**LABTASK 8**

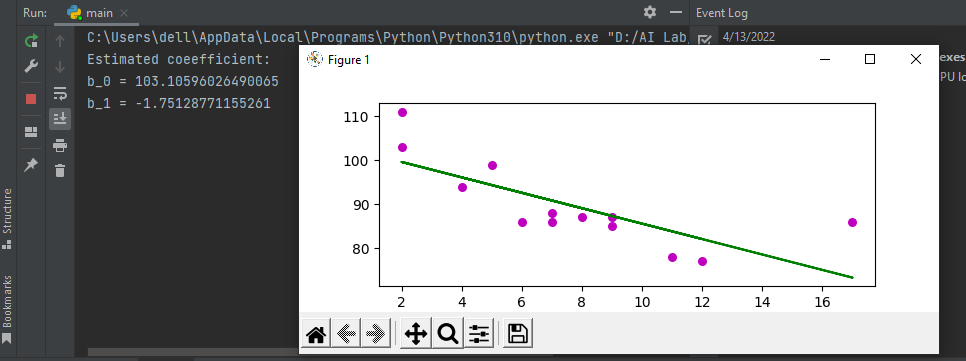
**NAME: SARIM AMIR**

**SID: 63686**

**LINEAR REGRESSION CODE:**

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  
 b1 = SS\_xy / SS\_xx  
 b0 = m\_y - b1 \* m\_x  
 return (b0, b1)  
  
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.xlabel('y')  
 plt.show()  
def main():  
 x = np.array([5, 7, 8, 7, 2, 17, 2, 9, 4, 11, 12, 9, 6])  
 y = np.array([99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86])  
 b = estimate\_coef(x, y)  
 print("Estimated coeefficient:\nb\_0 = {}\nb\_1 = {}".format(b[0], b[1]))  
 plot\_regression\_line(x, y, b)  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

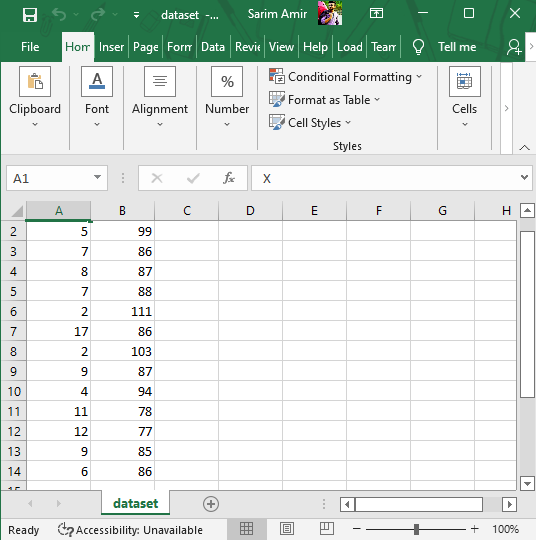
**OUTPUT:**



**LOGISTIC REGRESSION CODE:**

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn import metrics  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
data = pd.read\_csv('dataset.csv')  
X=data[['X']]  
Y=data.Y  
X\_Train, X\_Test, Y\_Train, Y\_Test = train\_test\_split(X, Y, test\_size = 0.25, random\_state = 0)  
logreg = LogisticRegression()  
logreg.fit(X\_Train,Y\_Train)  
y\_pred=logreg.predict(X\_Test)  
  
cnf\_matrix = metrics.confusion\_matrix(Y\_Test, y\_pred)  
cnf\_matrix  
  
class\_names=[0,1] # name of classes  
fig, ax = plt.subplots()  
tick\_marks = np.arange(len(class\_names))  
plt.xticks(tick\_marks, class\_names)  
plt.yticks(tick\_marks, class\_names)  
# create heatmap  
sns.heatmap(pd.DataFrame(cnf\_matrix), annot=True, cmap="YlGnBu" ,fmt='g')  
ax.xaxis.set\_label\_position("top")  
plt.tight\_layout()  
plt.title('Confusion matrix', y=1.1)  
plt.ylabel('Actual label')  
plt.xlabel('Predicted label')  
print("Accuracy:",metrics.accuracy\_score(Y\_Test, y\_pred))  
print("Precision:",metrics.precision\_score(Y\_Test, y\_pred,average='micro'))  
print("Recall:",metrics.recall\_score(Y\_Test, y\_pred,average='micro'))  
  
y\_pred\_proba = logreg.predict\_proba(X\_Test)[::,1]  
fpr, tpr, \_ = metrics.roc\_curve(Y\_Test, y\_pred\_proba,pos\_label='your\_label')  
auc = metrics.roc\_auc\_score(Y\_Test, y\_pred\_proba)  
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))  
plt.legend(loc=4)  
plt.show()

**DATASET:**



**OUTPUT:**

