Q: a

Aim:

For each of the two synthetic datasets (i) train the classifier, plot the (training-set) data points, the resulting class means, decision boundaries, and decision; also run the trained classifier to classify the data points from their inputs; give the classification error rate on the training set and error rate on the test set.

Program Code: Attached in appendix (Program 1)

OUTPUT:

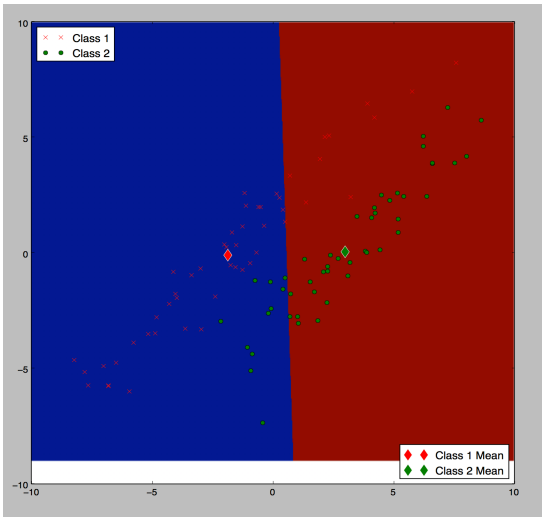

Fig 1: Plot for Synthetic1_Train

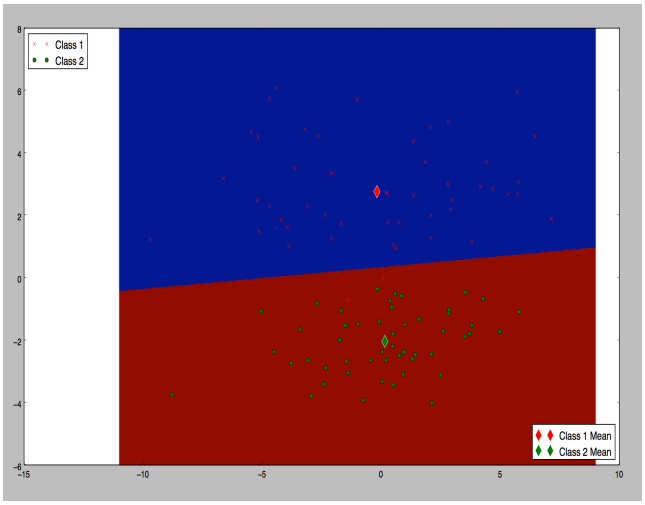

Fig 2: Plot for Synthetic2_Train

```
HariKrishnas-MacBook-Pro:MPR rickerish_nah$ python trial.py
Mean Values for Train 1:
[[-1.8731152  -0.1166418 ]
 [ 2.98095798  0.03548129]]


Mean Values for Train 2:
[[-0.2032685   2.75522592]
 [ 0.13275594 -2.0526066 ]]


Error_Rate of Train 1: 21
Error_Rate of Test 1: 24
Error_Rate of Train 2: 3
Error_Rate of TEST 2: 4
Error_Rate Ratio of Train 1: 21 / 100
Error_Rate Ratio of Test 1: 24 / 100
Error_Rate Ratio of Train 2: 3 / 100
Error_Rate Ratio of Test 2: 4 / 100
HariKrishnas-MacBook-Pro:MPR rickerish_nah$
```

Fig 3: Output for Question a; Mean, classification Error ratio

Q: b

Aim:
To analyze the difference in error rate between the two synthetic datasets

Explanation:
From the above we can draw out the facts that, YES there is a difference in the error rates of the given data points. Synthetic 1 data has an error of about 20% and Synthetic 2 data has an error less than about 5%.
The error rate for Synthetic1 data points is higher than Synthetic2 data points. This is due to the large variance in the distribution of data(Nature of DATA). Even though in Synthetic2 there is a wide spread distribution of data points, we notice that they are more likely linearly separable (More likely linearly separable it is, lesser will be the error rate).
Synthetic 2 data has a better demarcation of classes by its classifier.

Q: c

Aim:
Like Q:a, do the same for the first 2 columns of the 'Wine' data set.

Program Code: Attached in appendix (Program 2)

OUTPUT:

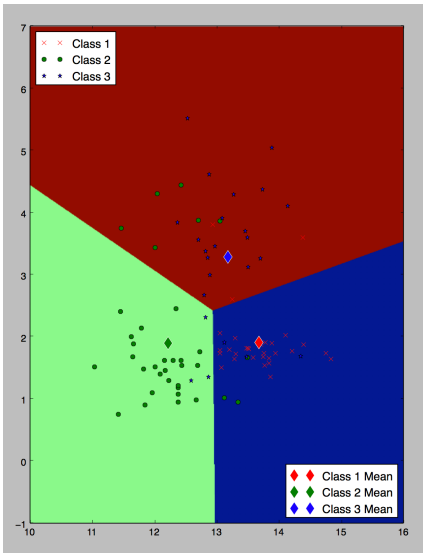

Fig 4: Plot for Wine_Train

```
Mean Values for Train 1:
[[ 13.675        1.904     ]
 [ 12.21457143   1.88885714]
 [ 13.17333333   3.28333333]]


Error_Rate of Train 1: 18
Error_Rate of Test 1: 20
Error_Rate Ratio of Train 1: 18 / 89
Error_Rate Ratio of Test 1: 20 / 89
HariKrishnas-MacBook-Pro:MPR rickerish_nah$
```

Fig 5: Output for Q:c

Q: d

Aim:

Find the 2 features among the 13 that achieve the minimum classification error on the training set. Plot the data points and decision boundaries in 2D for your best performing pair of features, and give its classification error on the training set. Then give its classification error on the test set and description of the method for choosing the best pair of features.

