EE559-Homework#1-Lei Lei

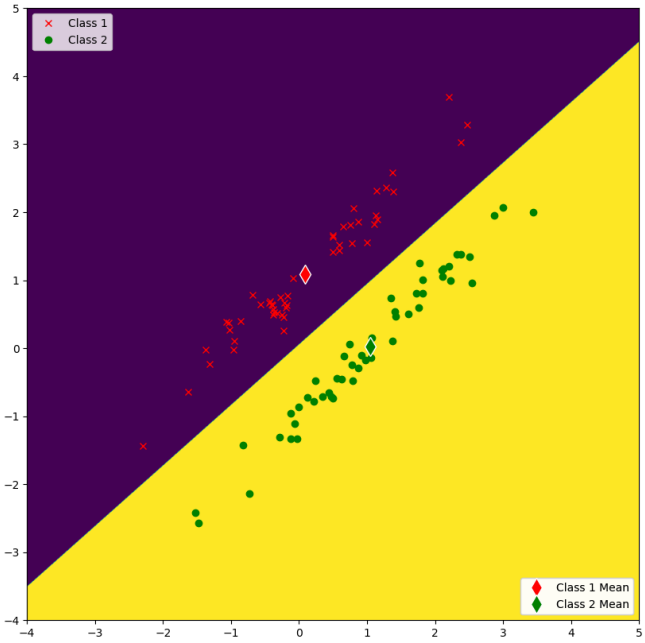
1. Write code to implement a 2-class nearest-means classifier for data that has 2 features.

(a) *Learning (training) and classification*. Use unnormalized data as supplied in the datasets.
For each dataset (1, 2, and 3), do the following.
Compute the class means on the training data.

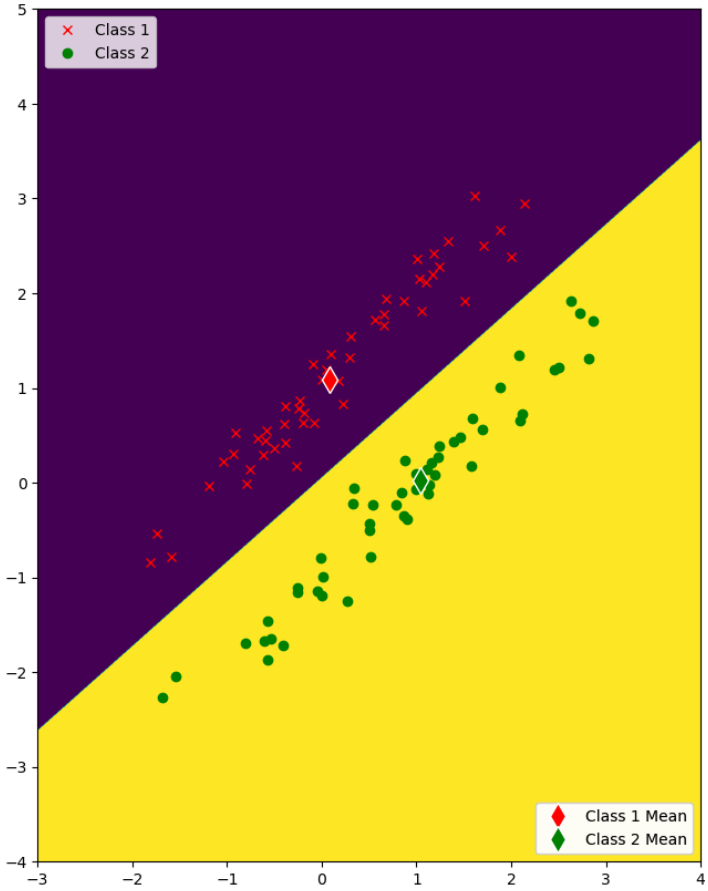
For the dataset1_train :
The sample mean for class 1 data is: 0.08893115, 1.08956606
The sample mean for class 2 data is: 1.04128622, 0.01994688
For the dataset2_train :
The sample mean for class 1 data is: -0.3536011, 0.99036239
The sample mean for class 2 data is: 1.03652797, -0.03223129
For the dataset3_train :
The sample mean for class 1 data is: -0.06738853, 0.21324203
The sample mean for class 2 data is: 0.52230078, 0.93267215

(i) Plot the training data (using different colors or symbols for the different classes), the class means, the decision boundary, and decision regions.
Classify all data points in the training set and in the test set, using the class means computed above.
(ii) Report the classification error rate on the training set, and separately on the test set.
Classification error rate = (Number of misclassified points) / (total number of points), expressed as percentage.

