.14601566e+00]
 [ 2.02016082e+00  3.78719297e-01]
 [-1.38221530e+00 -4.32498705e-01]
 [-1.18779381e+00 -1.01194013e+00]
 [ 1.92295008e+00 -9.25023920e-01]
 [ 3.67578135e-01  2.91803083e-01]
 [ 1.73156642e-01  1.46942725e-01]
 [ 2.02016082e+00  1.74040666e+00]
 [ 7.56421121e-01 -8.38107706e-01]
 [ 2.70367388e-01 -2.87638347e-01]
 [ 3.67578135e-01 -1.71750061e-01]
 [-1.18475597e-01  2.20395980e+00]
 [-1.47942605e+00 -6.35303205e-01]
 [-1.28500455e+00 -1.06988428e+00]
```

AIM:2

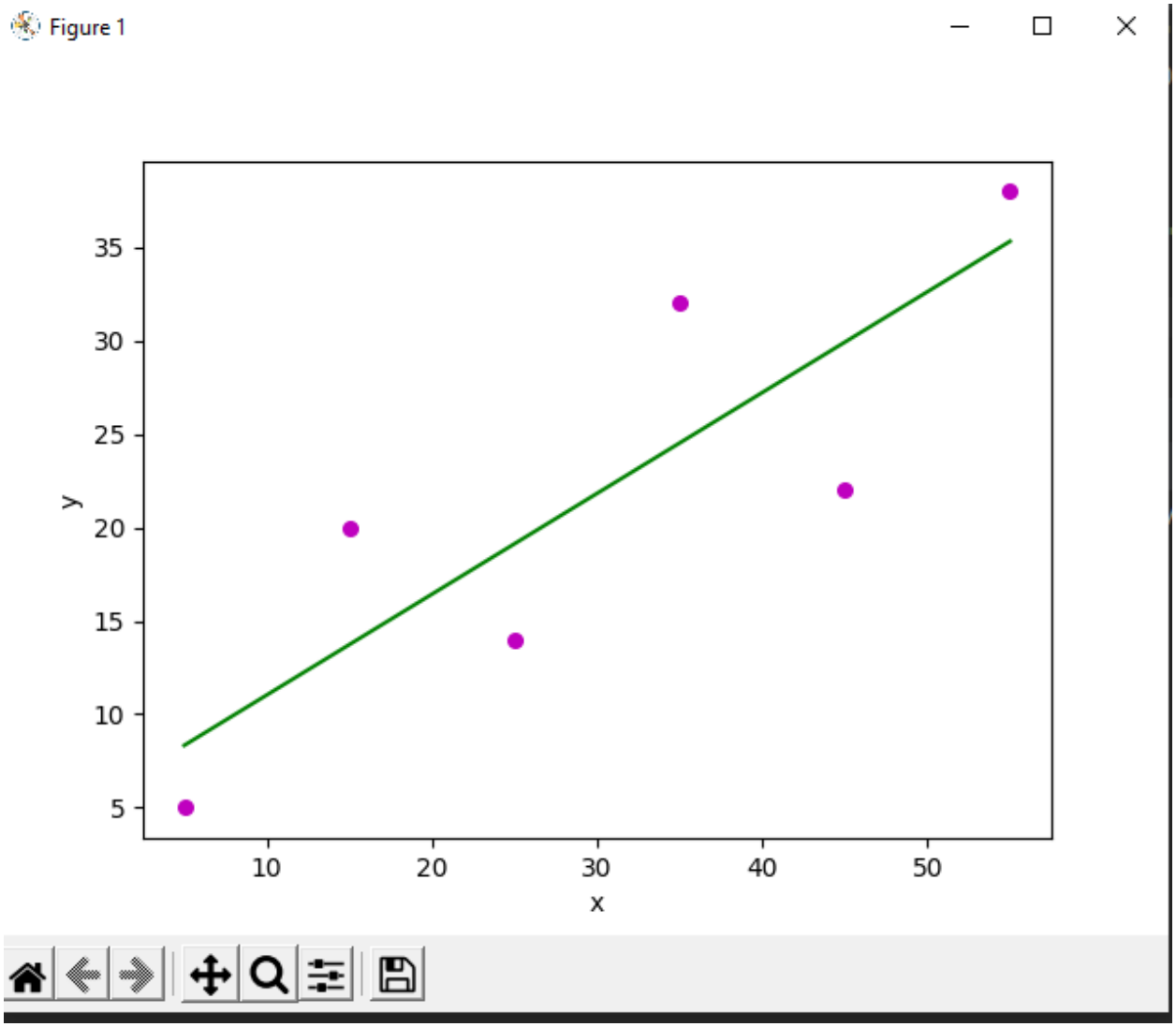
Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

PROGRAM

