Short Assignment 2

This is an individual assignment.

Due: Tuesday, October 4 @ 11:59pm

Crab Dataset Description

The Crab Data Set has 200 samples and 7 features (Frontal Lip, Rear Width, Length, Width, Depth, Male and Female), describing 5 morphological measurements on 50 crabs each of two color forms and both sexes, of the species *Leptograpsus* variegatus collected at Fremantle, W. Australia.

- Dataset Source: Campbell, N.A. and Mahon, R.J. (1974) A multivariate study of variation in two species of rock crab of genus *Leptograpsus*. *Australian Journal of Zoology* 22, 417–425.

The data set is saved in the file "crab.txt": the firt column corresponds to the class label (crab species) and the other 7 columns correspond to the features.

Use the first 140 samples as your training set and the last 60 samples as your test set.

In [6]:
```
import pandas as pd
import numpy as np
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from numpy.linalg import linalg

data = pd.read_csv("crab.txt", delimiter="\t").to_numpy()
#print(data.shape) #200,8
train_data=data[0:140,:] #train data of shape 140,8
test_data=data[-60:,:] #test data of shape 60,8


#print(x_train)
```

Problem Set

Answer the following questions:

1. Implement the Naive Bayes classifier, under the assumption that your data likelihood model $p(x|C_j)$ is a multivariate Gaussian and the prior probabilities $p(C_j)$ are dictated by the number of samples $n_j \in \mathbb{R}$ that you have for each class. Build your own code to implement the classifier.

2. Did you encounter any problems when implementing the probabilistic generative model? What is your solution for the problem? Explain why your solution works. (Note: There is

more than one solution.)

3. Report your classification results in terms of a confusion matrix in both training and test set. (You can use the function `confusion_matrix` from the module `sklearn.metrics` .)

In [2]:
```
#separating data in 2 classes
class0_old=train_data[np.where(train_data[:,0] == 0)]
class1_old=train_data[np.where(train_data[:,0] == 1)]

class0=np.delete(class0_old, [0,-1], 1) #deleting the redundant dependent column to av
class1=np.delete(class1_old, [0,-1], 1) #Answer 2 discribed in more detail below

# Mean
mu0 = np.mean(class0, axis=0)
mu1 = np.mean(class1, axis=0)

# Variances
cov0 = np.cov(class0.T)
cov1 = np.cov(class1.T)
#print('Cov of Class 0: ', cov0)

N1= len(class0)
N2=len(class1)
#print('Singular error cov=', mu1.shape)

# Estimating Prior Probabilities - relative frequency
N = N1+N2
p1 = N1/N
print('Probability of Train Class 1: ',p1)
p2 = N2/N
print('Probability of Train Class 2: ',p2)
```

```
Probability of Train Class 1:  0.5142857142857142
Probability of Train Class 2:  0.4857142857142857
```

Initially running the code for Answer 1 generated a "singular matrix" error. Answerering question 2, I concluded the reason for this error is because the given data has linearly dependent data. Male and Female features convey the same information and are dependent on each other. Mathematically,when you have linearly dependent columns in a matrix, few matrix operations can make one of the columns to have only zero values. Such a matrix would have a determinant equal to 0, which in turn defines a singular matrix.

The solution to this is elimating one of gender columns and thus cancelling that redundancy, like shown above.

In [3]:
```
x_train=np.delete(train_data, [0,-1], 1)

# Probabaility density function for each class
y0 = multivariate_normal.pdf(x_train, mean=mu0, cov=cov0) #P(x|C0) (72,0)
y1 = multivariate_normal.pdf(x_train, mean=mu1, cov=cov1) #P(x|C1) (68,0)

# Posterior distributions: they represent our classification decision
pos1 = y0*p1 / (y0*p1 + y1*p2) # P(C1|x)
pos2 = y1*p2 / (y0*p1 + y1*p2) # P(C2|x)
#print('pos1, pos2:',pos1.shape)
```

```
y_pred_train=[]
for value in range(len(pos1)):
    if pos1[value] > pos2[value]:
        y_pred_train.append(0)
    else:
        y_pred_train.append(1)

print('y_pred for train data:',y_pred_train)
```

```
y_pred for train data: [0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1,
0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1,
0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 0, 1, 0, 1, 0]
```