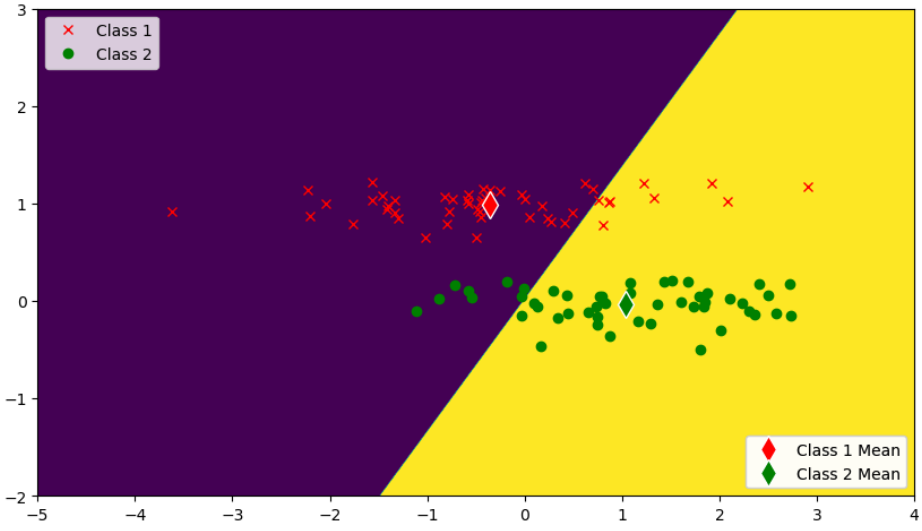
For the dataset1_train:
Error rate = 0.000%



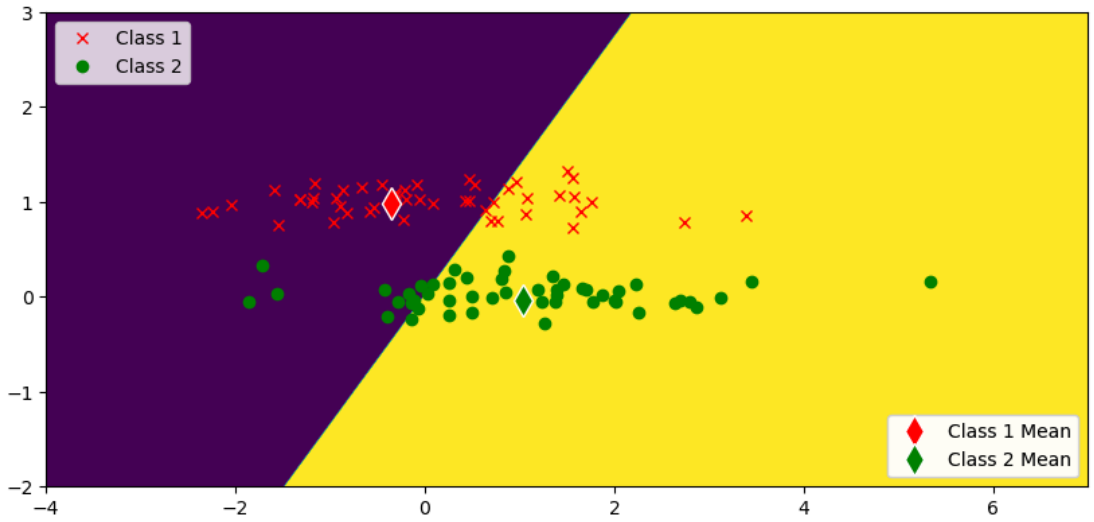
For the dataset1_test:
Error rate = 0.000%



