EE – 559 Homework 1

Name – Tamoghna Chattopadhyay

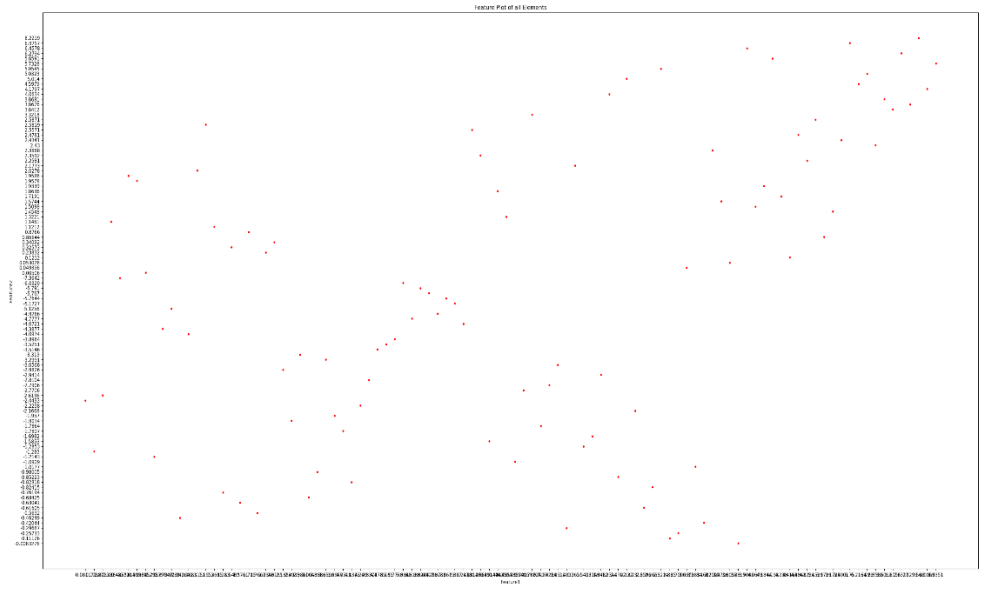
a) Code for training data set-1 : Synthetic1_train:

```
import csv
import numpy as np
import math
import matplotlib.pyplot as plt
import copy
from scipy.spatial.distance import cdist
from plotDecBoundaries import plotDecBoundaries



#Open the csv file and store it in a list
with open("D:\python3_files\python3\synthetic1_train.csv","r") as Feat_Train:
   Feat_Train_Reader = csv.reader(Feat_Train, delimiter=',')
   FeatureList = []
   for row in Feat_Train_Reader:
      if len (row) != 0:
         FeatureList = FeatureList +[row]

Feat_Train.close()

#Make a copy of the training data list with just the training data and no labels
FeatureListDec = copy.deepcopy(FeatureList)
for row in FeatureListDec:
   del row[-1]

#Make an array of the training data
result1 = np.array(FeatureListDec).astype("float")

#Make a list containing the class levels
Class = []
for row in FeatureList:
   Class.append(row[-1])

result2 = np.array(Class).astype("float")

FeatureList1 = []
FeatureList2 = []

#Divide the training data into two separate lists based on their class
for i in FeatureList:
```

