8-12-21:DS & ML\_Lab(2) Jeena Mathew

RMCA:A:42

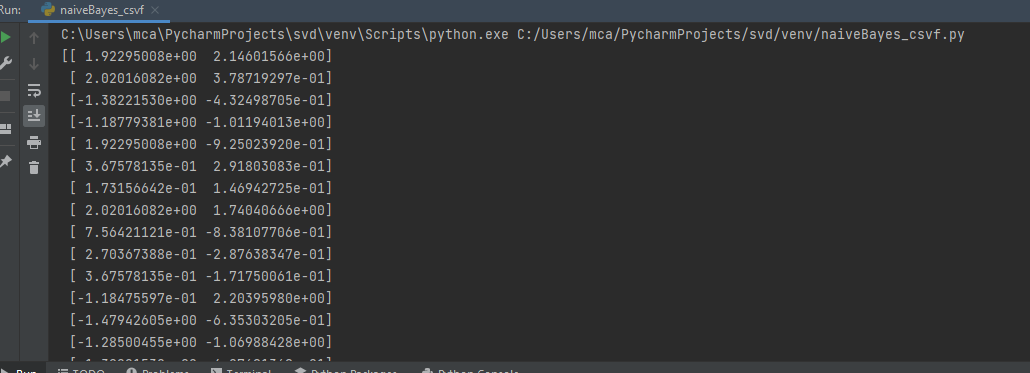
Course\_outcome2\_Co2: Naïve Bayes Algorithm

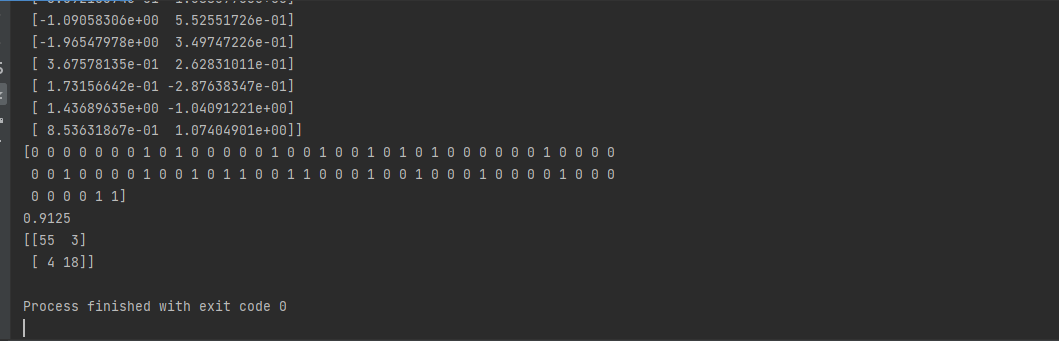
AIM: program to implement naïve bayes algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

Program

#importing the datasets  
dataset = pd.read\_csv('Social\_Network\_Ads.csv')  
x =dataset.iloc[:, [2,3]].values  
y =dataset.iloc[:, -1].values  
  
#splitting  
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\_test)  
  
#training the naive bayes model  
from sklearn.naive\_bayes import GaussianNB  
classifier = GaussianNB()  
classifier.fit(x\_train , y\_train)  
  
#predicting the test result  
y\_read = classifier.predict(x\_test)  
print(y\_read)