Program Code: Attached in appendix (Program 3)

Explanation:

The method used to train the classifier is **1 vs 1** i.e, each feature is compared with every other feature. Error for each binary pairwise classifier is found and the features that produce the least of the error is chosen as the best feature to train the classifier and check the test data points. In the given question we have 13 features, therefore we have 13C2 (= (13*12)/2) binary classifiers.

Best (Feature i , Feature j) = arg min (ERROR[Feature i , Feature j]) such that i =! J.

OUTPUT:

The best pair of features are Feature 1 and Feature 12. They (in Training DATA) produce the least error, total number of error=7. The corresponding total number of error produced in the Test Data =11.

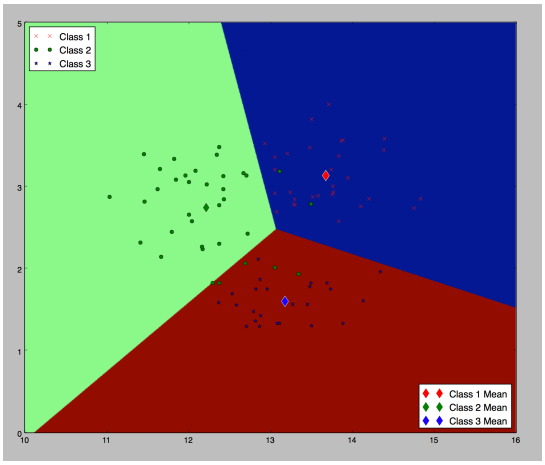

Fig 6: Plot for WINE_TRAIN

```
HariKrishnas-MacBook-Pro:MPR rickerish_nah$ python trial2.py
The Error_Rate for the pair(TRAIN) ( 0 , 1 ) is:  18    The Error_Rate for the pair(TEST) ( 0 , 1 ) is:  20
The Error_Rate for the pair(TRAIN) ( 0 , 2 ) is:  28    The Error_Rate for the pair(TEST) ( 0 , 2 ) is:  25
The Error_Rate for the pair(TRAIN) ( 0 , 3 ) is:  40    The Error_Rate for the pair(TEST) ( 0 , 3 ) is:  36
The Error_Rate for the pair(TRAIN) ( 0 , 4 ) is:  50    The Error_Rate for the pair(TEST) ( 0 , 4 ) is:  40
The Error_Rate for the pair(TRAIN) ( 0 , 5 ) is:  13    The Error_Rate for the pair(TEST) ( 0 , 5 ) is:  14
The Error_Rate for the pair(TRAIN) ( 0 , 6 ) is:  8     The Error_Rate for the pair(TEST) ( 0 , 6 ) is:  10
The Error_Rate for the pair(TRAIN) ( 0 , 7 ) is:  30    The Error_Rate for the pair(TEST) ( 0 , 7 ) is:  25
The Error_Rate for the pair(TRAIN) ( 0 , 8 ) is:  15    The Error_Rate for the pair(TEST) ( 0 , 8 ) is:  22
The Error_Rate for the pair(TRAIN) ( 0 , 9 ) is:  23    The Error_Rate for the pair(TEST) ( 0 , 9 ) is:  20
The Error_Rate for the pair(TRAIN) ( 0 , 10 ) is:  23   The Error_Rate for the pair(TEST) ( 0 , 10 ) is:  24
The Error_Rate for the pair(TRAIN) ( 0 , 11 ) is:  7    The Error_Rate for the pair(TEST) ( 0 , 11 ) is:  11
The Error_Rate for the pair(TRAIN) ( 0 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 0 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 1 , 2 ) is:  35    The Error_Rate for the pair(TEST) ( 1 , 2 ) is:  34
The Error_Rate for the pair(TRAIN) ( 1 , 3 ) is:  35    The Error_Rate for the pair(TEST) ( 1 , 3 ) is:  38
The Error_Rate for the pair(TRAIN) ( 1 , 4 ) is:  51    The Error_Rate for the pair(TEST) ( 1 , 4 ) is:  39
The Error_Rate for the pair(TRAIN) ( 1 , 5 ) is:  26    The Error_Rate for the pair(TEST) ( 1 , 5 ) is:  26
The Error_Rate for the pair(TRAIN) ( 1 , 6 ) is:  18    The Error_Rate for the pair(TEST) ( 1 , 6 ) is:  21
The Error_Rate for the pair(TRAIN) ( 1 , 7 ) is:  29    The Error_Rate for the pair(TEST) ( 1 , 7 ) is:  35
The Error_Rate for the pair(TRAIN) ( 1 , 8 ) is:  36    The Error_Rate for the pair(TEST) ( 1 , 8 ) is:  33
The Error_Rate for the pair(TRAIN) ( 1 , 9 ) is:  22    The Error_Rate for the pair(TEST) ( 1 , 9 ) is:  20
The Error_Rate for the pair(TRAIN) ( 1 , 10 ) is:  34   The Error_Rate for the pair(TEST) ( 1 , 10 ) is:  40
The Error_Rate for the pair(TRAIN) ( 1 , 11 ) is:  38   The Error_Rate for the pair(TEST) ( 1 , 11 ) is:  33
The Error_Rate for the pair(TRAIN) ( 1 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 1 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 2 , 3 ) is:  42    The Error_Rate for the pair(TEST) ( 2 , 3 ) is:  45
The Error_Rate for the pair(TRAIN) ( 2 , 4 ) is:  51    The Error_Rate for the pair(TEST) ( 2 , 4 ) is:  41
The Error_Rate for the pair(TRAIN) ( 2 , 5 ) is:  29    The Error_Rate for the pair(TEST) ( 2 , 5 ) is:  25
The Error_Rate for the pair(TRAIN) ( 2 , 6 ) is:  13    The Error_Rate for the pair(TEST) ( 2 , 6 ) is:  20
The Error_Rate for the pair(TRAIN) ( 2 , 7 ) is:  46    The Error_Rate for the pair(TEST) ( 2 , 7 ) is:  29
The Error_Rate for the pair(TRAIN) ( 2 , 8 ) is:  34    The Error_Rate for the pair(TEST) ( 2 , 8 ) is:  36
The Error_Rate for the pair(TRAIN) ( 2 , 9 ) is:  27    The Error_Rate for the pair(TEST) ( 2 , 9 ) is:  21
The Error_Rate for the pair(TRAIN) ( 2 , 10 ) is:  27   The Error_Rate for the pair(TEST) ( 2 , 10 ) is:  24
The Error_Rate for the pair(TRAIN) ( 2 , 11 ) is:  26   The Error_Rate for the pair(TEST) ( 2 , 11 ) is:  25
The Error_Rate for the pair(TRAIN) ( 2 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 2 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 3 , 4 ) is:  38    The Error_Rate for the pair(TEST) ( 3 , 4 ) is:  34
The Error_Rate for the pair(TRAIN) ( 3 , 5 ) is:  42    The Error_Rate for the pair(TEST) ( 3 , 5 ) is:  42
The Error_Rate for the pair(TRAIN) ( 3 , 6 ) is:  38    The Error_Rate for the pair(TEST) ( 3 , 6 ) is:  37
The Error_Rate for the pair(TRAIN) ( 3 , 7 ) is:  42    The Error_Rate for the pair(TEST) ( 3 , 7 ) is:  45
The Error_Rate for the pair(TRAIN) ( 3 , 8 ) is:  40    The Error_Rate for the pair(TEST) ( 3 , 8 ) is:  43
The Error_Rate for the pair(TRAIN) ( 3 , 9 ) is:  19    The Error_Rate for the pair(TEST) ( 3 , 9 ) is:  26
The Error_Rate for the pair(TRAIN) ( 3 , 10 ) is:  42   The Error_Rate for the pair(TEST) ( 3 , 10 ) is:  45
The Error_Rate for the pair(TRAIN) ( 3 , 11 ) is:  39   The Error_Rate for the pair(TEST) ( 3 , 11 ) is:  40
The Error_Rate for the pair(TRAIN) ( 3 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 3 , 12 ) is:  27
```