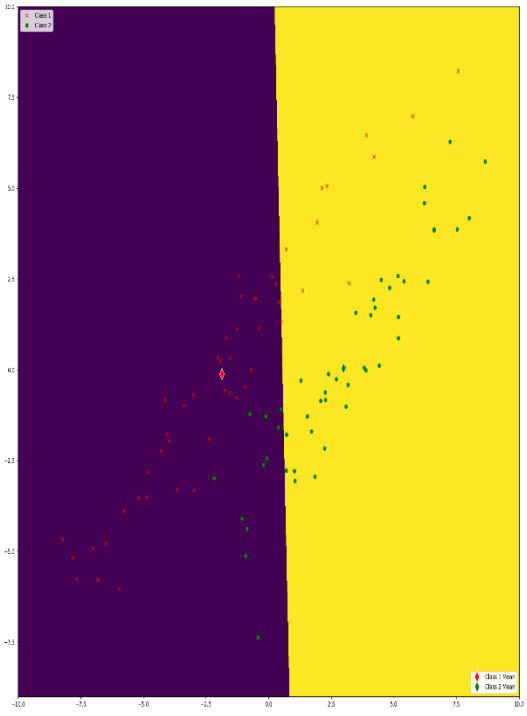
```
    if float(i[-1]) == 1:
        FeatureList1.append(i)
    else:
        FeatureList2.append(i)

Sum1 = 0
Sum2 = 0
Sum3 = 0
Sum4 = 0
count1 = 0
count2 = 0

#Calculate the means
for i in FeatureList1:
    Sum1 = Sum1 + float(i[0])
    Sum2 = Sum2 + float(i[1])
    count1 = count1 + 1

for i in FeatureList2:
    Sum3 = Sum3 + float(i[0])
    Sum4 = Sum4 + float(i[1])
    count2 = count2 + 1

Mean1 = Sum1/count1
Mean2 = Sum2/count1
Mean3 = Sum3/count2
Mean4 = Sum4/count2

Mean = [[Mean1,Mean2],[Mean3,Mean4]]
result3 = np.array(Mean).astype("float")

#Plot the data points
xs = [x[0] for x in FeatureList]
ys = [x[1] for x in FeatureList]
plt.plot(xs,ys,'r.')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.title('Feature Plot of all Elements')
plt.show()

#Train the classifier on training data
for i in FeatureList:
    if math.sqrt((Mean1 - float(i[0]))**2 + (Mean2 - float(i[1]))**2) > math.sqrt((Mean3 -
float(i[0]))**2 + (Mean4 - float(i[1]))**2):
```

```
        i.append('2')
    else:
        i.append('1')

Train_error = 0

#Find the training data error rate
for i in FeatureList:
    if i[-1] != i[-2]:
        Train_error = Train_error + 1

Train_Error_Rate = Train_error/len(FeatureList)

#Open the test data set
with open("D:\python3_files\python3\synthetic1_test.csv","r") as Feat_Test:
    Feat_Test_Reader = csv.reader(Feat_Test, delimiter=',')
    FeatureTestList = []
    for row in Feat_Test_Reader:
        if len (row) != 0:
            FeatureTestList = FeatureTestList +[row]

Feat_Test.close()

#Find the test data error rate
for i in FeatureTestList:
    if math.sqrt((Mean1 - float(i[0]))**2 + (Mean2 - float(i[1]))**2) > math.sqrt((Mean3 -
float(i[0]))**2 + (Mean4 - float(i[1]))**2):
        i.append('2')
    else:
        i.append('1')

Test_error = 0

for i in FeatureTestList:
    if i[-1] != i[-2]:
        Test_error = Test_error + 1

Test_Error_Rate = Test_error/len(FeatureTestList)

#Plot the decision boundary
plotDecBoundaries(result1,result2,result3)
```

Figures:



