```
pimport matplotlib.pyplot as plt
import numpy as np
from numpy.linalg import inv

Class0 = []
Class0_X0 = []
Class0_X1 = []
Class0_X0_total = 0
Class0_X1_total = 0
Class1 = []
Class1_X0 = []
Class1_X1 = []
Class1_X0_total = 0
Class1_X1_total = 0


m = np.loadtxt("data.csv", delimiter=",")
for index in m:
    if index[2] == 0:
        Class0.append(index[0:2])
        Class0_X0.append(index[0])
        Class0_X1.append(index[1])
        Class0_X0_total = Class0_X0_total + index[0]
        Class0_X1_total = Class0_X1_total + index[1]
    else:
        Class1.append(index[0:2])
        Class1_X0.append(index[0])
        Class1_X1.append(index[1])
        Class1_X0_total = Class1_X0_total + index[0]
        Class1_X1_total = Class1_X1_total + index[1]
for pick1 in Class0:
    plt.plot(pick1[0],pick1[1], 'o', color = 'blue')
for pick2 in Class1:
    plt.plot(pick2[0],pick2[1], 'x', color = 'red')

# finish drawing the data and the two classes are linearly separable

MU_X0_Class0 = Class0_X0_total / len(Class0)
MU_X1_Class0 = Class0_X1_total / len(Class0)
MU_X0_Class1 = Class1_X0_total / len(Class1)
MU_X1_Class1 = Class1_X1_total / len(Class1)
```

```
MU_CLASS0 = np.array([MU_X0_Class0,MU_X1_Class0])
MU_CLASS1 = np.array([MU_X0_Class1,MU_X1_Class1])



matrix = [[0,0],[0,0]]
for index in m:
    temp = [[0, 0],[0, 0]]
    if index[2] == 0:
        x_val = float(index[0] - MU_CLASS0[0])
        y_val = float(index[1] - MU_CLASS0[1])
        temp[0][0] = float(temp[0][0] + x_val * x_val)
        temp[1][1] = float(temp[1][1] + y_val * y_val)
        temp[0][1] = float(temp[0][1] + x_val * y_val)
        temp[1][0] = float(temp[0][1])
    elif index[2] == 1:
        x_val = float(index[0] - MU_CLASS1[0])
        y_val = float(index[1] - MU_CLASS1[1])
        temp[0][0] = float(temp[0][0] + x_val * x_val)
        temp[1][1] = float(temp[1][1] + y_val * y_val)
        temp[0][1] = float(temp[0][1] + x_val * y_val)
        temp[1][0] = float(temp[0][1])
    matrix[0][0] = float(matrix[0][0] + temp[0][0])
    matrix[1][1] = float(matrix[1][1] + temp[1][1])
    matrix[0][1] = float(matrix[0][1] + temp[0][1])
    matrix[1][0] = float(matrix[0][1])
matrix[0][0] = float(matrix[0][0]/len(m))
matrix[1][1] = float(matrix[1][1]/len(m))
matrix[0][1] = float(matrix[0][1]/len(m))
matrix[1][0] = float(matrix[0][1])
SIGMA = np.array(matrix)


print("P(Y=0) is ",      len(Class0)/len(m))
print("Sigma value is ", SIGMA)
print("MU in Class0 : ", MU_CLASS0)
print("MU in Class1 : ", MU_CLASS1)
# sigma, MU_CLASS0 and MU_CLASS1 : ndarray

W_T =   -np.dot(np.transpose(MU_CLASS1-MU_CLASS0), inv(SIGMA))
print("W_T value is ",W_T)
```

4-c

```
b = -(0.5)*(np.dot(np.dot(np.transpose(MU_CLASS0), inv(SIGMA)), MU_CLASS0) -
np.dot(np.dot(MU_CLASS1.T, inv(SIGMA)), MU_CLASS1)) + np.log(len(Class0)/len(Class1))
print("The value of b ", b)

axes = plt.gca()
axes.set_xlim([-1,10])
axes.set_ylim([-7,2.5])

plt.plot([0,8], [ -b/W_T[1], -(W_T[0]/W_T[1])*8 - b/W_T[1] ])

plt.show()
```

****

```
P(Y=0) is          0.485
Sigma value is  [[1.11872866 0.45204323]
                 [0.45204323 0.71371048]]
MU in Class0 :  [ 1.93477872 -2.97498216]
MU in Class1 :  [ 5.85647767 -1.11750415]
W_T value is    [-3.2978957   -0.51377501]
The value of b  11.736048868281369
```

****