In [4]:
```
x_test=np.delete(test_data, [0,-1], 1) #deleting species and Female column

# Data Likelihoods
y1_newPoint = multivariate_normal.pdf(x_test, mean=mu0, cov=cov0) #P(x|C1)
y2_newPoint = multivariate_normal.pdf(x_test, mean=mu1, cov=cov1) #P(x|C2)

print('Data likelihoods:')
print('P(x|C1) = ', y1_newPoint)
print('\nP(x|C2) = ', y2_newPoint,'\n')

# Posterior Probabilities
y1_pos = y1_newPoint*p1 / (y1_newPoint*p1 + y2_newPoint*p2) #P(C1|x)
y2_pos =  y2_newPoint*p2 / (y1_newPoint*p1 + y2_newPoint*p2) #P(C2|x)
#print(y1_pos.shape, y2_pos.shape)

print('\n\n\nPosterior probabilities:')
print('P(C1|x) = ', y1_pos)
print('\nP(C2|x) = ', y2_pos,'\n')
y_pred_test=[]
for value in range(len(y1_pos)):
    if y1_pos[value] > y2_pos[value]:
        y_pred_test.append(0)
    else:
        y_pred_test.append(1)

print('\n\ny_pred for test data:',y_pred_test)
```

```
Data likelihoods:
P(x|C1) =  [5.52908646e-03 1.19567639e-02 7.55839312e-14 3.14603377e-13
 1.18435549e-09 8.04200156e-05 6.11910190e-09 1.60327224e-04
 6.34032230e-03 1.36326532e-02 2.17024147e-03 5.35161630e-11
 2.28570347e-04 8.03305011e-11 1.51564563e-04 2.97397912e-04
 7.24011537e-09 1.08433497e-02 8.30968939e-04 1.22027557e-10
 1.06708280e-12 5.04631854e-05 9.40221137e-05 4.40489200e-07
 7.75106489e-07 2.43661916e-09 2.97900474e-04 4.74383061e-05
 9.33591599e-07 1.27856683e-07 4.28482756e-03 3.89366331e-09
 3.68444754e-10 8.14768974e-06 2.38560013e-08 5.80681716e-04
 1.60562908e-06 6.64579218e-03 7.18451019e-03 3.48782583e-11
 5.77443568e-09 1.71501663e-06 3.19907619e-09 5.71188423e-09
 2.05711195e-09 2.07675696e-04 4.75589781e-06 4.61205650e-10
 4.44504585e-03 1.22814772e-08 1.03580878e-02 3.00521041e-04
 5.93096851e-04 1.99332843e-03 1.21554081e-04 1.21824094e-09
 3.35193372e-06 1.49330991e-05 2.49512471e-12 1.85473603e-03]

P(x|C2) =  [1.17171578e-12 1.29075541e-19 2.70739528e-05 8.33614083e-04
 2.77563131e-02 1.23664966e-06 8.01570625e-05 1.11279588e-24
 9.88494834e-14 1.48504448e-11 7.01090097e-22 2.36946789e-05
 3.72928133e-14 6.52369260e-05 1.18552397e-28 7.50536615e-30
 5.76066497e-05 2.86313857e-09 4.35171820e-06 3.07261765e-03
 3.02415911e-03 1.19334985e-16 2.20591764e-09 1.08597460e-03
 2.72097420e-02 1.25137395e-03 5.65696401e-09 3.60324286e-25
 8.58428747e-03 1.42864808e-02 5.07201893e-10 1.28145613e-02
 1.98721043e-06 2.57376218e-03 1.25313309e-03 5.30704089e-25
 4.38615479e-03 1.79457244e-13 5.11205441e-18 4.54933882e-05
 1.01899151e-02 3.75707027e-03 4.00788522e-03 2.38446589e-03
 3.55764902e-03 1.10062340e-13 7.76051434e-03 1.55066017e-04
 4.73239419e-08 2.88019999e-03 1.06835488e-09 5.00156206e-12
 8.68666205e-23 9.19316623e-24 8.82496608e-31 3.62521719e-04
 1.07597895e-02 1.18396881e-03 1.27784500e-05 6.69013030e-10]



Posterior probabilities:
P(C1|x) =  [1.00000000e+00 1.00000000e+00 2.95597932e-09 3.99596725e-10
