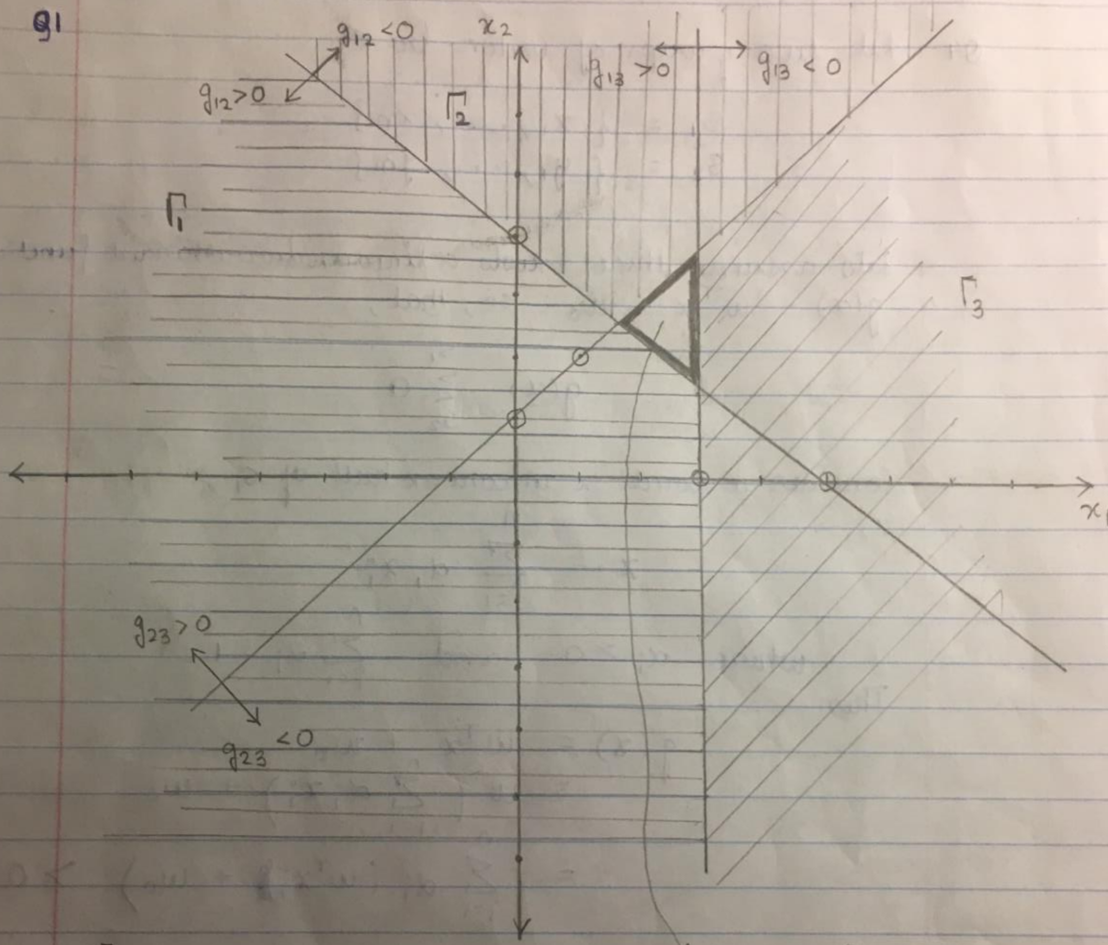
Q1

Q1



Indeterminate Region

Γ_1 — class 1
Γ_2 — class 2
Γ_3 — class 3

$\underline{x} = (4,1) \longrightarrow$ CLASS 3 (Γ_3)
$\underline{x} = (1,5) \longrightarrow$ CLASS 2 (Γ_2)
$\underline{x} = (0,0) \longrightarrow$ CLASS 1 (Γ_1)

eg. $\underline{x} = (+3,+3)$ lies on decision boundary $g_{13}(\underline{x})$ and lies in indeterminate Region.
$g_{13}(3,3) = 0$ $\quad g_{23}(3,3) = -1$ $\quad$ This is a contradiction.

Q2

a. From the wine training and data set, the classification accuracy for the training set is determined is **0.7416** and the error rate is **0.2584**. On the classification accuracy for the test data is **0.7079** and the error rate is **0.2921**.

b. Plots for each individual 2 class decision boundary based on one vs rest method. The two different colors are used to demarcate the training data according to the decision boundaries.