Fig 7 Part a: OUTPUT Q:d

```
The Error_Rate for the pair(TRAIN) ( 4 , 5 ) is:  50    The Error_Rate for the pair(TEST) ( 4 , 5 ) is:  40
The Error_Rate for the pair(TRAIN) ( 4 , 6 ) is:  50    The Error_Rate for the pair(TEST) ( 4 , 6 ) is:  37
The Error_Rate for the pair(TRAIN) ( 4 , 7 ) is:  51    The Error_Rate for the pair(TEST) ( 4 , 7 ) is:  41
The Error_Rate for the pair(TRAIN) ( 4 , 8 ) is:  50    The Error_Rate for the pair(TEST) ( 4 , 8 ) is:  40
The Error_Rate for the pair(TRAIN) ( 4 , 9 ) is:  44    The Error_Rate for the pair(TEST) ( 4 , 9 ) is:  39
The Error_Rate for the pair(TRAIN) ( 4 , 10 ) is:  51   The Error_Rate for the pair(TEST) ( 4 , 10 ) is:  41
The Error_Rate for the pair(TRAIN) ( 4 , 11 ) is:  50   The Error_Rate for the pair(TEST) ( 4 , 11 ) is:  39
The Error_Rate for the pair(TRAIN) ( 4 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 4 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 5 , 6 ) is:  20    The Error_Rate for the pair(TEST) ( 5 , 6 ) is:  22
The Error_Rate for the pair(TRAIN) ( 5 , 7 ) is:  31    The Error_Rate for the pair(TEST) ( 5 , 7 ) is:  32
The Error_Rate for the pair(TRAIN) ( 5 , 8 ) is:  36    The Error_Rate for the pair(TEST) ( 5 , 8 ) is:  29
The Error_Rate for the pair(TRAIN) ( 5 , 9 ) is:  25    The Error_Rate for the pair(TEST) ( 5 , 9 ) is:  20
The Error_Rate for the pair(TRAIN) ( 5 , 10 ) is:  29   The Error_Rate for the pair(TEST) ( 5 , 10 ) is:  25
The Error_Rate for the pair(TRAIN) ( 5 , 11 ) is:  22   The Error_Rate for the pair(TEST) ( 5 , 11 ) is:  23
The Error_Rate for the pair(TRAIN) ( 5 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 5 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 6 , 7 ) is:  15    The Error_Rate for the pair(TEST) ( 6 , 7 ) is:  22
The Error_Rate for the pair(TRAIN) ( 6 , 8 ) is:  15    The Error_Rate for the pair(TEST) ( 6 , 8 ) is:  24
The Error_Rate for the pair(TRAIN) ( 6 , 9 ) is:  19    The Error_Rate for the pair(TEST) ( 6 , 9 ) is:  14
The Error_Rate for the pair(TRAIN) ( 6 , 10 ) is:  14   The Error_Rate for the pair(TEST) ( 6 , 10 ) is:  22
The Error_Rate for the pair(TRAIN) ( 6 , 11 ) is:  12   The Error_Rate for the pair(TEST) ( 6 , 11 ) is:  16
The Error_Rate for the pair(TRAIN) ( 6 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 6 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 7 , 8 ) is:  38    The Error_Rate for the pair(TEST) ( 7 , 8 ) is:  42
The Error_Rate for the pair(TRAIN) ( 7 , 9 ) is:  27    The Error_Rate for the pair(TEST) ( 7 , 9 ) is:  21
The Error_Rate for the pair(TRAIN) ( 7 , 10 ) is:  30   The Error_Rate for the pair(TEST) ( 7 , 10 ) is:  31
The Error_Rate for the pair(TRAIN) ( 7 , 11 ) is:  36   The Error_Rate for the pair(TEST) ( 7 , 11 ) is:  29
The Error_Rate for the pair(TRAIN) ( 7 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 7 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 8 , 9 ) is:  27    The Error_Rate for the pair(TEST) ( 8 , 9 ) is:  22
The Error_Rate for the pair(TRAIN) ( 8 , 10 ) is:  33   The Error_Rate for the pair(TEST) ( 8 , 10 ) is:  35
The Error_Rate for the pair(TRAIN) ( 8 , 11 ) is:  31   The Error_Rate for the pair(TEST) ( 8 , 11 ) is:  28
The Error_Rate for the pair(TRAIN) ( 8 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 8 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 9 , 10 ) is:  27   The Error_Rate for the pair(TEST) ( 9 , 10 ) is:  21
The Error_Rate for the pair(TRAIN) ( 9 , 11 ) is:  24   The Error_Rate for the pair(TEST) ( 9 , 11 ) is:  20
The Error_Rate for the pair(TRAIN) ( 9 , 12 ) is:  22   The Error_Rate for the pair(TEST) ( 9 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 10 , 11 ) is:  36  The Error_Rate for the pair(TEST) ( 10 , 11 ) is:  29
The Error_Rate for the pair(TRAIN) ( 10 , 12 ) is:  22  The Error_Rate for the pair(TEST) ( 10 , 12 ) is:  27
The Error_Rate for the pair(TRAIN) ( 11 , 12 ) is:  22  The Error_Rate for the pair(TEST) ( 11 , 12 ) is:  27

The Min Error_Rate for the TRAINING pair ( 0.0 , 11.0 ) is:  7
The Error_Rate for the corresponding TEST pair ( 0.0 , 11.0 ) is:  11
HariKrishnas-MacBook-Pro:MPR rickerish_nah$
```