Data Points in Synthetic_Train 1




Decision Boundary for Synthetic_Train1

## Train_Error_Rate = 0.21
## Test_Error_Rate = 0.24

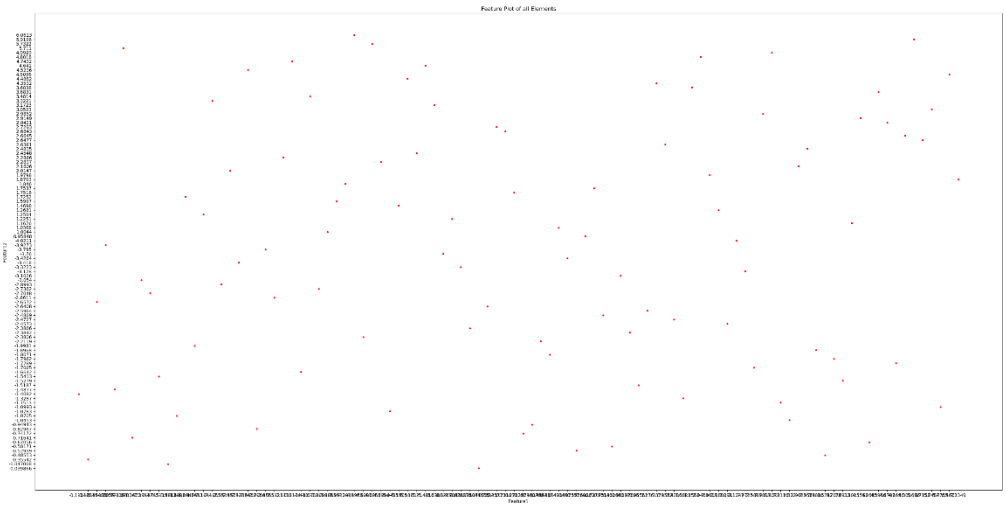
Code for training data set-2 : Synthetic2_train:

```
import csv
import math
import matplotlib.pyplot as plt
import numpy as np
import copy
from scipy.spatial.distance import cdist
from plotDecBoundaries import plotDecBoundaries

#Open the csv file and store it in a list
with open("D:\python3_files\python3\synthetic2_train.csv","r") as Feat_Train:
   Feat_Train_Reader = csv.reader(Feat_Train, delimiter=',')
   FeatureList = []
   for row in Feat_Train_Reader:
      if len (row) != 0:
         FeatureList = FeatureList +[row]

Feat_Train.close()

#Make a copy of the training data list with just the training data and no labels
FeatureListDec = copy.deepcopy(FeatureList)
for row in FeatureListDec:
   del row[-1]

#Make an array of the training data
result1 = np.array(FeatureListDec).astype("float")

#Make a list containing the class levels
Class = []
for row in FeatureList:
   Class.append(row[-1])

result2 = np.array(Class).astype("float")

FeatureList1 = []
FeatureList2 = []

#Divide the training data into two separate lists based on their class
for i in FeatureList:
   if float(i[-1]) == 1:
```

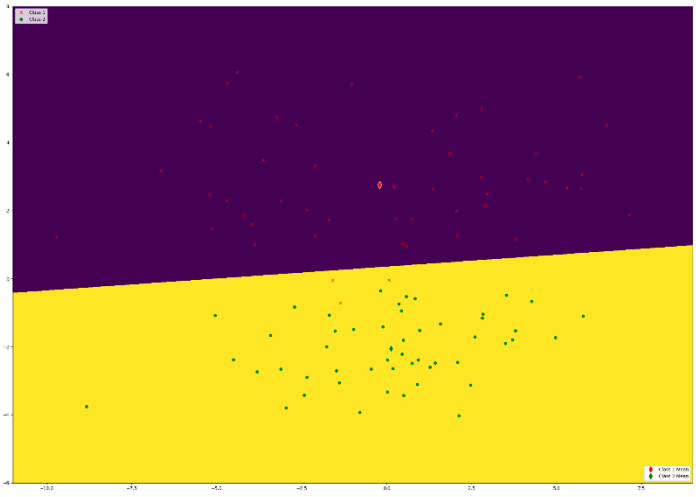
```
        FeatureList1.append(i)
    else:
        FeatureList2.append(i)

Sum1 = 0
Sum2 = 0
Sum3 = 0
Sum4 = 0
count1 = 0
count2 = 0

#Calculate the means
for i in FeatureList1:
    Sum1 = Sum1 + float(i[0])
    Sum2 = Sum2 + float(i[1])
    count1 = count1 + 1

for i in FeatureList2:
    Sum3 = Sum3 + float(i[0])