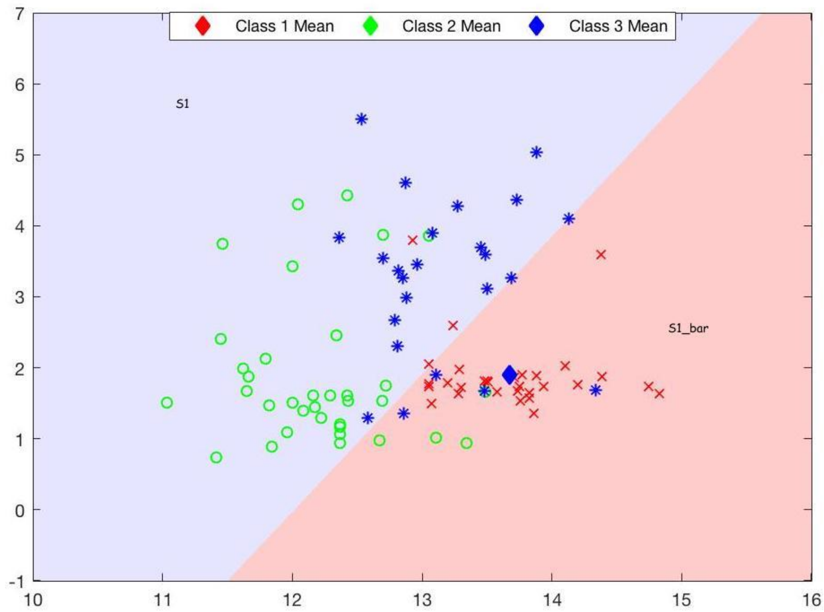


Figure 1: Class 1 and Class 1_bar

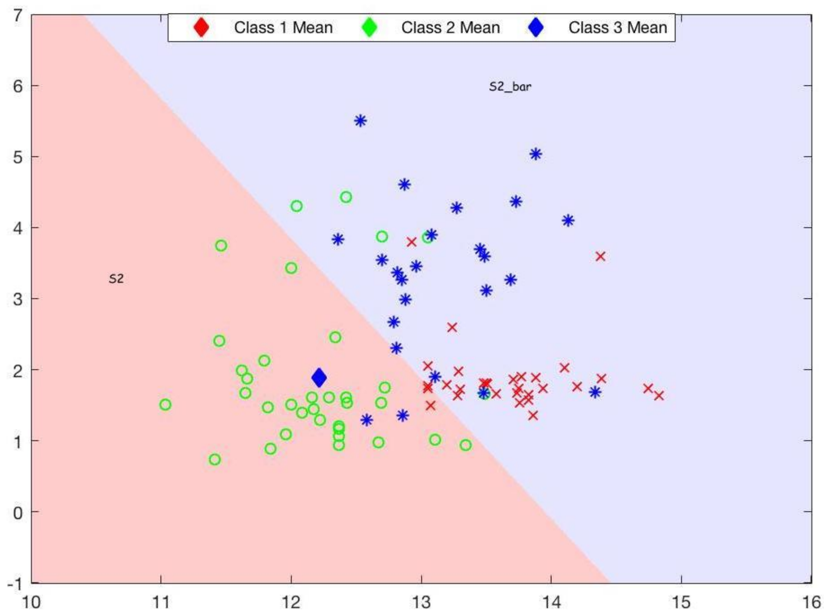


Figure 2: Class 2 and Class 2_bar

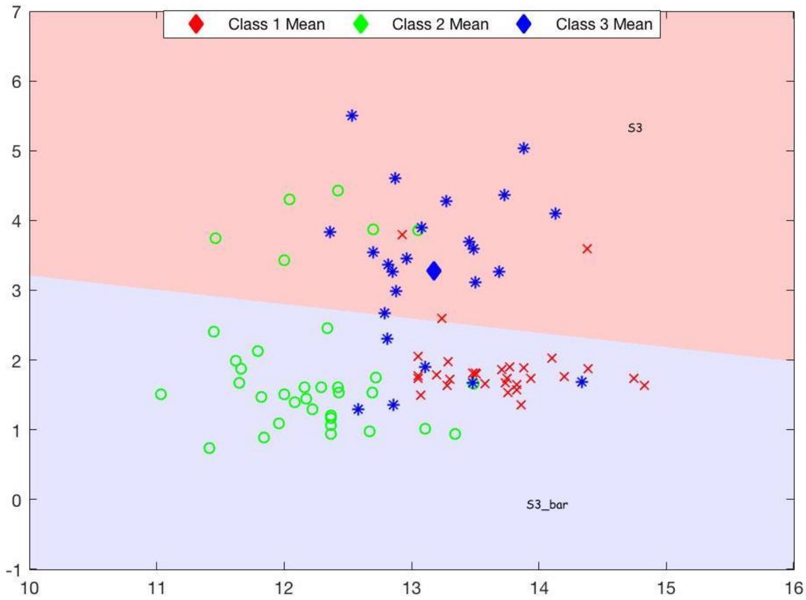
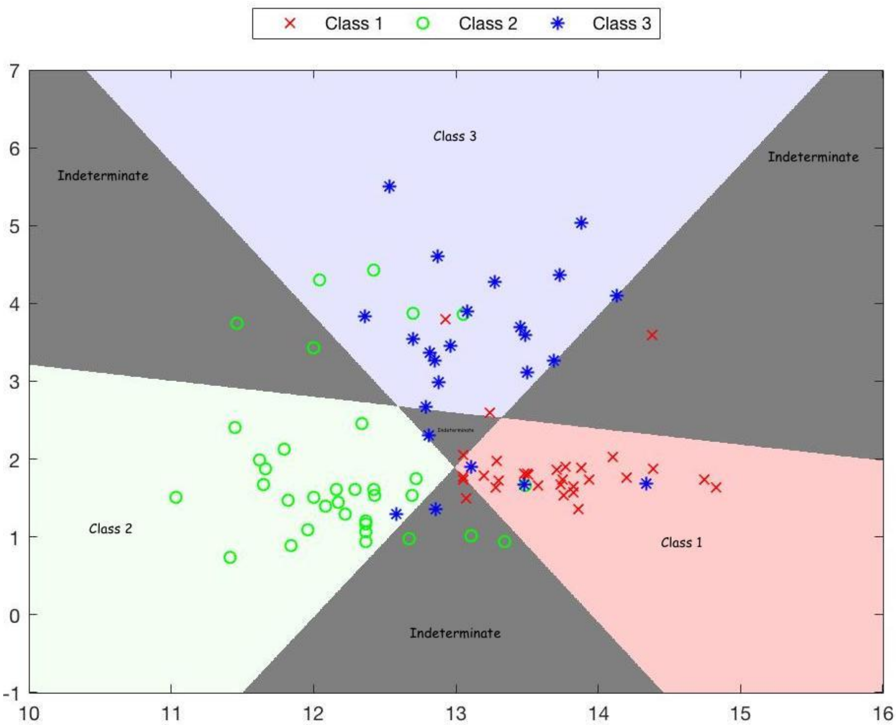


Figure 3 : Class 3 and Class 3_bar

c. Shown below is a plot showing final decision boundaries and the decision regions for the three class problem set, including the indeterminate regions. The Gray color in the plot defines the indeterminate regions which are four in number.



Codes:

i. MATLAB Code:

Main Code:

```
clc;
clear all;

%Compute sample mean

load('wine.mat');

s=size(feature_train,1);
feature_train1=zeros(s,2);
feature_train2=zeros(s,2);
feature_train3=zeros(s,2);

count1=1;
count2=1;
count3=1;



for i=1:s
   if (label_train(i)==1)
      feature_train1(count1,:)=feature_train(i,1:2);
      count1=count1+1;
   elseif(label_train(i)==2)
      feature_train2(count2,:)=feature_train(i,1:2);
      count2=count2+1;
   else
      feature_train3(count3,:)=feature_train(i,1:2);
      count3=count3+1;
   end
end

%Mean for Class 1
Mean1 = sum(feature_train1,1)./(count1-1);

%Mean for Class 2
Mean2 = sum(feature_train2,1)./(count2-1);

%Mean for Class 3
Mean3 = sum(feature_train3,1)./(count3-1);

Mean12 = (sum(feature_train1,1)+sum(feature_train2,1))./(s-count3+1);
Mean13 = (sum(feature_train1,1)+sum(feature_train3,1))./(s-count2+1);
Mean23 = (sum(feature_train2,1)+sum(feature_train3,1))./(s-count1+1);

%classification accuracy on training&test dataset
a=0;
for i=1:s
   dis1=sqrt((feature_train(i,1)-Mean1(1))^2+(feature_train(i,2)-Mean1(2))^2);
   dis2=sqrt((feature_train(i,1)-Mean2(1))^2+(feature_train(i,2)-Mean2(2))^2);
```

```
    dis3=sqrt((feature_train(i,1)-Mean3(1))^2+(feature_train(i,2)-Mean3(2))^2);