Fig 7 Part b: OUTPUT Q:d

Q: e

Aim: Check if there is much difference in error rate for different pairs of features

Observation: Yes, from the output generated we can see that there is different error rate for different pairwise binary classifiers.
Example: In the TRAINING data set,
pairs (0,4) has an error of 50
pair (9,11) has an error of 27
Likewise in TEST data set too,
pair (2,5) has an error of 25
pair (4,10) has an error of 41

APPENDIX

Program 1:Qa

```
import numpy as np
import matplotlib
import math
import plotDecBoundaries as pB
given_Data_TRAIN1 = np.genfromtxt("synthetic1_train.csv", delimiter=',')
given_Data_TEST1 = np.genfromtxt("synthetic1_test.csv", delimiter=',')
given_Data_TRAIN2 = np.genfromtxt("synthetic2_train.csv", delimiter=',')
given_Data_TEST2 = np.genfromtxt("synthetic2_test.csv", delimiter=',')
class_Feature_Average_TRAIN1=np.zeros((2,2))
class_Feature_Average_TRAIN2=np.zeros((2,2))
collen_TRAIN1=len(given_Data_TRAIN1)
collen_TRAIN2=len(given_Data_TRAIN2)
collen_TEST1=len(given_Data_TEST1)
collen_TEST2=len(given_Data_TEST2)
collen=0
error_Rate_TRAIN1=0
error_Rate_TEST1=0
dist_1_TRAIN1=0
dist_2_TRAIN1=0
dist_1_TEST1=0
dist_2_TEST1=0
error_Rate_TRAIN2=0
error_Rate_TEST2=0
dist_1_TRAIN2=0
dist_2_TRAIN2=0
dist_1_TEST2=0
dist_2_TEST2=0
train_1_CLASS1_length=0
train_1_CLASS2_length=0
train_2_CLASS1_length=0
train_2_CLASS2_length=0

for i in range(0,collen_TRAIN1):
        if(given_Data_TRAIN1[i][2] ==1):
                train_1_CLASS1_length+=1
        elif(given_Data_TRAIN1[i][2]==2):
                train_1_CLASS2_length+=1
for i in range(0,collen_TRAIN2):
        if(given_Data_TRAIN2[i][2] ==1):
                train_2_CLASS1_length+=1
        elif(given_Data_TRAIN2[i][2]==2):
```

```
                    train_2_CLASS2_length+=1

#print"\tT1:",;print(train_1_CLASS1_length),;print"\tT2:",;print(train_1_CLASS2_length),;print"\
tT1:",;print(train_2_CLASS1_length),;print"\tT2:",;print(train_2_CLASS1_length)

for i in range(0,collen_TRAIN1):
        if(i<train_1_CLASS1_length):

        class_Feature_Average_TRAIN1[0][0]+=given_Data_TRAIN1[i][0]/(train_1_CLASS1_lengt
h)  #MEAN OF FEATURE 1 OF CLASS 1(TRAIN)

        class_Feature_Average_TRAIN1[0][1]+=given_Data_TRAIN1[i][1]/(train_1_CLASS1_lengt
h)  #MEAN OF FEATURE 2 OF CLASS 1(TRAIN)
        if(i>=train_1_CLASS1_length and i<(train_1_CLASS1_length+train_1_CLASS2_length)):

        class_Feature_Average_TRAIN1[1][0]+=given_Data_TRAIN1[i][0]/(train_1_CLASS2_lengt
h)  #MEAN OF FEATURE 1 OF CLASS 2(TRAIN)

        class_Feature_Average_TRAIN1[1][1]+=given_Data_TRAIN1[i][1]/(train_1_CLASS2_lengt
h)  #MEAN OF FEATURE 2 OF CLASS 2(TRAIN)
        if(i<train_2_CLASS1_length):

        class_Feature_Average_TRAIN2[0][0]+=given_Data_TRAIN2[i][0]/(train_2_CLASS1_lengt
h)  #MEAN OF FEATURE 1 OF CLASS 1(TRAIN)

        class_Feature_Average_TRAIN2[0][1]+=given_Data_TRAIN2[i][1]/(train_2_CLASS1_lengt
h)  #MEAN OF FEATURE 2 OF CLASS 1(TRAIN)
        if(i>=train_2_CLASS1_length and i<(train_2_CLASS1_length+train_2_CLASS2_length)):

        class_Feature_Average_TRAIN2[1][0]+=given_Data_TRAIN2[i][0]/(train_2_CLASS2_lengt
h)  #MEAN OF FEATURE 1 OF CLASS 2(TRAIN)

        class_Feature_Average_TRAIN2[1][1]+=given_Data_TRAIN2[i][1]/(train_2_CLASS2_lengt
h)  #MEAN OF FEATURE 2 OF CLASS 2(TRAIN)
        i+=1


for i in range(0,collen_TRAIN1):# all lengths are same
        dist_1_TRAIN1=math.sqrt(((given_Data_TRAIN1[i][0]-
class_Feature_Average_TRAIN1[0][0])**2)+((given_Data_TRAIN1[i][1]-