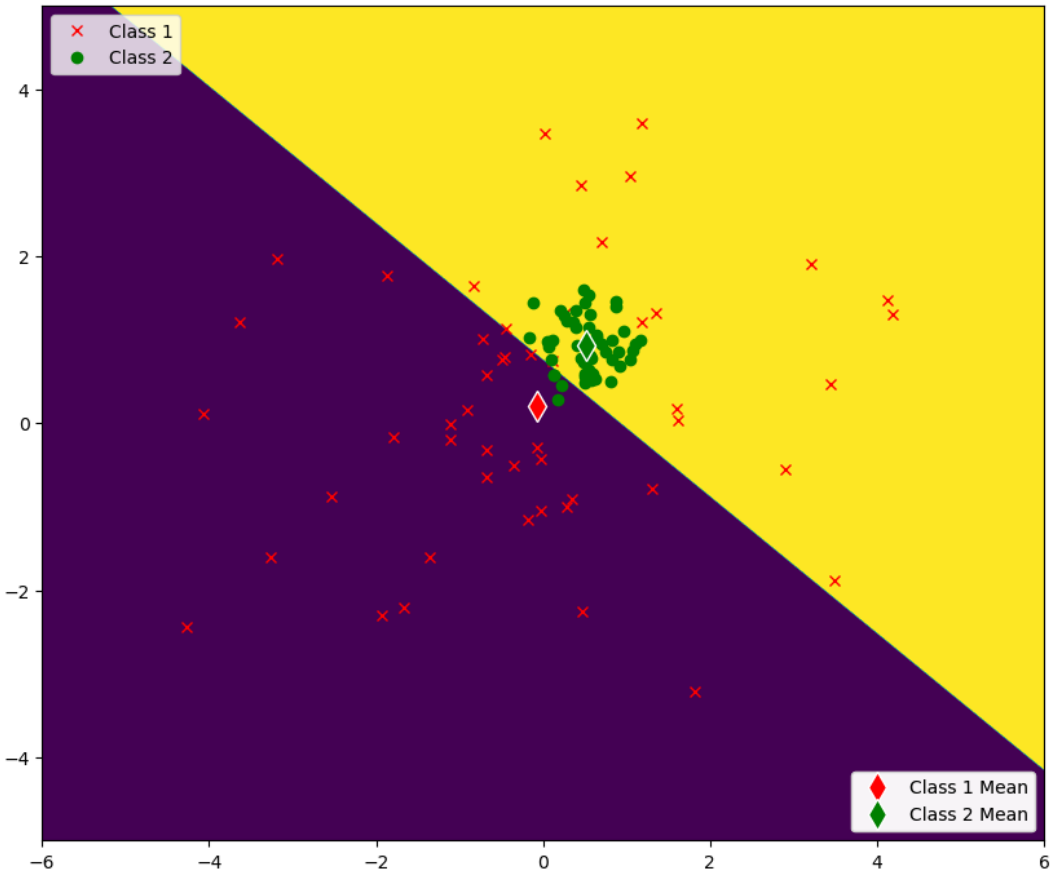
For the dataset2_train:
Error rate = 17.000%



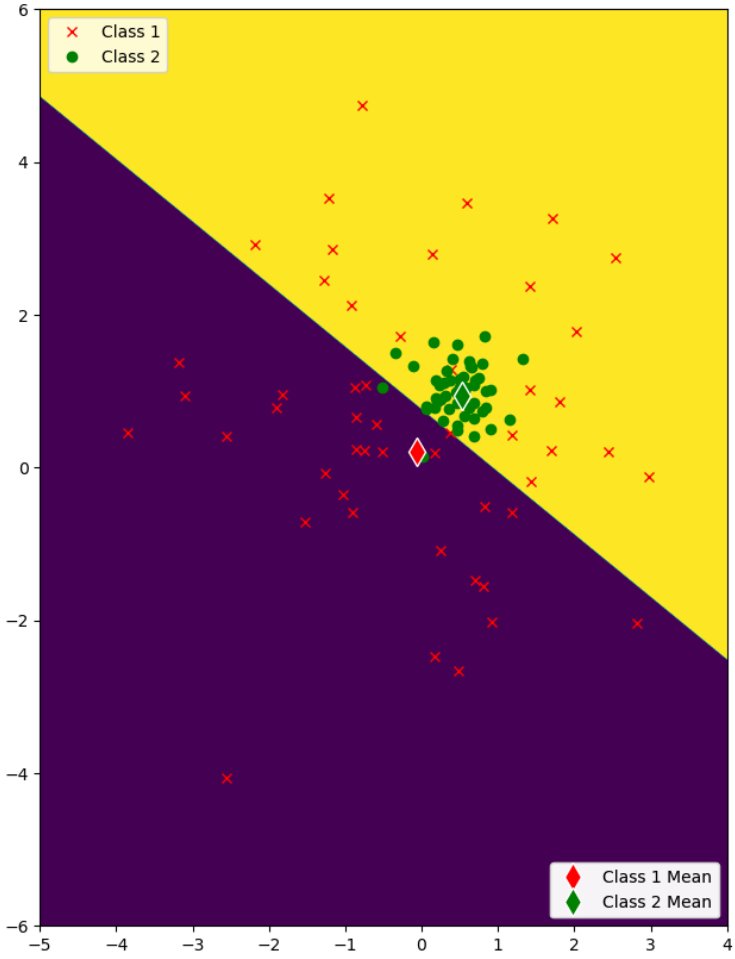
For the dataset2_test:
Error rate = 26.000%