    dis12=sqrt((feature_train(i,1)-Mean12(1))^2+(feature_train(i,2)-Mean12(2))^2);
    dis13=sqrt((feature_train(i,1)-Mean13(1))^2+(feature_train(i,2)-Mean13(2))^2);
    dis23=sqrt((feature_train(i,1)-Mean23(1))^2+(feature_train(i,2)-Mean23(2))^2);
    if (dis1<dis23)&&(dis13<dis2)&&(dis12<dis3)&&(label_train(i)==1)
        a=a+1;
    end
    if (dis2<dis13)&&(dis23<dis1)&&(dis12<dis3)&&(label_train(i)==2)
        a=a+1;
    end
    if (dis3<dis12)&&(dis13<dis2)&&(dis23<dis1)&&(label_train(i)==3)
        a=a+1;
    end
end
accutrain = a./s;

a1=0;
s1=size(feature_test,1);
for i=1:s1
    dis1=sqrt((feature_test(i,1)-Mean1(1))^2+(feature_test(i,2)-Mean1(2))^2);
    dis2=sqrt((feature_test(i,1)-Mean2(1))^2+(feature_test(i,2)-Mean2(2))^2);
    dis3=sqrt((feature_test(i,1)-Mean3(1))^2+(feature_test(i,2)-Mean3(2))^2);
    dis12=sqrt((feature_test(i,1)-Mean12(1))^2+(feature_test(i,2)-Mean12(2))^2);
    dis13=sqrt((feature_test(i,1)-Mean13(1))^2+(feature_test(i,2)-Mean13(2))^2);
    dis23=sqrt((feature_test(i,1)-Mean23(1))^2+(feature_test(i,2)-Mean23(2))^2);
    if (dis1<dis23)&&(dis13<dis2)&&(dis12<dis3)&&(label_test(i)==1)
        a1=a1+1;
    elseif (dis2<dis13)&&(dis23<dis1)&&(dis12<dis3)&&(label_test(i)==2)
        a1=a1+1;
    elseif (dis3<dis12)&&(dis13<dis2)&&(dis23<dis1)&&(label_test(i)==3)
        a1=a1+1;
    end
end
accutest = a1./s;

%plot 2-class decision boundraies
sample_mean1=[Mean1;Mean23];
sample_mean2=[Mean2;Mean13];
sample_mean3=[Mean3;Mean12];
plotDecBoundaries(feature_train, label_train, sample_mean1);
plotDecBoundaries(feature_train, label_train, sample_mean2);
plotDecBoundaries(feature_train, label_train, sample_mean3);

%plot final decision boundraies and regions
plotDecBoundaries3(feature_train, label_train, sample_mean1,sample_mean2,sample_mean3);
```

