**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

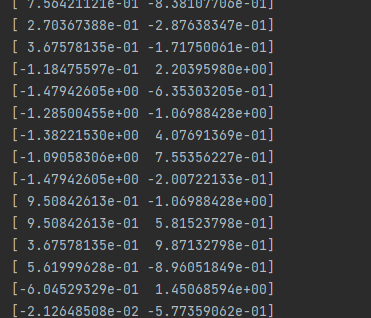
y\_pred = classifier.predict(X\_test)

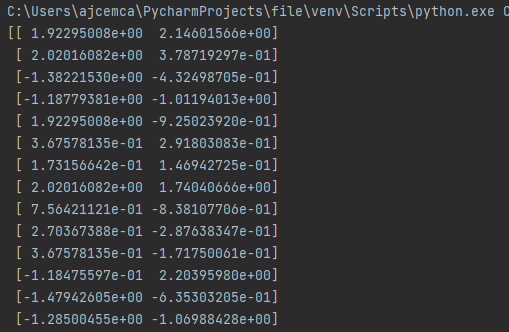
print(y\_pred)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix, acc

OUTPUT





**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)

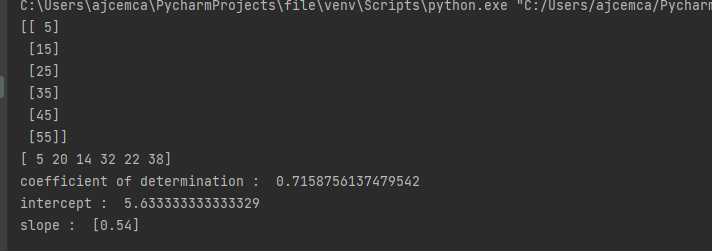
plt.plot(x, y\_pred, color="g")

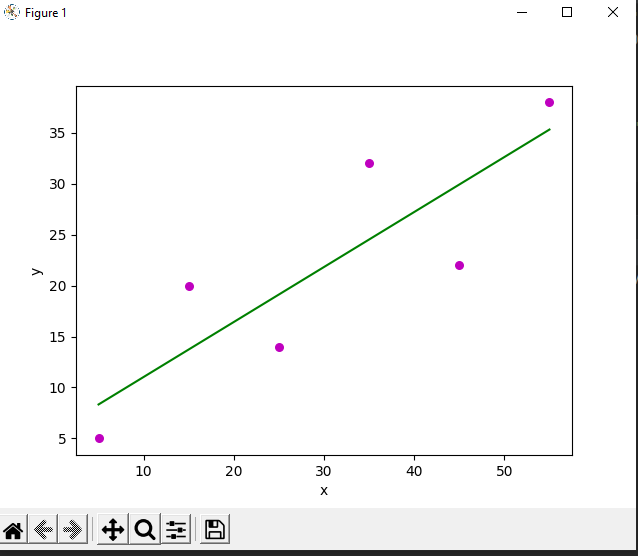
plt.xlabel('x')

plt.ylabel('y')

plt.show()

**OUTPUT**





**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**