For the dataset3_train:
Error rate = 23.000%



For the dataset3_test:
Error rate = 23.000%



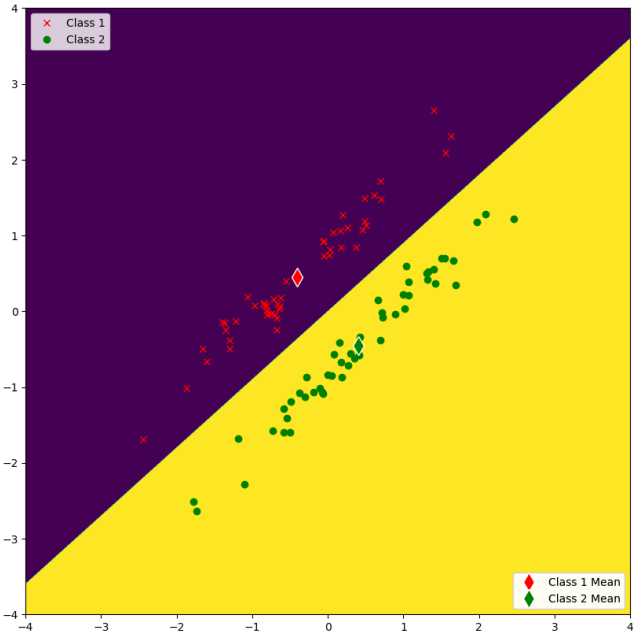
(b) Compare and comment on the results: how do the test error rates on datasets 1, 2, and 3 compare? Try to explain why.

The error rates of both train1 and test1 are 0 % because the data distribution and their means are very similar and close.
The error rates of train2 and test2 are different from each other due to the data distribution. Although the data seems to be parallel to each other but the line connecting their midpoints is a diagonal line. Therefore, there must be some data points crossed the border and get the wrong classification.
The error rates of train3 and test3 are the same 23% because one of the clusters is dispersed. So that the mean value cannot precisely describe the characteristic of the data distribution.

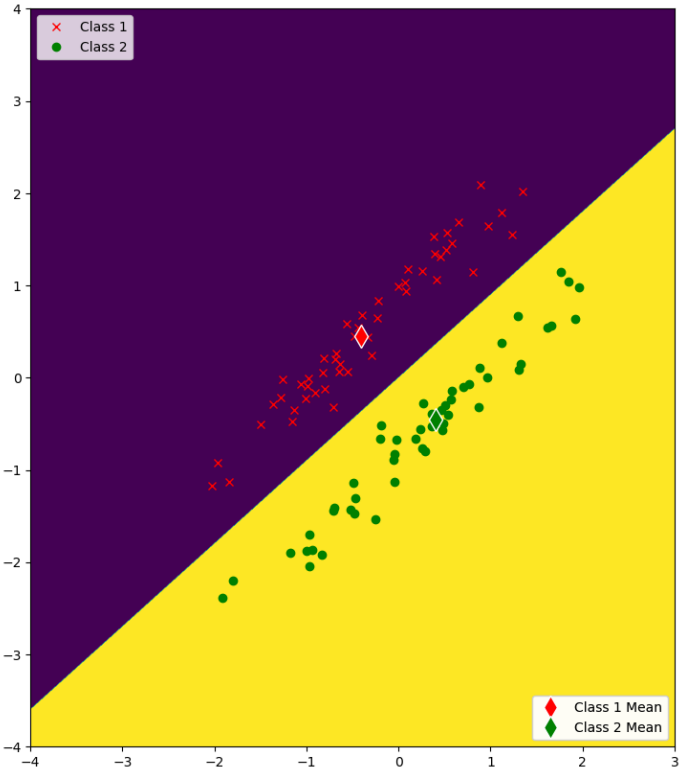
(c) *Preprocessing: normalization.* Standardize the data (so that each feature, across both classes combined, has sample mean = 0 and sample variance = 1). For each dataset, compute the normalizing parameters from the training data, and then use those parameter values to standardize the training data and test data. The result is the (standardized) data you will use for this part.