    Sum4 = Sum4 + float(i[1])
    count2 = count2 + 1

Mean1 = Sum1/count1
Mean2 = Sum2/count1
Mean3 = Sum3/count2
Mean4 = Sum4/count2

Mean = [[Mean1,Mean2],[Mean3,Mean4]]

result3 = np.array(Mean).astype("float")

#Plot the data points
xs = [x[0] for x in FeatureList]
ys = [x[1] for x in FeatureList]
plt.plot(xs,ys,'r.')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.title('Feature Plot of all Elements')
plt.show()

#Train the classifier on training data
for i in FeatureList:
    if math.sqrt((Mean1 - float(i[0]))**2 + (Mean2 - float(i[1]))**2) > math.sqrt((Mean3 -
float(i[0]))**2 + (Mean4 - float(i[1]))**2):
```

```
        i.append('2')
    else:
        i.append('1')

Train_error = 0

#Find the training data error rate
for i in FeatureList:
    if i[-1] != i[-2]:
        Train_error = Train_error + 1

Train_Error_Rate = Train_error/len(FeatureList)

#Open the test data set
with open("D:\python3_files\python3\synthetic2_test.csv","r") as Feat_Test:
    Feat_Test_Reader = csv.reader(Feat_Test, delimiter=',')
    FeatureTestList = []
    for row in Feat_Test_Reader:
        if len (row) != 0:
            FeatureTestList = FeatureTestList +[row]

Feat_Test.close()

#Find the test data error rate
for i in FeatureTestList:
    if math.sqrt((Mean1 - float(i[0]))**2 + (Mean2 - float(i[1]))**2) > math.sqrt((Mean3 -
float(i[0]))**2 + (Mean4 - float(i[1]))**2):
        i.append('2')
    else:
        i.append('1')

Test_error = 0

for i in FeatureTestList:
    if i[-1] != i[-2]:
        Test_error = Test_error + 1

Test_Error_Rate = Test_error/len(FeatureTestList)

#Plot the decision boundary
plotDecBoundaries(result1,result2,result3)
```