 4.51797541e-08 9.85684814e-01 8.08228897e-05 1.00000000e+00
 1.00000000e+00 9.99999999e-01 1.00000000e+00 2.39142456e-06
 1.00000000e+00 1.30379708e-06 1.00000000e+00 1.00000000e+00
 1.33057287e-04 9.99999751e-01 9.95078362e-01 4.20506740e-08
 3.73608773e-10 1.00000000e+00 9.99977842e-01 4.29291907e-04
 3.01611179e-05 2.06168937e-06 9.99982066e-01 1.00000000e+00
 1.15140009e-04 9.47583829e-06 9.99999888e-01 3.21720028e-07
 1.96275844e-04 3.34069156e-03 2.01565074e-05 1.00000000e+00
 3.87450793e-04 1.00000000e+00 1.00000000e+00 8.11763907e-07
 6.00015272e-07 4.83095225e-04 8.45147526e-07 2.53635924e-06
 6.12234988e-07 9.99999999e-01 6.48461025e-04 3.14919971e-06
 9.99989945e-01 4.51491506e-06 9.99999903e-01 9.99999984e-01
 1.00000000e+00 1.00000000e+00 1.00000000e+00 3.55812498e-06
 3.29740281e-04 1.31786764e-02 2.06746217e-07 9.99999659e-01]

P(C2|x) =  [2.00145261e-10 1.01954575e-17 9.99999997e-01 1.00000000e+00
 9.99999955e-01 1.43151865e-02 9.99919177e-01 6.55518049e-21
 1.47244637e-11 1.02881074e-09 3.05099988e-19 9.99997609e-01
 1.54092562e-10 9.99998696e-01 7.38735695e-25 2.38347381e-26
 9.99866943e-01 2.49376319e-07 4.92163807e-03 9.99999958e-01
 1.00000000e+00 2.23341556e-12 2.21577713e-05 9.99570708e-01
 9.99969839e-01 9.99997938e-01 1.79341521e-05 7.17365980e-21
 9.99884860e-01 9.99990524e-01 1.11795387e-07 9.99999678e-01
 9.99803724e-01 9.96659308e-01 9.99979843e-01 8.63158792e-22
 9.99612549e-01 2.55029637e-11 6.72008426e-16 9.99999188e-01
 9.99999400e-01 9.99516905e-01 9.99999155e-01 9.99997464e-01
 9.99999388e-01 5.00529278e-10 9.99351539e-01 9.99996851e-01
 1.00548760e-05 9.99995485e-01 9.74119692e-08 1.57183584e-08
 1.38325970e-19 4.35574723e-21 6.85677531e-27 9.99996442e-01
 9.99670260e-01 9.86821324e-01 9.99999793e-01 3.40665956e-07]



y_pred for test data: [0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0]
```

In [5]:

```
t_test=test_data[:,0]
t_train=train_data[:,0]
#print(t_train.shape)
#print(t_train.shape,y_pred.shape, y1.shape)
print('Confusion matrix for test set:\n',confusion_matrix(t_test, y_pred_test))
print('\nConfusion matrix for train set:\n',confusion_matrix(t_train, y_pred_train))

print('\nWe see there is no False Positives and False Negatives in both train and test
```

```
Confusion matrix for test set:
 [[28  0]
 [ 0 32]]

Confusion matrix for train set:
 [[72  0]
 [ 0 68]]

We see there is no False Positives and False Negatives in both train and test confusi
on matrix. We also see the train set performs better than the test set, due to the pa
rameters given in above
```

Submit Your Solution

Confirm that you've successfully completed the assignment.

Along with the Notebook, include a PDF of the notebook with your solutions.

`add` and `commit` the final version of your work, and `push` your code to your GitHub repository.

Submit the URL of your GitHub Repository as your assignment submission on Canvas.