Code for plotDecBoundaries3:

```
function [] = plotDecBoundaries3(training, label_train, sample_mean1,sample_mean2,sample_mean3)
%Plot the decision boundaries and data points for minimum distance to
%class mean classifier

% training: traning data
% label_train: class lables correspond to training data
% sample_mean: mean vector for each class
```

```
% Total number of classes
nclass =  max(unique(label_train));
nclasss = 2;

% Set the feature range for ploting
max_x = ceil(max(training(:, 1))) + 1;
min_x = floor(min(training(:, 1))) - 1;
max_y = ceil(max(training(:, 2))) + 1;
min_y = floor(min(training(:, 2))) - 1;

xrange = [min_x max_x];
yrange = [min_y max_y];

% step size for how finely you want to visualize the decision boundary.
inc = 0.005;

% generate grid coordinates. this will be the basis of the decision
% boundary visualization.
[x, y] = meshgrid(xrange(1):inc:xrange(2), yrange(1):inc:yrange(2));

% size of the (x, y) image, which will also be the size of the
% decision boundary image that is used as the plot background.
image_size = size(x);                                   %question!
xy = [x(:) y(:)]; % make (x,y) pairs as a bunch of row vectors.

% distance measure evaluations for each (x,y) pair.
dist_mat1 = pdist2(xy, sample_mean1); %dist_mat1 is x*y by 2.
[~, pred_label1] = min(dist_mat1, [], 2); %pred_lable1 is x*y by 1, value is 1 or 2
dist_mat2 = pdist2(xy, sample_mean2);
[~, pred_label2] = min(dist_mat2, [], 2);
dist_mat3 = pdist2(xy, sample_mean3);
[~, pred_label3] = min(dist_mat3, [], 2);

pred_label=zeros(size(pred_label1,1),1);
for i=1:size(pred_label1,1)
   if (pred_label1(i,:)==1)&&(pred_label2(i,:)==2)&&(pred_label3(i,:)==2)
      pred_label(i,:)=1;
   elseif (pred_label1(i,:)==2)&&(pred_label2(i,:)==1)&&(pred_label3(i,:)==2)
      pred_label(i,:)=2;
   elseif (pred_label1(i,:)==2)&&(pred_label2(i,:)==2)&&(pred_label3(i,:)==1)
      pred_label(i,:)=3;
   else
      pred_label(i,:)=4;
   end
end
% reshape the idx (which contains the class label) into an image.
decisionmap = reshape(pred_label, image_size);

figure;

%show the image, give each coordinate a color according to its class label
imagesc(xrange,yrange,decisionmap);
hold on;
set(gca,'ydir','normal');

% colormap for the classes:
cmap = [1 0.8 0.8; 0.95 1 0.95; 0.9 0.9 1;0.5 0.5 0.5];
```

```
colormap(cmap);

% plot the class training data.
plot(training(label_train == 1,1),training(label_train == 1,2), 'rx');
plot(training(label_train == 2,1),training(label_train == 2,2),'go');
if nclass == 3
   plot(training(label_train == 3,1),training(label_train == 3,2), 'b*');
end

% include legend for training data
if nclass == 3
   legend('Class 1', 'Class 2', 'Class 3', ...
   'Location','northoutside','Orientation', 'horizontal');
else
   legend('Class 1', 'Class 2', ...
   'Location','northoutside','Orientation', 'horizontal');
end

% plot the class mean vector.
% mean1 = plot(sample_mean(1,1),sample_mean(1,2), 'rd', ...
%            'MarkerSize', 8, 'MarkerFaceColor', 'r');
% mean2 = plot(sample_mean(2,1),sample_mean(2,2), 'gd', ...
%            'MarkerSize', 8, 'MarkerFaceColor', 'g');
% if nclasss == 3
%    mean3 = plot(sample_mean(3,1),sample_mean(3,2), 'bd',...
%              'MarkerSize', 8, 'MarkerFaceColor', 'b');
% end

% create a new axis for lengends of class mean vectors
ah=axes('position',get(gca,'position'),'visible','off');

% include legend for class mean vector
% if nclasss == 3
%    legend(ah, [mean1, mean2, mean3], {'Class 1 Mean', 'Class 2 Mean', 'Class 3 Mean'}, ...
%    'Location','northoutside','Orientation', 'horizontal');
% else
%    legend(ah, [mean1, mean2], {'NewClass 1 Mean', 'NewClass 2 Mean'},  ...
%    'Location','northoutside','Orientation', 'horizontal');
% end

% label the axes.
xlabel('featureX');
ylabel('featureY');
end
```