Figures:



Data Points in Synthetic_Train 2



Decision Boundary for Synthetic_Train2

**Train_Error_Rate = 0.03**

**Test_Error_Rate = 0.04**

b) **Yes, there is a big difference between the error rates for the two synthetic datasets. As we can see from the plots of the decision boundary for the two synthetic datasets, we can see that the decision boundary for the second dataset is much better classifier than that for the first dataset. This leads to less error rate for the Synthetic 2 dataset.**

c),d),e)

Code:

```
import csv
import math
import matplotlib.pyplot as plt
import copy
from scipy.spatial.distance import cdist
import numpy as np
from plotDecBoundaries import plotDecBoundaries

#Open the csv file and store it in a list
with open("D:\python3_files\python3\wine_train.csv","r") as Feat_Train:
    Feat_Train_Reader = csv.reader(Feat_Train, delimiter=',')
    FeatureList = []
    for row in Feat_Train_Reader:
        if len (row) != 0:
            FeatureList = FeatureList +[row]

Feat_Train.close()

#Make a copy of the training data list with just the training data and no labels
FeatureListDec = copy.deepcopy(FeatureList)
for i in range(0,12):
    for row in FeatureListDec:
        del row[-1]

#Make an array of the training data
result1 = np.array(FeatureListDec).astype("float")

#Make a list containing the class labels
Class = []
for row in FeatureList:
    Class.append(row[-1])

result2 = np.array(Class).astype("float")
```

```
FeatureList1 = []
FeatureList2 = []
FeatureList3 = []

#Divide the training data into three separate lists based on their class
for i in FeatureList:
   if float(i[-1]) == 1:
      FeatureList1.append(i)
   elif float(i[-1]) == 2:
      FeatureList2.append(i)
   else:
      FeatureList3.append(i)

Sum1 = 0
Sum2 = 0
Sum3 = 0
Sum4 = 0
Sum5 = 0
Sum6 = 0
count1 = 0
count2 = 0
count3 = 0

#Calculate the means
for i in FeatureList1:
   Sum1 = Sum1 + float(i[0])
   Sum2 = Sum2 + float(i[1])
   count1 = count1 + 1

for i in FeatureList2:
   Sum3 = Sum3 + float(i[0])
   Sum4 = Sum4 + float(i[1])
   count2 = count2 + 1

for i in FeatureList3:
   Sum5 = Sum5 + float(i[0])
   Sum6 = Sum6 + float(i[1])
   count3 = count3 + 1


Mean1 = Sum1/count1
Mean2 = Sum2/count1
Mean3 = Sum3/count2
```

```
Mean4 = Sum4/count2
Mean5 = Sum5/count3
Mean6 = Sum6/count3

Mean = [[Mean1,Mean2],[Mean3,Mean4], [Mean5,Mean6]]

result3 = np.array(Mean).astype("float")

#Plot the data points
xs1 = [x[0] for x in FeatureList1]
ys1 = [x[1] for x in FeatureList1]
xs2 = [x[0] for x in FeatureList2]
ys2 = [x[1] for x in FeatureList2]
xs3 = [x[0] for x in FeatureList3]
ys3 = [x[1] for x in FeatureList3]

plt.plot(xs1,ys1,'r.',xs2,ys2,'b^',xs3,ys3,'g.')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.title('Feature Plot of all Elements')
plt.show()

DistanceTraining = []

#Train the classifier on training data
for i in FeatureList:
   DistanceTraining.append(math.sqrt((Mean1 - float(i[0]))**2 + (Mean2 - float(i[1]))**2))
   DistanceTraining.append(math.sqrt((Mean3 - float(i[0]))**2 + (Mean4 - float(i[1]))**2))
   DistanceTraining.append(math.sqrt((Mean5 - float(i[0]))**2 + (Mean6 - float(i[1]))**2))
   i.append(DistanceTraining.index(min(DistanceTraining))+1)
   DistanceTraining = []

Train_error = 0

#Find the training data error rate
for i in FeatureList:
   if int(i[-1]) != int(i[-2]):
      Train_error = Train_error + 1

Train_Error_Rate = Train_error/len(FeatureList)

#Open the test data set
with open("D:\python3_files\python3\wine_test.csv","r") as Feat_Test:
   Feat_Test_Reader = csv.reader(Feat_Test, delimiter=',')
```