Repeat part (a), except on the normalized data.

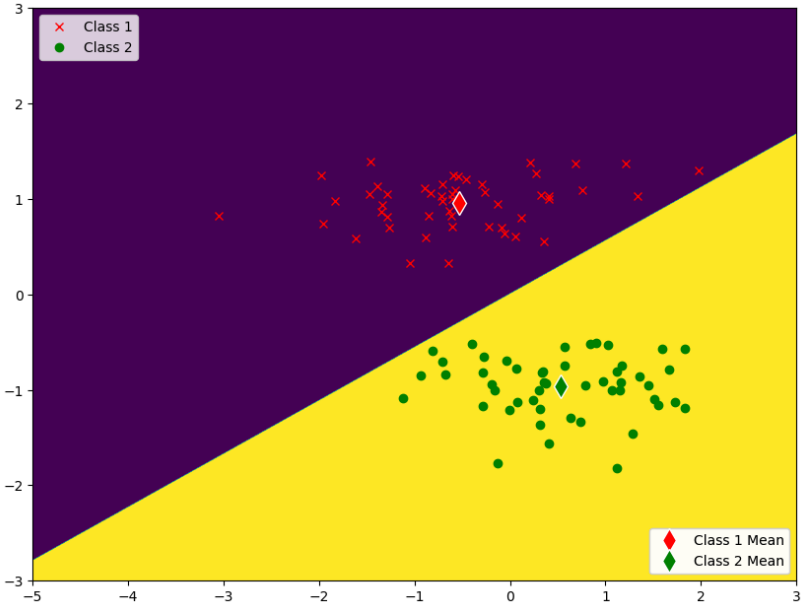
For the dataset1_train:
The sample mean for class 1 data is: -0.40664534, 0.45213344
The sample mean for class 2 data is: 0.40664534, -0.45213344
Error rate = 0.000%



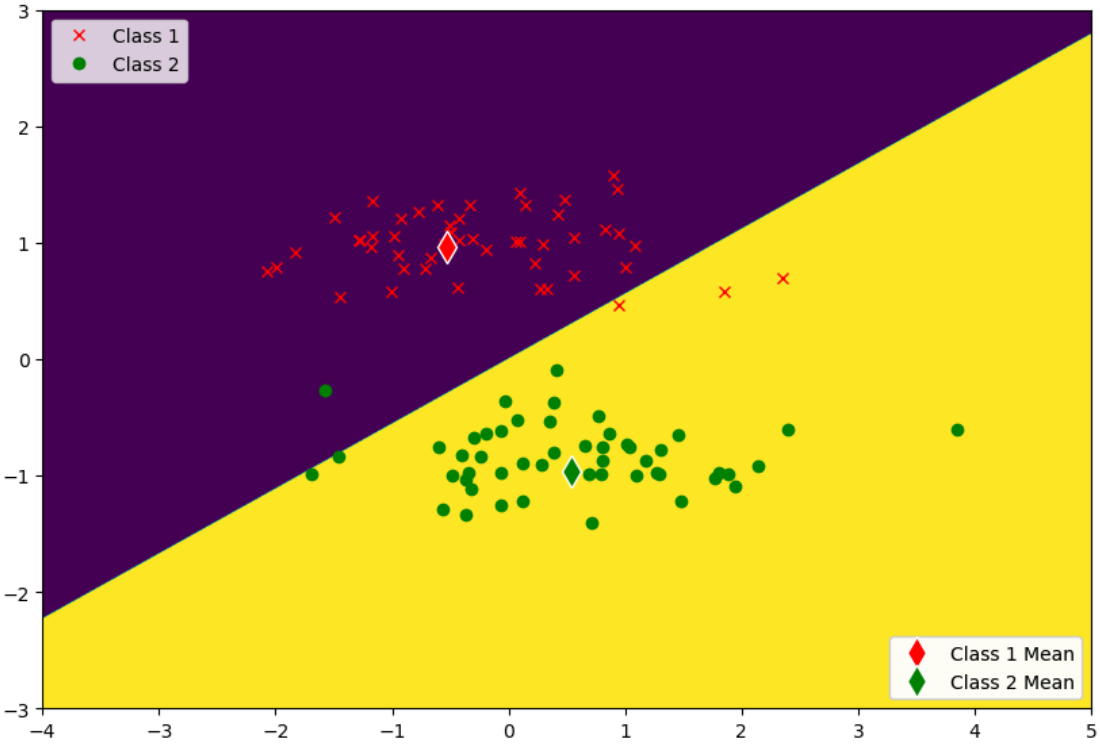
For the dataset1_test:
Error rate = 0.000%