Output



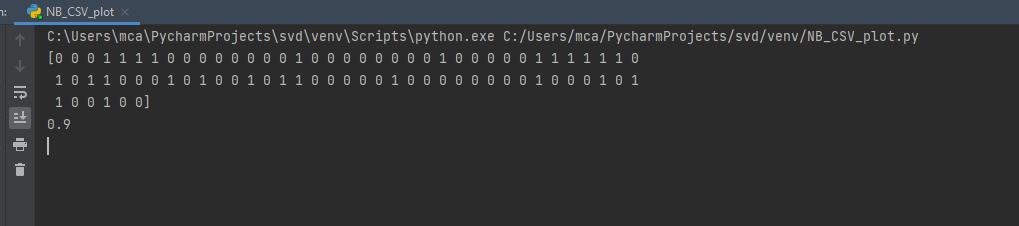


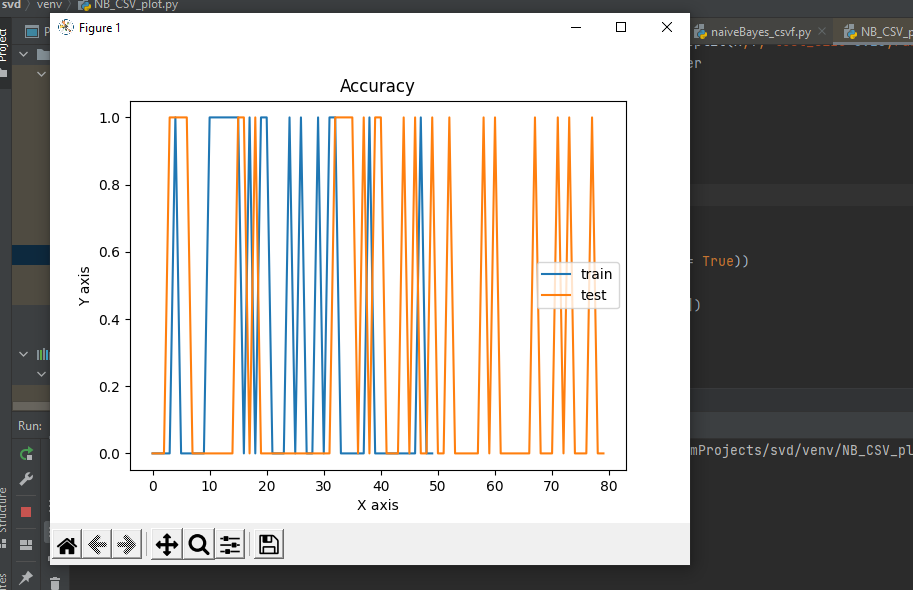
2)plot csv

Program

import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
from sklearn import preprocessing  
from sklearn.model\_selection import train\_test\_split  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.metrics import accuracy\_score  
  
dataset=pd.read\_csv('Social\_Network\_Ads.csv')  
X= dataset.iloc[:,[2,3]].values  
Y= dataset.iloc[:,-1].values  
  
X\_train , X\_test, Y\_train, Y\_test = train\_test\_split(X,Y, test\_size=0.20,random\_state=99)  
from sklearn.preprocessing import StandardScaler  
  
sc = StandardScaler()  
X\_train =sc.fit\_transform(X\_train)  
X\_test = sc.fit\_transform(X\_test)  
clf = GaussianNB()  
clf.fit(X\_train, Y\_train)  
Y\_pred = clf.predict(X\_test)  
print(Y\_pred)  
print(accuracy\_score(Y\_test, Y\_pred, normalize = True))  
  
plt.plot([i for i in range (0,50)],Y\_pred[20:70])  
plt.plot([i for i in range (0,80)],Y\_test)  
  
  
plt.legend(["train","test"])  
plt.xlabel('X axis')  
plt.ylabel('Y axis')  
plt.title('Accuracy')  
plt.show()