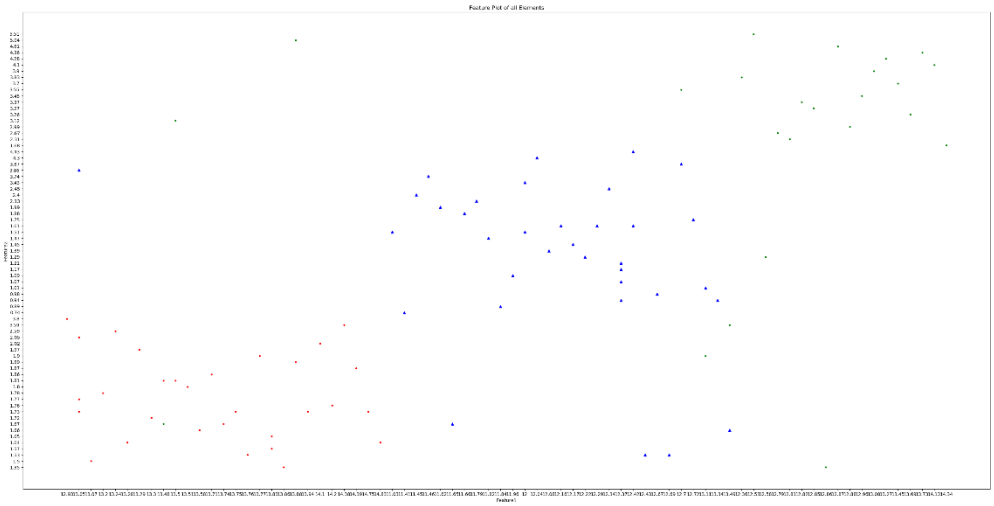
```
    FeatureTestList = []
    for row in Feat_Test_Reader:
        if len (row) != 0:
            FeatureTestList = FeatureTestList +[row]

Feat_Test.close()

DistanceTest = []

#Find the test data error rate
for i in FeatureTestList:
    DistanceTest.append(math.sqrt((Mean1 - float(i[0]))**2 + (Mean2 - float(i[1]))**2))
    DistanceTest.append(math.sqrt((Mean3 - float(i[0]))**2 + (Mean4 - float(i[1]))**2))
    DistanceTest.append(math.sqrt((Mean5 - float(i[0]))**2 + (Mean6 - float(i[1]))**2))
    i.append(DistanceTest.index(min(DistanceTest))+1)
    DistanceTest = []

Test_error = 0

for i in FeatureTestList:
    if int(i[-1]) != int(i[-2]):
        Test_error = Test_error + 1

Test_Error_Rate = Test_error/len(FeatureTestList)

#Plot the decision boundary
plotDecBoundaries(result1,result2,result3)

for row in FeatureList:
    del row[-1]

for row in FeatureTestList:
    del row[-1]

y = len(FeatureList[0])
length = y-1

Sums1 = [0]*13
Sums2 = [0]*13
Sums3 = [0]*13
Means1 = [0]*13
Means2 = [0]*13
Means3 = [0]*13
Distance_Trains = []
```

```
Minimum_Error_Train = 500

Mean = [[Means1],[Means2],[Means3]]

#Find Means for all features
for k in FeatureList1:
   for i in range(0,length):
      Sums1[i] = Sums1[i] + float(k[i])

for k in FeatureList2:
   for i in range(0,length):
      Sums2[i] = Sums2[i] + float(k[i])

for k in FeatureList3:
   for i in range(0,length):
      Sums3[i] = Sums3[i] + float(k[i])

for i in range(0,length):
   Means1[i] = Sums1[i]/count1
   Means2[i] = Sums2[i]/count2
   Means3[i] = Sums3[i]/count3

#Find the best two features in training data set which gives least error and calculate the Training
Error Rate
for i in range(0,y-2):
   for j in range(i+1,y-1):
      error1 = 0
      for k in FeatureList:
         Distance_Trains.append(math.sqrt((Means1[i] - float(k[i]))**2 + (Means1[j] -
float(k[j]))**2))
         Distance_Trains.append(math.sqrt((Means2[i] - float(k[i]))**2 + (Means2[j] -
float(k[j]))**2))
         Distance_Trains.append(math.sqrt((Means3[i] - float(k[i]))**2 + (Means3[j] -
float(k[j]))**2))
         if int(k[-1]) != (Distance_Trains.index(min(Distance_Trains))+1):
            error1 = error1 + 1
         Distance_Trains = []
      if error1 < Minimum_Error_Train:
         Minimum_Error_Train = error1
         features_train = [i,j]

Minimum_Error_Train_Rate = Minimum_Error_Train/len(FeatureList)

Minimum_Error_Test = 500
```