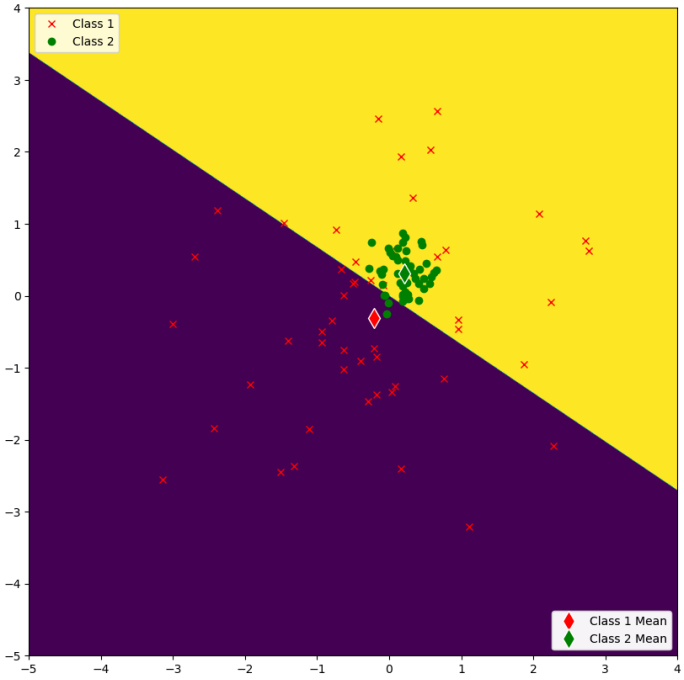
For the dataset2_train :
The sample mean for class 1 data is: -0.53490724, 0.95882365
The sample mean for class 2 data is: 0.53490724, -0.95882365
Error rate = 0.000%



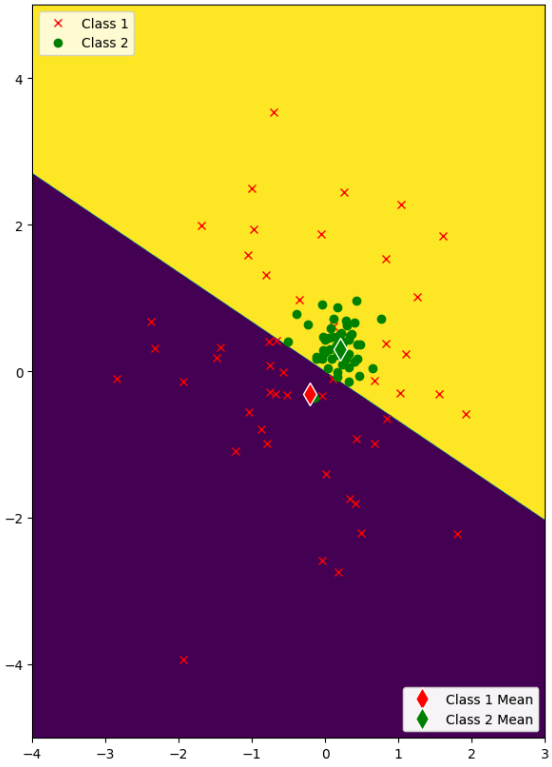
For the dataset2_test :
Error rate = 4.000%



For the dataset3_train :
The sample mean for class 1 data is: -0.2061046, -0.30508577
The sample mean for class 2 data is: 0.2061046, 0.30508577
Error rate = 24.000%