ii. Python 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
from PlotDecBoundary import plotDecBoundary
```

```
#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])

Class1 = copy.deepcopy(Class)
Class2 = copy.deepcopy(Class)
Class3 = copy.deepcopy(Class)

result1not1 = np.array(Class1).astype("float")
result2not2 = np.array(Class2).astype("float")
result3not3 = np.array(Class3).astype("float")

result1not1[result1not1 == 3] = 2
result2not2[result2not2 == 3] = 1
result3not3[result3not3 == 2] = 1
result3not3[result3not3 == 3] = 2

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")

Sum13 = Sum1 + Sum3
Sum35 = Sum5 + Sum3
Sum15 = Sum1 + Sum5
Sum24 = Sum2 + Sum4
Sum26 = Sum2 + Sum6
Sum46 = Sum4 + Sum6

Mean13 = Sum13/(count1+count2)
Mean35 = Sum35/(count2+count3)
Mean15 = Sum15/(count1+count3)
Mean24 = Sum24/(count1+count2)
Mean26 = Sum26/(count1+count3)
Mean46 = Sum46/(count2+count3)

Means1 = [[Mean1,Mean2],[Mean35,Mean46]]
result4 = np.array(Means1).astype("float")

Means2 = [[Mean3,Mean4],[Mean15,Mean26]]
result5 = np.array(Means2).astype("float")

Means3 = [[Mean5,Mean6],[Mean13,Mean24]]
result6 = np.array(Means3).astype("float")
```

```
MeanFinal =
[[Mean1,Mean2],[Mean35,Mean46],[Mean3,Mean4],[Mean15,Mean26],[Mean5,Mean6],[Mean
13,Mean24]]
result7 = np.array(MeanFinal).astype("float")

MeanVal1 = [[Mean1,Mean2]]
result8 = np.array(MeanVal1).astype("float")

MeanVal2 = [[Mean3,Mean4]]
result9 = np.array(MeanVal2).astype("float")

MeanVal3 = [[Mean4,Mean5]]
result10 = np.array(MeanVal3).astype("float")

#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((Mean35 -
float(i[0]))**2 + (Mean46 - float(i[1]))**2):
      i.append(int(1))
   else:
      i.append(int(2))

for i in FeatureList:
   if math.sqrt((Mean3 - float(i[0]))**2 + (Mean4 - float(i[1]))**2) < math.sqrt((Mean15 -
float(i[0]))**2 + (Mean26 - float(i[1]))**2):
      i.append(int(2))
   else:
      i.append(int(3))

for i in FeatureList:
   if math.sqrt((Mean5 - float(i[0]))**2 + (Mean6 - float(i[1]))**2) < math.sqrt((Mean13 -
float(i[0]))**2 + (Mean24 - float(i[1]))**2):
      i.append(int(3))
   else:
      i.append(int(1))