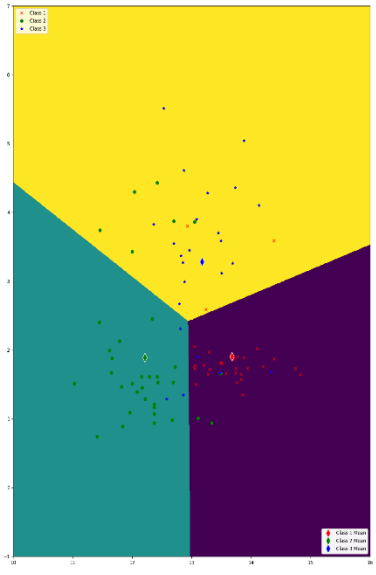
```
Distance_Tests = []

#Find the best two features in test data set which gives least error and calculate the Test Error Rate
for i in range(0,y-2):
   for j in range(i+1,y-1):
      error2 = 0
      for k in FeatureTestList:
         Distance_Tests.append(math.sqrt((Means1[i] - float(k[i]))**2 + (Means1[j] - float(k[j]))**2))
         Distance_Tests.append(math.sqrt((Means2[i] - float(k[i]))**2 + (Means2[j] - float(k[j]))**2))
         Distance_Tests.append(math.sqrt((Means3[i] - float(k[i]))**2 + (Means3[j] - float(k[j]))**2))
         if int(k[-1]) != (Distance_Tests.index(min(Distance_Tests))+1):
            error2 = error2 + 1
         Distance_Tests = []
      if error2 < Minimum_Error_Test:
         Minimum_Error_Test = error2
         features_test = [i,j]

Minimum_Error_Test_Rate = Minimum_Error_Test/len(FeatureTestList)

Features = [[row[features_train[0]],row[features_train[1]]] for row in FeatureList]
result4 = np.array(Features).astype("float")
plotDecBoundaries(result4,result2,result3)

#Find the Test Error corresponding to the two features in training which gave minimal error
for i in FeatureTestList:
   DistanceTest.append(math.sqrt((Mean1 - float(i[features_train[0]]))**2 + (Mean2 - float(i[features_train[1]]))**2))
   DistanceTest.append(math.sqrt((Mean3 - float(i[features_train[0]]))**2 + (Mean4 - float(i[features_train[1]]))**2))
   DistanceTest.append(math.sqrt((Mean5 - float(i[features_train[0]]))**2 + (Mean6 - float(i[features_train[1]]))**2))
   i.append(DistanceTest.index(min(DistanceTest))+1)
   DistanceTest = []

Test_error_for_besfea = 0

for i in FeatureTestList:
   if int(i[-1]) != int(i[-2]):
      Test_error_for_besfea = Test_error_for_besfea + 1
```

Test_Error_Rate_besfea = Test_error_for_besfea/len(FeatureTestList)

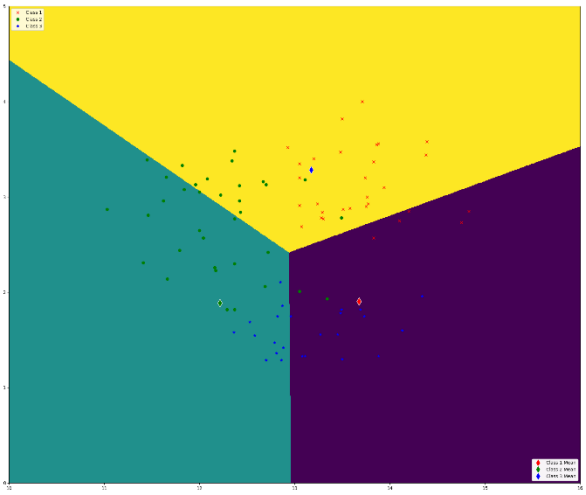
Figures :



Data Points in Wine_train



Decision Boundary based on first two features



Decision Boundary based on two best features

**Considering first two features, we get:**
**Train_Error_Rate = 0.20224719101123595**
**Test_Error_Rate = 0.2247191011235955**

**For the best two features:**
**Features which give best result: Feature no 1 and Feature no 12**
**Minimum_Train_Error_Rate = 0.07865168539325842**

**For the same two features, on the test set we get the following result:**
**Test_Error_Rate_for_best_features = 0.7752808988764045**

**If we select the two features from test set which give the minimum test error rate, we get:**
**Minimum_Test_Error_Rate = 0.11235955056179775**

**Description of method of choosing best two features:**

I ran three nested for loops. The for loops run through each combination of two features from 1,2 to 12,13. For each combination, I checked the distance of each point in the training dataset from the means and classified them according to the minimum distance from the means. This way by comparing between the given class labels and obtained class value, I got the Training_Error involved with each combination of features. I stored the minimum Training Error got and selected the corresponding combination as the best features.

In the training data set, I got the feature nos 1 and 12 as the best features as they gave the least Training Error.

**In the wine training data set, there were vast differences on the errors which I got for two different combinations of features. The least was 7 but the maximum error was around 51 errors for certain combinations of features. This is because of the way the decision boundary are drawn which in some case can lead to wrong classification of a data into a different class. This may also be because of the fact that for some combination of features, the mean of a wrong class may be closer to some particular data point which can again lead to wrong classification when the classifier is run. Hence I got various errors for the different datasets.**