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 [1]:
```
import pandas as pd
data = pd.read_csv("crab.txt", delimiter="\t")

data
```

Out[1]:

	Species	FrontalLip	RearWidth	Length	Width	Depth	Male	Female
0	0	20.6	14.4	42.8	46.5	19.6	1	0
1	1	13.3	11.1	27.8	32.3	11.3	1	0
2	0	16.7	14.3	32.3	37.0	14.7	0	1
3	1	9.8	8.9	20.4	23.9	8.8	0	1
4	0	15.6	14.1	31.0	34.5	13.8	0	1
...	...	...	...	...	...	...	...	...
195	1	12.3	11.0	26.8	31.5	11.4	1	0
196	1	12.0	11.1	25.4	29.2	11.0	0	1
197	1	8.8	7.7	18.1	20.8	7.4	1	0
198	1	16.2	15.2	34.5	40.1	13.9	0	1
199	0	15.6	14.0	31.6	35.3	13.8	0	1

200 rows × 8 columns

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]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import confusion_matrix
import scipy.stats as stats
```

In [3]:
```
data_Train = data.iloc[:140,:]
print(data_Train)
data_Test = data.iloc[140:,:]
#print(data_Test)
```

```
     Species  FrontalLip  RearWidth  Length  Width  Depth  Male  Female
0          0        20.6       14.4    42.8   46.5   19.6     1       0
1          1        13.3       11.1    27.8   32.3   11.3     1       0
2          0        16.7       14.3    32.3   37.0   14.7     0       1
3          1         9.8        8.9    20.4   23.9    8.8     0       1
4          0        15.6       14.1    31.0   34.5   13.8     0       1
..       ...         ...        ...     ...    ...    ...   ...     ...
135        0        20.0       16.7    40.4   45.1   17.7     0       1
136        1        15.7       13.9    33.6   38.5   14.1     0       1
137        0        18.6       13.5    36.9   40.2   17.0     1       0
138        1        14.7       12.5    30.1   34.7   12.5     0       1
139        0        16.1       13.7    31.4   36.1   13.9     0       1

[140 rows x 8 columns]
```

In [4]:
```
#Estimate parameters
#Means
mu1 = np.mean(np.array(data_Train[(data_Train['Species']==1)])[:,1:],axis = 0)
print("Mean for crab : ",mu1)
mu2 = np.mean(np.array(data_Train[(data_Train['Species']==0)])[:,1:],axis = 0)
print("Mean for not a crab: ",mu2)

#Covariance
cov1 = np.cov((np.array(data_Train[(data_Train['Species']==1)])[:,1:]).T)
print("Cov for crab: ",cov1)
cov2 = np.cov((np.array(data_Train[(data_Train['Species']==0)])[:,1:]).T)
print("Cov for not a crab: ",cov2)
```

```
### Estimate Prior Probability

N1 = len(np.array(data_Train[(data_Train['Species']==1)]))
print(N1)
N2 = len(np.array(data_Train[(data_Train['Species']==0)]))
print(N2)

N = N1+N2
p1 = N1/N
print("Probability of crab: ",p1)
p2 = N2/N
print("Probability of not a crab:",p2)
```

```
Mean for crab :  [14.03529412 11.84705882 30.08676471 34.73382353 12.56764706  0.5147
0588
  0.48529412]
Mean for not a crab:  [17.10555556 13.62083333 34.1375     38.10277778 15.46527778
0.51388889
  0.48611111]
Cov for crab:  [[ 9.21903424  5.97159789 21.19107112 24.21281826  9.46712906  0.56365
233
  -0.56365233]
 [ 5.97159789  4.57118525 13.72510975 15.78868306  6.24841089  0.06944688
  -0.06944688]
 [21.19107112 13.72510975 49.21638938 56.12493196 21.88956541  1.40094381
  -1.40094381]
 [24.21281826 15.78868306 56.12493196 64.1691374  25.01991659  1.52561457
  -1.52561457]
 [ 9.46712906  6.24841089 21.88956541 25.01991659  9.89117647  0.55122915
  -0.55122915]
 [ 0.56365233  0.06944688  1.40094381  1.52561457  0.55122915  0.25351185
  -0.25351185]
 [-0.56365233 -0.06944688 -1.40094381 -1.52561457 -0.55122915 -0.25351185
   0.25351185]]
Cov for not a crab:  [[ 9.29827856  7.23185446 18.83640845 21.15223787  8.68850548 -
0.43106416
   0.43106416]
 [ 7.23185446  6.85913732 14.3518838  16.57768779  6.55777582 -0.79254695
   0.79254695]
 [18.83640845 14.3518838  38.99167254 43.51876761 18.02005282 -0.63644366
   0.63644366]
 [21.15223787 16.57768779 43.51876761 48.98168232 20.07671753 -0.92539124
   0.92539124]
 [ 8.68850548  6.55777582 18.02005282 20.07671753  8.43948161 -0.26218701
   0.26218701]
 [-0.43106416 -0.79254695 -0.63644366 -0.92539124 -0.26218701  0.25332551
  -0.25332551]
 [ 0.43106416  0.79254695  0.63644366  0.92539124  0.26218701 -0.25332551
   0.25332551]]
68
72
Probability of crab:  0.4857142857142857
Probability of not a crab: 0.5142857142857142
```

In [5]:
```
new_cov1 = cov1 + np.eye(cov1.shape[0])*1e-4