Num1 = []
Num2 = []
Num3 = []
Final = []

for i in FeatureList:
   if i[-3] == 1:
      Num1.append(int(1))
   else:
      Num1.append(int(0))

for i in FeatureList:
   if i[-2] == 2:
      Num2.append(int(1))
   else:
      Num2.append(int(0))

for i in FeatureList:
   if i[-1] == 3:
      Num3.append(int(1))
   else:
```

```
        Num3.append(int(0))

for i in range(0,len(FeatureList)):
    if( Num1[i]+Num2[i]+Num3[i] > 1):
        Final.append('4')
    elif( Num1[i]+Num2[i]+Num3[i] == 0):
        Final.append('4')
    elif( Num1[i]+Num2[i]+Num3[i] == 1):
        if( Num1[i] == 1):
            Final.append('1')
        elif( Num2[i] == 1):
            Final.append('2')
        elif( Num3[i] == 1):
            Final.append('3')

for i in range(0,3):
    for row in FeatureList:
        del row[-1]

Train_error = 0

#Find the training data error rate
for i in range(0,len(FeatureList)):
    if int(FeatureList[i][-1]) != int(Final[i]):
        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()

#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((Mean35 - 
float(i[0]))**2 + (Mean46 - float(i[1]))**2):
        i.append(int(1))
    else:
        i.append(int(2))

for i in FeatureTestList:
    if math.sqrt((Mean3 - float(i[0]))**2 + (Mean4 - float(i[1]))**2) < math.sqrt((Mean15 - 
float(i[0]))**2 + (Mean26 - float(i[1]))**2):
        i.append(int(2))
    else:
        i.append(int(3))

for i in FeatureTestList:
    if math.sqrt((Mean5 - float(i[0]))**2 + (Mean6 - float(i[1]))**2) < math.sqrt((Mean13 - 
float(i[0]))**2 + (Mean24 - float(i[1]))**2):
        i.append(int(3))
```

```
    else:
        i.append(int(1))

Num1 = []
Num2 = []
Num3 = []
Final = []

for i in FeatureTestList:
    if int(i[-3]) == 1:
        Num1.append(int(1))
    else:
        Num1.append(int(0))

for i in FeatureTestList:
    if int(i[-2]) == 2:
        Num2.append(int(1))
    else:
        Num2.append(int(0))

for i in FeatureTestList:
    if int(i[-1]) == 3:
        Num3.append(int(1))
    else:
        Num3.append(int(0))

for i in range(0,len(FeatureTestList)):
    if( Num1[i]+Num2[i]+Num3[i] > 1):
        Final.append('4')
    elif( Num1[i]+Num2[i]+Num3[i] == 0):
        Final.append('4')
    elif( Num1[i]+Num2[i]+Num3[i] == 1):
        if( Num1[i] == 1):
            Final.append('1')
        elif( Num2[i] == 1):
            Final.append('2')
        elif( Num3[i] == 1):
            Final.append('3')

for i in range(0,3):
    for row in FeatureTestList:
      del row[-1]

Test_error = 0

for i in range(0,len(FeatureTestList)):
    if int(FeatureTestList[i][-1]) != int(Final[i]):
        Test_error = Test_error + 1

Test_Error_Rate = Test_error/len(FeatureTestList)