class_Feature_Average_TRAIN1[0][1])**2))
        dist_2_TRAIN1=math.sqrt(((given_Data_TRAIN1[i][0]-
class_Feature_Average_TRAIN1[1][0])**2)+((given_Data_TRAIN1[i][1]-
class_Feature_Average_TRAIN1[1][1])**2))
```

```
        dist_1_TEST1=math.sqrt(((given_Data_TEST1[i][0]-
class_Feature_Average_TRAIN1[0][0])**2)+((given_Data_TEST1[i][1]-
class_Feature_Average_TRAIN1[0][1])**2))
        dist_2_TEST1=math.sqrt(((given_Data_TEST1[i][0]-
class_Feature_Average_TRAIN1[1][0])**2)+((given_Data_TEST1[i][1]-
class_Feature_Average_TRAIN1[1][1])**2))

        dist_1_TRAIN2=math.sqrt(((given_Data_TRAIN2[i][0]-
class_Feature_Average_TRAIN2[0][0])**2)+((given_Data_TRAIN2[i][1]-
class_Feature_Average_TRAIN2[0][1])**2))
        dist_2_TRAIN2=math.sqrt(((given_Data_TRAIN2[i][0]-
class_Feature_Average_TRAIN2[1][0])**2)+((given_Data_TRAIN2[i][1]-
class_Feature_Average_TRAIN2[1][1])**2))

        dist_1_TEST2=math.sqrt(((given_Data_TEST2[i][0]-
class_Feature_Average_TRAIN2[0][0])**2)+((given_Data_TEST2[i][1]-
class_Feature_Average_TRAIN2[0][1])**2))
        dist_2_TEST2=math.sqrt(((given_Data_TEST2[i][0]-
class_Feature_Average_TRAIN2[1][0])**2)+((given_Data_TEST2[i][1]-
class_Feature_Average_TRAIN2[1][1])**2))
        #For Train_1
        if dist_1_TRAIN1<dist_2_TRAIN1:
                if given_Data_TRAIN1[i][2]!=1:
                        error_Rate_TRAIN1+=1
        else :
                if given_Data_TRAIN1[i][2]!=2:
                        error_Rate_TRAIN1+=1

        #For Test_1
        if dist_1_TEST1<dist_2_TEST1:
                if given_Data_TEST1[i][2]!=1:
                        error_Rate_TEST1+=1
        else :
                if given_Data_TEST1[i][2]!=2:
                        error_Rate_TEST1+=1
        #For Train_2
        if dist_1_TRAIN2<dist_2_TRAIN2:
                if given_Data_TRAIN2[i][2]!=1:
                        error_Rate_TRAIN2+=1
        else :
                if given_Data_TRAIN2[i][2]!=2:
                        error_Rate_TRAIN2+=1
```

```
	#For Test_2
	if dist_1_TEST2<dist_2_TEST2:
		if given_Data_TEST2[i][2]!=1:
			error_Rate_TEST2+=1
	else :
		if given_Data_TEST2[i][2]!=2:
			error_Rate_TEST2+=1
	#print"i:",;print(i),;print"\tDistance 1:",;print(dist_1_TEST2),;print"\t
Distance_2:",;print(dist_2_TEST2),;print"\t error_Rate_TEST2:",;print(error_Rate_TEST2)
	#print"i:",;print(i),;print"\tDistance 1:",;print(dist_1_TRAIN2),;print"\t
Distance_2:",;print(dist_2_TRAIN2),;print"\t error_Rate_TEST2:",;print(error_Rate_TRAIN2)
	i+=1


print"Mean Values for Train 1:"
print(class_Feature_Average_TRAIN1)
print"\n\nMean Values for Train 2:"
print(class_Feature_Average_TRAIN2)
print"\n\nError_Rate of Train 1:",;print(error_Rate_TRAIN1)
print"Error_Rate of Test 1:",;print(error_Rate_TEST1)
print"Error_Rate of Train 2:",;print(error_Rate_TRAIN2)
print"Error_Rate of TEST 2:",;print(error_Rate_TEST2)
print"Error_Rate Ratio of Train 1:",;print(error_Rate_TRAIN1),;print"/",;print(collen_TRAIN1)
print"Error_Rate Ratio of Test 1:",;print(error_Rate_TEST1),;print"/",;print(collen_TEST1)
print"Error_Rate Ratio of Train 2:",;print(error_Rate_TRAIN2),;print"/",;print(collen_TRAIN2)
print"Error_Rate Ratio of Test 2:",;print(error_Rate_TEST2),;print"/",;print(collen_TEST2)

#pB.plotDecBoundary(given_Data_TRAIN1[:,0:2],given_Data_TRAIN1[:,2],class_Feature_Averag
e_TRAIN1)
	#(Up to plat Training 1 OR Down to plot Training 2)

pB.plotDecBoundary(given_Data_TRAIN2[:,0:2],given_Data_TRAIN2[:,2],class_Feature_Average
_TRAIN2)
```