Output





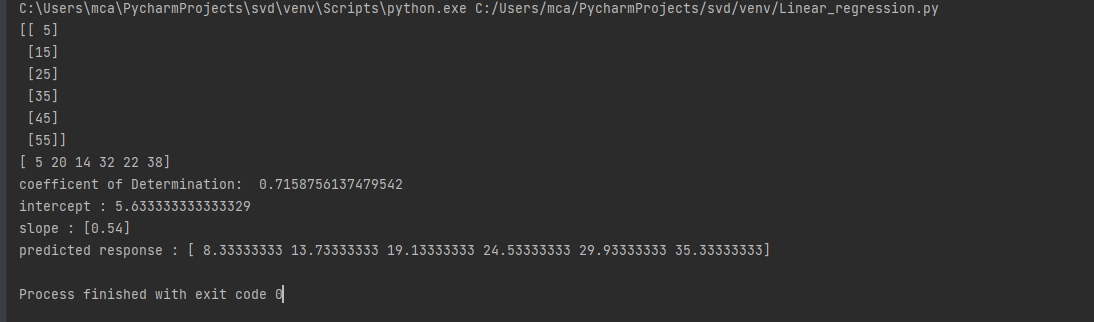
**Program 2)Linear Regression:**

Aim: program to implement linear and multiple regression techniques using any standard dataset available in public domain and evaluating performance.

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('coefficent of Determination: ',r\_sq)  
print('intercept :' , model.intercept\_)  
print('slope :' , model.coef\_)  
y\_pred=model.predict(x)  
print('predicted response :' ,y\_pred)

output

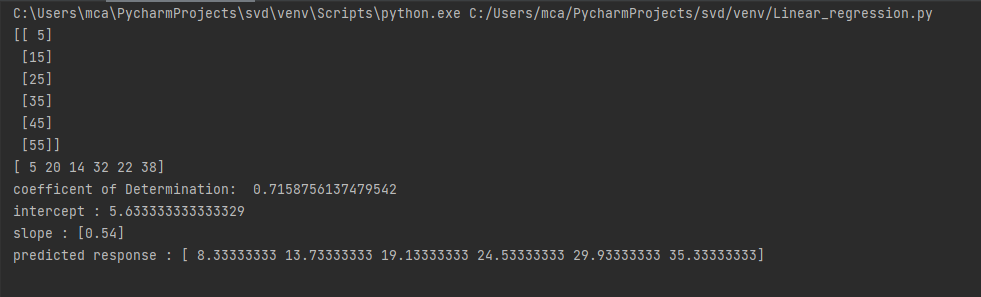


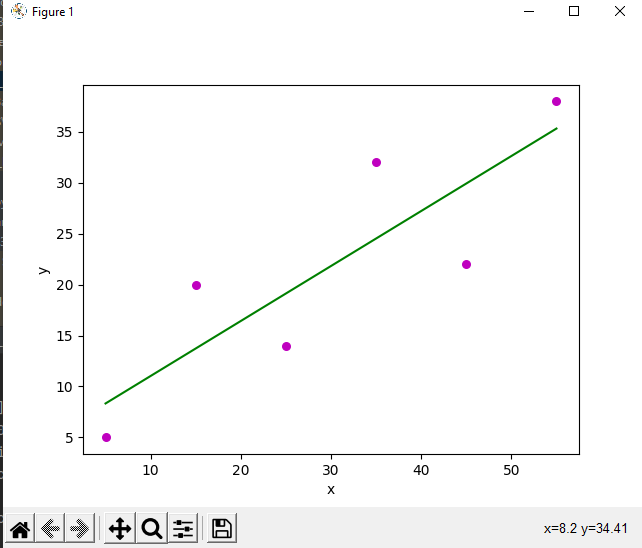
B)plot

Program

import matplotlib.pyplot as plt  
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('coefficent 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",marker = "o" , s=30)  
plt.plot(x, y\_pred,color= "g")  
  
plt.xlabel('x')  
plt.ylabel('y')  
  
plt.show()

Output





c) liner regression Without using in built 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\_x \* m\_x  
 b\_1 = SS\_xy / 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([11,16,6,8,10,12])  
 y = np.array([14,25,18,20,22,24])  
 b = estimate\_coef(x, y)  
 print("estimated coefficient : \nb\_0 ={} \ \nb\_1 = {} ".format(b[0], b[1]))  
 plot\_regression\_line(x, y, b)  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

output