```
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('coefficient of determination : ', r_sq)
print('intercept : ', model.intercept_)
print('slope : ', model.coef_)
y_pred=model.predict(x)

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\PycharmProjects\file\venv\Scripts\python.exe "C:/Users/ajcemca/Pycharm
[[ 5]
 [15]
 [25]
 [35]
 [45]
 [55]]
[ 5 20 14 32 22 38]
coefficient of determination :  0.7158756137479542
intercept :  5.633333333333329
slope :  [0.54]
```



AIM:3

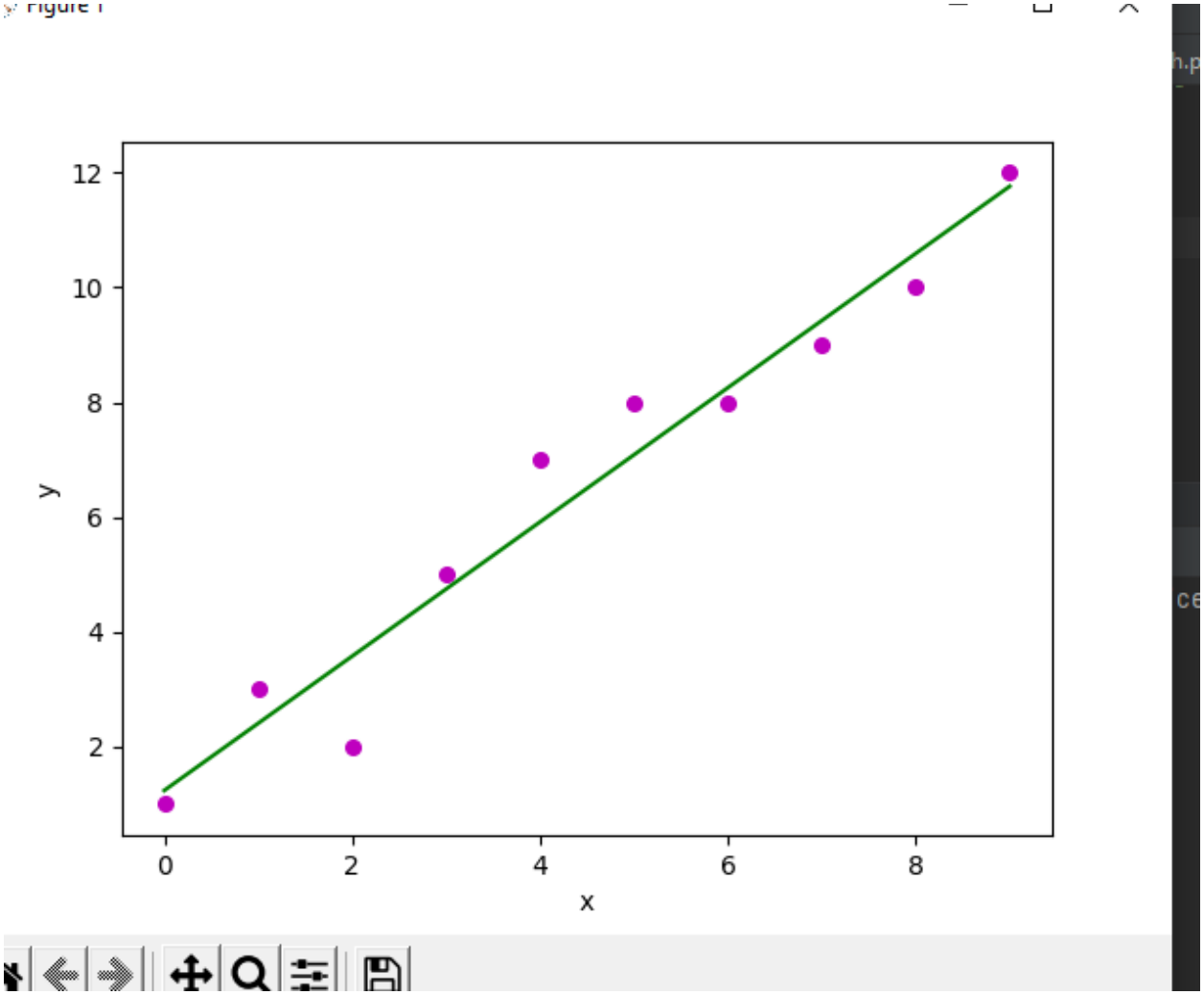
Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance(without using inbuilt function).

PROGRAM

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

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

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

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

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

  return (b_0, b_1)

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

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

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

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

```
  # function to show plot
  plt.show()

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

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




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

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

OUTPUT


12
10
8
6
4
2
y
0
2
4
6
8
x