PROGRAM 2:Qc

```
import numpy as np
import matplotlib
import math
import plotDecBoundaries as pB


given_Data_TRAIN1 = np.genfromtxt("wine_train.csv", delimiter=',')
given_Data_TEST1 = np.genfromtxt("wine_test.csv", delimiter=',')

class_Feature_Average_TRAIN1=np.zeros((3,2))


collen_TRAIN1=len(given_Data_TRAIN1)

collen_TEST1=len(given_Data_TEST1)
train_DATA=np.zeros((collen_TRAIN1))
collen=0
error_Rate_TRAIN1=0
error_Rate_TEST1=0
dist_1_TRAIN1=0
dist_2_TRAIN1=0
dist_1_TEST1=0
dist_2_TEST1=0

train_1_CLASS1_length=0
train_1_CLASS2_length=0
train_1_CLASS3_length=0


for i in range(0,collen_TRAIN1):
        if(given_Data_TRAIN1[i][13] ==1):
                train_1_CLASS1_length+=1
        if(given_Data_TRAIN1[i][13]==2):
                train_1_CLASS2_length+=1
        if(given_Data_TRAIN1[i][13] ==3):
                train_1_CLASS3_length+=1


print"\tT1:",;print(train_1_CLASS1_length),;print"\tT2:",;print(train_1_CLASS2_length),;print"\t
T3:",;print(train_1_CLASS3_length)
```

```
for k in range(0,collen_TRAIN1): #Calculating Mean Value
        if k<train_1_CLASS1_length:

        class_Feature_Average_TRAIN1[0][0]+=given_Data_TRAIN1[k][0]/(train_1_CLASS1_leng
th)  #MEAN OF FEATURE 1 OF CLASS 1(TRAIN)

        class_Feature_Average_TRAIN1[0][1]+=given_Data_TRAIN1[k][1]/(train_1_CLASS1_leng
th)  #MEAN OF FEATURE 2 OF CLASS 1(TRAIN)
        if k>=train_1_CLASS1_length and k<(train_1_CLASS2_length+train_1_CLASS1_length):

        class_Feature_Average_TRAIN1[1][0]+=given_Data_TRAIN1[k][0]/(train_1_CLASS2_leng
th)

        class_Feature_Average_TRAIN1[1][1]+=given_Data_TRAIN1[k][1]/(train_1_CLASS2_leng
th)
        if k>=(train_1_CLASS1_length+train_1_CLASS2_length): #and k<collen_TRAIN1:

        class_Feature_Average_TRAIN1[2][0]+=given_Data_TRAIN1[k][0]/(train_1_CLASS3_leng
th)

        class_Feature_Average_TRAIN1[2][1]+=given_Data_TRAIN1[k][1]/(train_1_CLASS3_leng
th)
        k+=1
error_RATE_TRAIN=0
dist_1_TRAIN=0
dist_2_TRAIN=0
dist_3_TRAIN=0
error_RATE_TEST=0
dist_1_TEST=0
dist_2_TEST=0
dist_3_TEST=0
for k in range(0,collen_TRAIN1):
        dist_1_TRAIN=math.sqrt(((given_Data_TRAIN1[k][0]-
class_Feature_Average_TRAIN1[0][0])**2)+((given_Data_TRAIN1[k][1]-
class_Feature_Average_TRAIN1[0][1])**2))
        dist_2_TRAIN=math.sqrt(((given_Data_TRAIN1[k][0]-
class_Feature_Average_TRAIN1[1][0])**2)+((given_Data_TRAIN1[k][1]-
class_Feature_Average_TRAIN1[1][1])**2))
        dist_3_TRAIN=math.sqrt(((given_Data_TRAIN1[k][0]-
class_Feature_Average_TRAIN1[2][0])**2)+((given_Data_TRAIN1[k][1]-
class_Feature_Average_TRAIN1[2][1])**2))
        dist_1_TEST=math.sqrt(((given_Data_TEST1[k][0]-
class_Feature_Average_TRAIN1[0][0])**2)+((given_Data_TEST1[k][1]-
class_Feature_Average_TRAIN1[0][1])**2))
```

```
        dist_2_TEST=math.sqrt(((given_Data_TEST1[k][0]-
class_Feature_Average_TRAIN1[1][0])**2)+((given_Data_TEST1[k][1]-
class_Feature_Average_TRAIN1[1][1])**2))
        dist_3_TEST=math.sqrt(((given_Data_TEST1[k][0]-
class_Feature_Average_TRAIN1[2][0])**2)+((given_Data_TEST1[k][1]-
class_Feature_Average_TRAIN1[2][1])**2))

        #TRAIN
        if dist_1_TRAIN<dist_2_TRAIN and dist_1_TRAIN<dist_3_TRAIN :
                if given_Data_TRAIN1[k][13]!=1:
                        error_RATE_TRAIN+=1
        elif dist_2_TRAIN<dist_1_TRAIN and dist_2_TRAIN<dist_3_TRAIN :
                if given_Data_TRAIN1[k][13]!=2:
                        error_RATE_TRAIN+=1
        elif dist_3_TRAIN<dist_1_TRAIN and dist_3_TRAIN<dist_2_TRAIN :
                if given_Data_TRAIN1[k][13]!=3:
                        error_RATE_TRAIN+=1

        #TEST
        if dist_1_TEST<dist_2_TEST and dist_1_TEST<dist_3_TEST :
                if given_Data_TEST1[k][13]!=1:
                        error_RATE_TEST+=1
        elif dist_2_TEST<dist_1_TEST and dist_2_TEST<dist_3_TEST :
                if given_Data_TEST1[k][13]!=2:
                        error_RATE_TEST+=1
        elif dist_3_TEST<dist_1_TEST and dist_3_TEST<dist_2_TEST :
                if given_Data_TEST1[k][13]!=3:
                        error_RATE_TEST+=1
        k+=1
'''for k in range(0,collen_TRAIN1):
        train_DATA[k][0]=given_Data_TRAIN1[k][0]
        train_DATA[k][1]=given_Data_TRAIN1[k][1]'''


print"Mean Values for Train 1:"
print(class_Feature_Average_TRAIN1)

print"\n\nError_Rate of Train 1:",;print(error_RATE_TRAIN)
print"Error_Rate of Test 1:",;print(error_RATE_TEST)

print"Error_Rate Ratio of Train 1:",;print(error_RATE_TRAIN),;print"/",;print(collen_TRAIN1)
print"Error_Rate Ratio of Test 1:",;print(error_RATE_TEST),;print"/",;print(collen_TEST1)
```