new_cov2 = cov2 + np.eye(cov2.shape[0])*1e-4
```

In [6]:
```
x = [15.6,14.0,31.6,35.8,13.8,0,1]
```

```
# Data Likelihoods
y1_newPoint = stats.multivariate_normal.pdf(x, mean=mu1, cov=new_cov1) #np.multiply(np
y2_newPoint = stats.multivariate_normal.pdf(x, mean=mu2, cov=new_cov2) #np.multiply(np

print('Data likelihoods:')
print('P(x|C1) = ', y1_newPoint)
print('P(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('Posterior probabilities:')
print('P(C1|x) = ', y1_pos)
print('P(C2|x) = ', y2_pos,'\n')

if y1_pos > y2_pos:
    print('x = ',x,' belongs to class 1')
else:
    print('x = ',x,' belongs to class 0')
```

```
Data likelihoods:
P(x|C1) =  8.677474513855524e-06
P(x|C2) =  0.08149039023740824

Posterior probabilities:
P(C1|x) =  0.0001005587094925225
P(C2|x) =  0.9998994412905075

x =  [15.6, 14.0, 31.6, 35.8, 13.8, 0, 1]  belongs to class 0
```

Yes, I faced a problem while trying to load cov matrix into the stats.multivariate_normal.pdf for data likelihood of the new value,it gave an LinAlgErroe : singular matrix.To solve this problem I used to approached loaded the data diagonally by adding covmatrix and identity matrix of same size multplied by some constant and the second approach was element wise multiplication of cov matrix with identity matrix of same size doing this all values except the values of sigma^2 (varinace) becomes zero. Both the appraoches worked for me and produced accurate results.

In [7]:

```
def output(x,mu1,mu2,cov1,cov2):

    y = np.zeros(x.shape[0]);

    for i in range((x.shape[0])):

        y1_newPoint = stats.multivariate_normal.pdf(x[i], mean=mu1, cov=cov1) #np.mult
        y2_newPoint = stats.multivariate_normal.pdf(x[i], mean=mu2, cov=cov2) #np.mult

        #print('Data likelihoods:')
        #print('P(x|C1) = ', y1_newPoint)
        #print('P(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('Posterior probabilities:')
```

```
        #print('P(C1|x) = ', y1_pos)
        #print('P(C2|x) = ', y2_pos,'\n')

        if y1_pos > y2_pos:
            #print('x = ',x,' belongs to class 1')
            y[i] = 1;
        else:
            #print('x = ',x,' belongs to class 0')
            y[i] = 0;

    return y
```

In [8]:
```
X_train = np.array(data_Train.iloc[:,1:]);
X_test =  np.array(data_Test.iloc[:,1:]);

#Output for train data
y1 = output(X_train,mu1,mu2,new_cov1,new_cov2)
t1 = np.array(data_Train.iloc[:,:1])

#Output for test data
y2 = output(X_test,mu1,mu2,new_cov1,new_cov2)
t2 = np.array(data_Test.iloc[:,:1])

# Confusion Matrix for Train and Test data
CM_Train = confusion_matrix(y1,t1)
print(CM_Train)

CM_Test = confusion_matrix(y2,t2)
print(CM_Test)
```

```
[[72  0]
 [ 0 68]]
[[28  0]
 [ 0 32]]
```

In [ ]:
```
```

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.