#Plotting Decision Boundaries for each two class regions formed
plotDecBoundaries(result1,result1not1,result4)
plotDecBoundaries(result1,result2not2,result5)
plotDecBoundaries(result1,result3not3,result6)
```

It Gives the same results till Question 2 part b. I faced some difficulties in the code for Question 2 part c, hence not included here.

Q4 Let two groups of vectors be :

$$S_1 = \{x_1, \dots, x_n\}$$
$$S_2 = \{y_1, \dots, y_m\}$$

Lets assume there exists a linear discriminant function $g(x) = w^T x + w_0$, so, that,

$$g(\underline{x}) \underset{S_2}{\overset{S_1}{\gtrless}} 0$$

Consider a point x in convex hull of S_1

$\therefore$

$$x = \sum_{i=1}^{n} \alpha_i x_i$$

where $\alpha_i > 0$ and $\sum_{i=1}^{n} \alpha_i = 1$

Then,

$$\begin{aligned} g(x) &= w^T x + w_0 \\ &= w^T\left(\sum_{i=1}^{n} \alpha_i x_i\right) + w_0 \\ &= \sum_{i=1}^{n} \alpha_i \left(w^T x_i + w_0\right) > 0 \end{aligned}$$

$\because$ $w^T x_i + w_0 > 0$ for $1 \le i \le n$ and $\sum_{i=1}^{n} \alpha_i = 1$

Now, let x is also in convex hull of S_2

$\therefore$

$$x = \sum_{i=1}^{m} \beta_i y_i$$

where $\beta_i > 0$ and $\sum_{i=1}^{m} \beta_i = 1$

Then,

$$\begin{aligned} g(x) &= w^T x + w_0 \\ &= w^T\left(\sum_{i=1}^{m} \beta_i y_i\right) + w_0 = \sum_{i=1}^{m} \beta_i \left(w^T y_i + w_0\right) < 0 \end{aligned}$$

So, in this case $g(x) < 0$
Because, $g(y_j) = w^T y_i + w_0 < 0$ for each y_i as they lie in S_2.

Thus we have a contradiction for same point x such that $g(x) > 0$ and $g(x) < 0$ and hence clearly, the intersection is empty.

$\therefore$ Either two sets of vectors are either linearly separable or their convex hulls intersect.

Q3. a)

$\underline{x}$ A (x_1, x_2) B $\mu_1 [\mu_{11}, \mu_{12}]$

The distance AB is given by

$$\sqrt{(\mu_{11} - x_1)^2 + (\mu_{12} - x_2)^2} = (\mu_1 - \underline{x})^T (\mu_1 - \underline{x})$$

So we can say that in minimum distance to class means classifier, we have to

$$\min \ (\mu_1 - x)^T (\mu_1 - x)$$
$$= \min \ \mu_1^T \mu_1 - 2x^T \mu_1 + x^T x$$
$$\simeq \min \ -2x^T \mu_1 + \mu_1^T \mu_1$$
$$\simeq \max \ x^T \mu_1 - \frac{1}{2} \mu_1^T \mu_1$$

So, $g(x) = g_1(x) - g_2(x)$
$$= x^T \mu_1 - \frac{1}{2} \mu_1^T \mu_1 - x^T \mu_2 + \frac{1}{2} \mu_2^T \mu_2$$
$$= x^T (\mu_1 - \mu_2) - \frac{1}{2} (\mu_1^T \mu_1 - \mu_2^T \mu_2)$$

We know, $g(x) = w_0 + w^T x$

So comparing,

$$w_0 = -\frac{1}{2}(\mu_1^T\mu_1 - \mu_2^T\mu_2)$$

$$w = \mu_1 - \mu_2$$

so, if $g(x) > 0$ assign x to class 1
if $g(x) < 0$ assign x to class 2
if $g(x) = 0$ x is in dundecided region.
The classifier is linear.

b)