pB.plotDecBoundary(given_Data_TRAIN1[:,0:2],given_Data_TRAIN1[:,13],class_Feature_Average_TRAIN1)

PROGRAM 3:Qd

```
import numpy as np
import matplotlib
import math
import plotDecBoundaries as pB


given_Data_TRAIN1 = np.genfromtxt("wine_train.csv", delimiter=',')
given_Data_TEST1 = np.genfromtxt("wine_test.csv", delimiter=',')
collen_TRAIN1=len(given_Data_TRAIN1)
collen_TEST1=len(given_Data_TEST1)
train_DATA=np.zeros((collen_TRAIN1,2))
label_TRAIN=np.zeros(collen_TRAIN1)
avg=np.zeros((3,2))
min_err_TRAIN=100
test_err=0
memory_TRAIN=np.zeros(2)
col_length_1=0
col_length_2=0
col_length_3=0


for i in range(0,collen_TRAIN1):
        if(given_Data_TRAIN1[i][13] ==1):
                col_length_1+=1
        elif(given_Data_TRAIN1[i][13]==2):
                col_length_2+=1
        elif(given_Data_TRAIN1[i][13]==3):
                col_length_3+=1

#print(col_length_1),;print"\t",;print(col_length_2),;print"\t",;print(col_length_3),;print"\t",;print(col_length_1+col_length_2+col_length_3)

for i in range(0,13):
        for j in range(i+1,13): #nCr comination loop
                #print"i:",;print(i),;print"j:",;print(j)
                class_Feature_Average_TRAIN1=np.zeros((3,2))
                #l1=0
                #l2=0
                #l3=0
                for k in range(0,collen_TRAIN1): #Calculating Mean Value
```

```
                        if k<col_length_1:
                                #l1+=1

        class_Feature_Average_TRAIN1[0][0]+=given_Data_TRAIN1[k][i]/(col_length_1)  #MEAN
OF FEATURE 1 OF CLASS 1(TRAIN)

        class_Feature_Average_TRAIN1[0][1]+=given_Data_TRAIN1[k][j]/(col_length_1)  #MEAN
OF FEATURE 2 OF CLASS 1(TRAIN)
                        if k>=col_length_1 and k<(col_length_2+col_length_1):
                                #print"Hey Class 2"
                                #l2+=1

        class_Feature_Average_TRAIN1[1][0]+=given_Data_TRAIN1[k][i]/(col_length_2)

        class_Feature_Average_TRAIN1[1][1]+=given_Data_TRAIN1[k][j]/(col_length_2)
                        if k>=(col_length_1+col_length_2): #and k<collen_TRAIN1:
                                #print(k)
                                #l3+=1

        class_Feature_Average_TRAIN1[2][0]+=given_Data_TRAIN1[k][i]/(col_length_3)

        class_Feature_Average_TRAIN1[2][1]+=given_Data_TRAIN1[k][j]/(col_length_3)
                        k+=1
                error_RATE_TRAIN=0
                dist_1_TRAIN=0
                dist_2_TRAIN=0
                dist_3_TRAIN=0
                error_RATE_TEST=0
                dist_1_TEST=0
                dist_2_TEST=0
                dist_3_TEST=0
                for k in range(0,collen_TRAIN1):
                        dist_1_TRAIN=math.sqrt(((given_Data_TRAIN1[k][i]-
class_Feature_Average_TRAIN1[0][0])**2)+((given_Data_TRAIN1[k][j]-
class_Feature_Average_TRAIN1[0][1])**2))
                        dist_2_TRAIN=math.sqrt(((given_Data_TRAIN1[k][i]-
class_Feature_Average_TRAIN1[1][0])**2)+((given_Data_TRAIN1[k][j]-
class_Feature_Average_TRAIN1[1][1])**2))
                        dist_3_TRAIN=math.sqrt(((given_Data_TRAIN1[k][i]-
class_Feature_Average_TRAIN1[2][0])**2)+((given_Data_TRAIN1[k][j]-
class_Feature_Average_TRAIN1[2][1])**2))

                        #TRAIN
                        if dist_1_TRAIN<dist_2_TRAIN and dist_1_TRAIN<dist_3_TRAIN :
```

```
                        if given_Data_TRAIN1[k][13]!=1:
