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

import numpy as np

import matplotlib.pyplot as plt

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=0.20, random\_state=1)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train =sc.fit\_transform(X\_train)

X\_test = sc.fit\_transform(X\_test)

print(X\_train)

print(X\_test)

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, Y\_train)

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

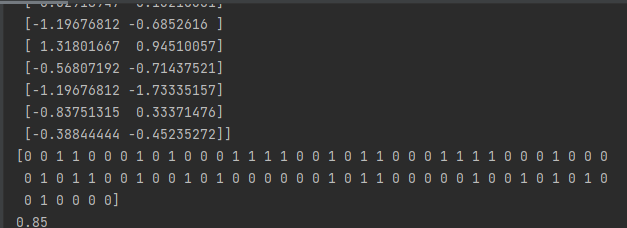
print(Y\_pred)

from sklearn.metrics import confusion\_matrix, accuracy\_score

ac = accuracy\_score(Y\_test, Y\_pred)

print(ac)

**Output 2**

****

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

Linear regression

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

x=np.array([23,43,41,73,71]).reshape((-1,1))

y=np.array([68,24,72,36,56])

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)

print('predicted response:', y\_pred)

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

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

plt.xlabel('x')

plt.ylabel('y')

plt.show()

Linear regression (without using regression function)

import numpy as np

import matplotlib.pyplot as plt

def estimate\_coef(x,y):

n=np.size(x)

m\_x = np.mean(x)

m\_y = np.mean(y)

ss\_xy = np.sum(y \* x) - n \* m\_y \* m\_x

ss\_xx = np.sum(x \* x) - n \* m\_x \* m\_x

b\_1 = ss\_xy / ss\_xx

b\_0 = m\_y -b\_1 \* m\_x

return (b\_0,b\_1)

def plot\_regression\_line(x,y,b):

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

y\_pred = b[0]+b[1]\*x

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

plt.xlabel('x')

plt.ylabel('y')

plt.show()

def main():

# observations / data

x = np.array([9,5,6,7,4,5])

y = np.array([3,2,1,5,7,10])

b = estimate\_coef(x, y)

print("Estimated coefficients:\nb\_0 = {} \

\nb\_1 = {}".format(b[0], b[1]))

plot\_regression\_line(x, y, b)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**output**