$$\mu_1 = \begin{bmatrix} 0 \\ -2 \end{bmatrix} \qquad \mu_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mu_1^T\mu_1 = [0\ -2]\begin{bmatrix} 0 \\ -2 \end{bmatrix} = 4$$

$$\mu_1 - \mu_2 = \begin{bmatrix} 0 \\ -3 \end{bmatrix}$$

$$\mu_2^T\mu_2 = [0\ 1]\begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1$$

so,

$$w = \begin{bmatrix} 0 \\ -3 \end{bmatrix}$$

$$w_0 = -\frac{1}{2}(4-1) = -\frac{3}{2}.$$

so,

$$g(x) = -\frac{3}{2} + [x_1\ x_2]\begin{bmatrix} 0 \\ -3 \end{bmatrix}$$

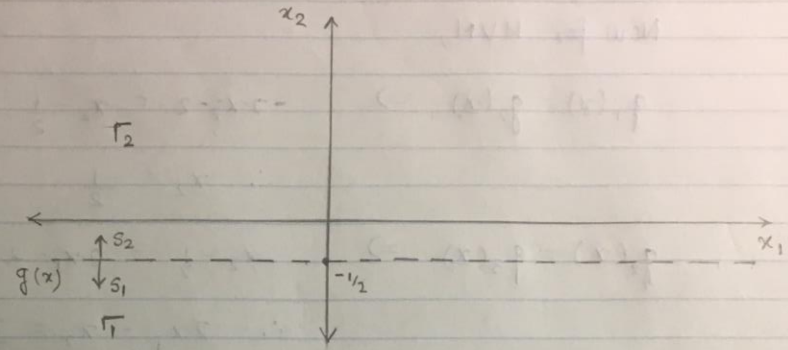
$$= -\frac{3}{2} - 3x_2$$

so,

$$\boxed{x_2 = -\frac{1}{2}}$$

Hence,

decision boundary:



From (a),

c) $g_1(\underline{x}) = x^T\mu_1 - \frac{1}{2}\mu_1^T\mu_1$ $\qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$g_2(\underline{x}) = x^T\mu_2 - \frac{1}{2}\mu_2^T\mu_2$

$\mu = \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}$

$g_3(\underline{x}) = x^T\mu_3 - \frac{1}{2}\mu_3^T\mu_3$

d) $\mu_1 = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$ $\quad \mu_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad \mu_3 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$

$$g_1(\underline{x}) = [x_1 \ \ x_2]\begin{bmatrix} 0 \\ -2 \end{bmatrix} - \frac{1}{2}[0 \ -2]\begin{bmatrix} 0 \\ -2 \end{bmatrix}$$

$$= -2x_2 - 2$$

$$g_2(\underline{x}) = [x_1 \ \ x_2]\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \frac{1}{2}[0 \ 1]\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= x_2 - \frac{1}{2}$$

$$g_3(\underline{x}) = [x_1 \ \ x_2]\begin{bmatrix} 2 \\ 0 \end{bmatrix} - \frac{1}{2}[2 \ 0]\begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$= 2x_1 - 2$$

They are all linear classifiers.

Now for MVM,

$g_1(\underline{x}) = g_2(\underline{x}) \Rightarrow \quad -2x_2 - 2 = x_2 - \frac{1}{2}$

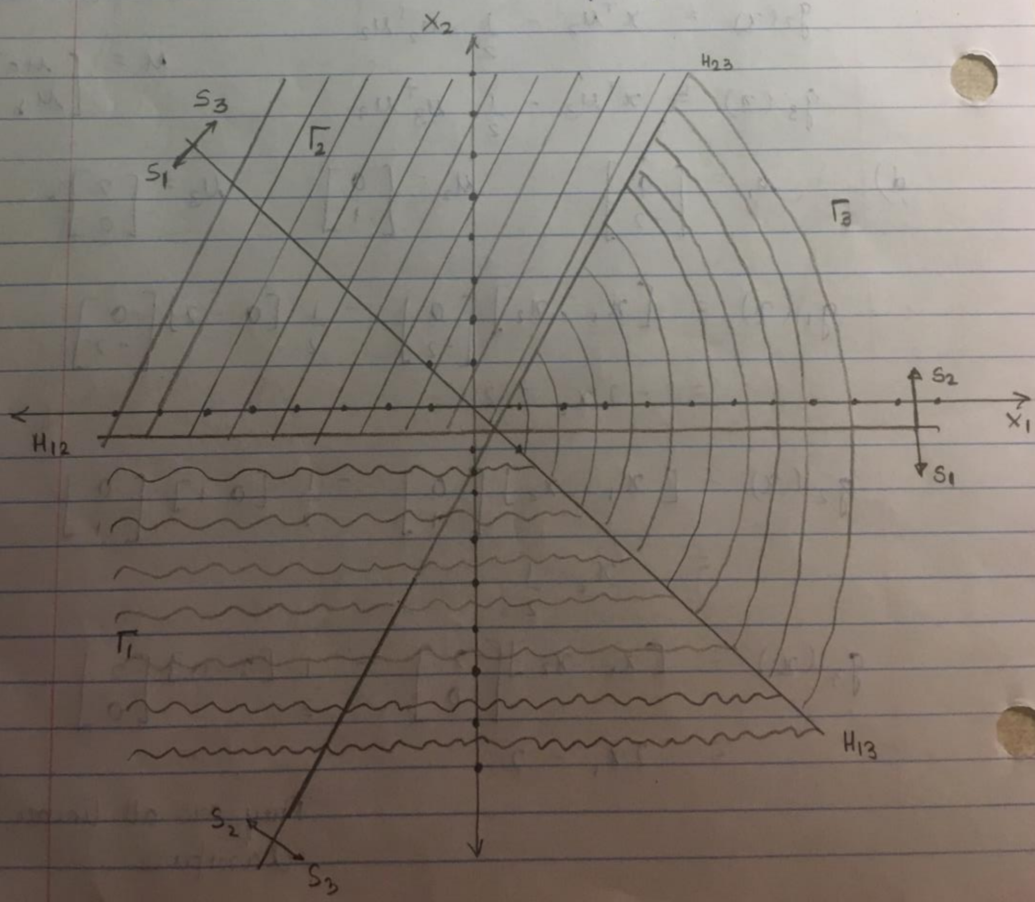
$\therefore \quad x_2 = \frac{-1}{2}$

$g_2(\underline{x}) = g_3(\underline{x}) \Rightarrow \quad x_2 - \frac{1}{2} = 2x_1 - 2$

$\therefore \quad 2x_1 - x_2 = \frac{3}{2}$

$g_1(\underline{x}) = g_3(\underline{x}) \Rightarrow \quad -2x_2 - 2 = 2x_1 - 2$

$\therefore \quad x_1 + x_2 = 0$



c) From (a), we have,

$$\sqrt{(\mu_{11} - x_1)^2 + (\mu_{12} - x_2)^2}$$

$$= (\mu_1 - \underline{x})^T (\mu_1 - \underline{x})$$

$$= \mu_1^T \mu_1 - 2 x^T \mu_1 + x^T x$$

$$\simeq \min \quad - 2 x^T \mu_1 + \mu_1^T \mu_1$$

$$\simeq \max \quad x^T \mu_1 - \frac{1}{2} \mu_1^T \mu_1$$

So, $$g(x) = x^T \mu_i - \frac{1}{2} \mu_i^T \mu_i$$

So, we get,

$$g_1(x) = x^T \mu_1 - \frac{1}{2} \mu_1^T \mu_1$$

$$g_2(x) = x^T \mu_2 - \frac{1}{2} \mu_2^T \mu_2$$

$$g_3(x) = x^T \mu_3 - \frac{1}{2} \mu_3^T \mu_3$$