                            error_RATE_TRAIN+=1
                    elif dist_2_TRAIN<dist_1_TRAIN and dist_2_TRAIN<dist_3_TRAIN :
                        if given_Data_TRAIN1[k][13]!=2:
                            error_RATE_TRAIN+=1
                    elif dist_3_TRAIN<dist_1_TRAIN and dist_3_TRAIN<dist_2_TRAIN :
                        if given_Data_TRAIN1[k][13]!=3:
                            error_RATE_TRAIN+=1
                    k+=1
                if min_err_TRAIN>=error_RATE_TRAIN:
                    min_err_TRAIN=error_RATE_TRAIN
                    memory_TRAIN[0]=i
                    memory_TRAIN[1]=j
                    avg[0][0]=class_Feature_Average_TRAIN1[0][0]
                    avg[0][1]=class_Feature_Average_TRAIN1[0][1]
                    avg[1][0]=class_Feature_Average_TRAIN1[1][0]
                    avg[1][1]=class_Feature_Average_TRAIN1[1][1]
                    avg[2][0]=class_Feature_Average_TRAIN1[2][0]
                    avg[2][1]=class_Feature_Average_TRAIN1[2][1]
                for k in range(0,collen_TEST1): #Test Error Calculation
                    dist_1_TEST=math.sqrt(((given_Data_TEST1[k][i]-
class_Feature_Average_TRAIN1[0][0])**2)+((given_Data_TEST1[k][j]-
class_Feature_Average_TRAIN1[0][1])**2))
                    dist_2_TEST=math.sqrt(((given_Data_TEST1[k][i]-
class_Feature_Average_TRAIN1[1][0])**2)+((given_Data_TEST1[k][j]-
class_Feature_Average_TRAIN1[1][1])**2))
                    dist_3_TEST=math.sqrt(((given_Data_TEST1[k][i]-
class_Feature_Average_TRAIN1[2][0])**2)+((given_Data_TEST1[k][j]-
class_Feature_Average_TRAIN1[2][1])**2))
                    #TEST
                    if dist_1_TEST<dist_2_TEST and dist_1_TEST<dist_3_TEST :
                        if given_Data_TEST1[k][13]!=1:
                            error_RATE_TEST+=1
                    elif dist_2_TEST<dist_1_TEST and dist_2_TEST<dist_3_TEST :
                        if given_Data_TEST1[k][13]!=2:
                            error_RATE_TEST+=1
                    elif dist_3_TEST<dist_1_TEST and dist_3_TEST<dist_2_TEST :
                        if given_Data_TEST1[k][13]!=3:
                            error_RATE_TEST+=1
                if(i==memory_TRAIN[0]):
                    if(j==memory_TRAIN[1]):
                        test_err=error_RATE_TEST
```

```
                print"The Error_Rate for the pair(TRAIN) (",;print(i),;print",",;print(j),;print") is:
",;print(error_RATE_TRAIN),;print"\tThe Error_Rate for the pair(TEST)
(",;print(i),;print",",;print(j),;print") is: ",;print(error_RATE_TEST)
                #print"The Error_Rate for the pair(TEST) (",;print(i),;print",",;print(j),;print") is:
",;print(error_RATE_TEST)

                j+=1
        i+=1

for k in range(0,collen_TRAIN1):
        train_DATA[k][0]=given_Data_TRAIN1[k][memory_TRAIN[0]]
        train_DATA[k][1]=given_Data_TRAIN1[k][memory_TRAIN[1]]
        label_TRAIN[k]=given_Data_TRAIN1[k][13]
        avg
        label_TRAIN


print"\n\n\nThe Min Error_Rate for the TRAINING pair
(",;print(memory_TRAIN[0]),;print",",;print(memory_TRAIN[1]),;print") is:
",;print(min_err_TRAIN)
print"The Error_Rate for the corresponding TEST pair
(",;print(memory_TRAIN[0]),;print",",;print(memory_TRAIN[1]),;print") is: ",;print(test_err)
pB.plotDecBoundary(train_DATA[:,0:2],label_TRAIN[:],avg[:,:])
```

-------THANK YOU-------