For the dataset3_test :
Error rate = 21.000%



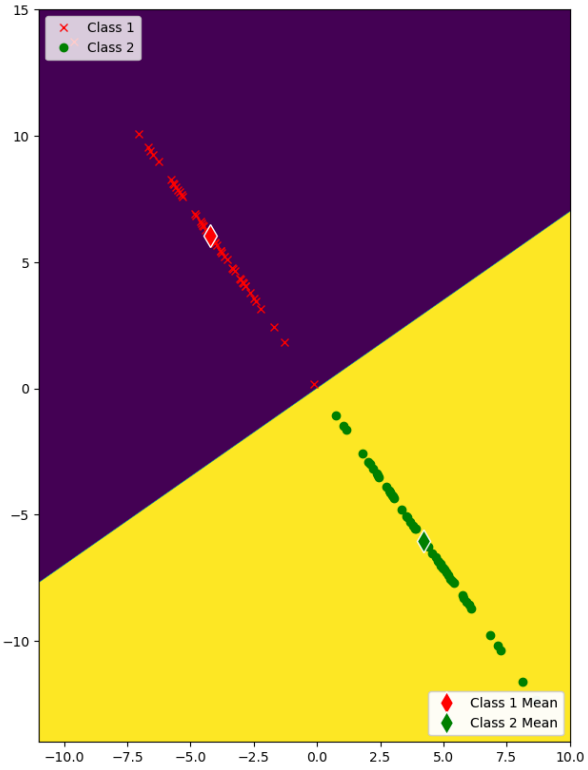
(d) Compare and comment on the results of part (a) to those of part (c): how do the error rates on normalized (standardized) and unnormalized data compare for each given dataset? Try to explain why.

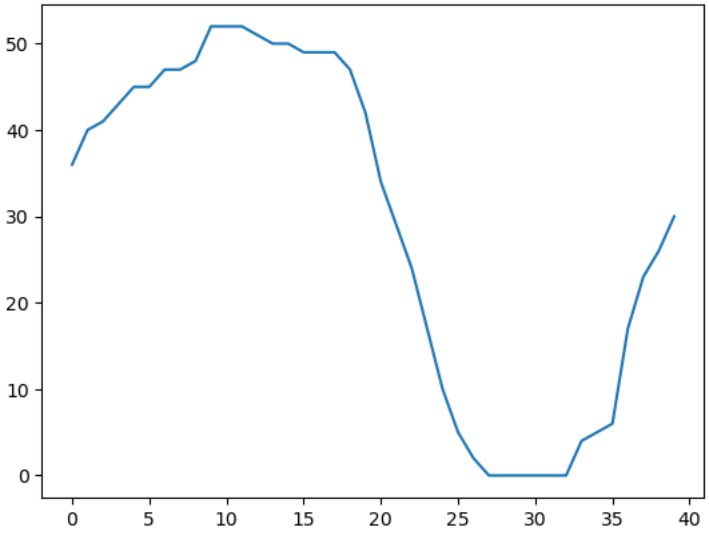
The error rate of both dataset 1 remains 0.00% because the process of normalization donot change the data distribution of the data set. Using nearest mean function can divide two cluster greatly.
The error rates of dataset 2 are 0.00% and 4.00% which are smaller than the 17.00% and 26.00%. Probably because the normalization process makes the standard deviation equals to 1 which means the cluster is much closer to the midpoint so that the nearest mean function behaves better.
The error rates of dataset 3 remain larger than 20% which are 24.00% and 21.00% since the data distribution of class 1 is dispersed. Therefore, although the normalized data is much closer to the midpoint there still contain part of the cluster overlapping with the class 2 data.

(e) *Feature transformation: feature reduction by projection.* For this part start with the standardized data. Reduce the number of features to 1 by using a projection transformation, as described below.
A projection transformation will project all data points onto the same line, to result in a new, 1D feature space.
For each (standardized) dataset, do the following:
To get a somewhat optimal result for the feature transformation, try different directions for the line by using the following brute-force technique on the training set:
Try 40 different vector directions r to project onto, as follows: !
$r = (10, m)$, m = 0,1,2,⋯,9 !
$r_m = (20 - m, 10)$, m = 10,11,12, ⋯ ,29
$r = (-10, 40 - m)$, m = 30,31,32, ⋯ ,39 !
For each value of m, project all the training data points onto r . The result will be a set data ! points in 1D (projected) feature space. Then train the 1D classifier by computing the class means. Next, classify all the training data points in the 1D feature space.
To optimize the feature transformation, plot the training error rate vs. m. Report which value(s) of m give(s) the lowest training error. Call it m^*; if more than one value of m give the lowest training error, then let m^* be the lowest such m.
Also report r_{m^*}.
Then classify all the points in the test set in the 1D space for $m = m^*$. Show the projected points, as well as the decision boundaries and regions, on a 2D plot (using the same axes as the other 2D plots), all for $m = m^*$. Report the resulting test error.

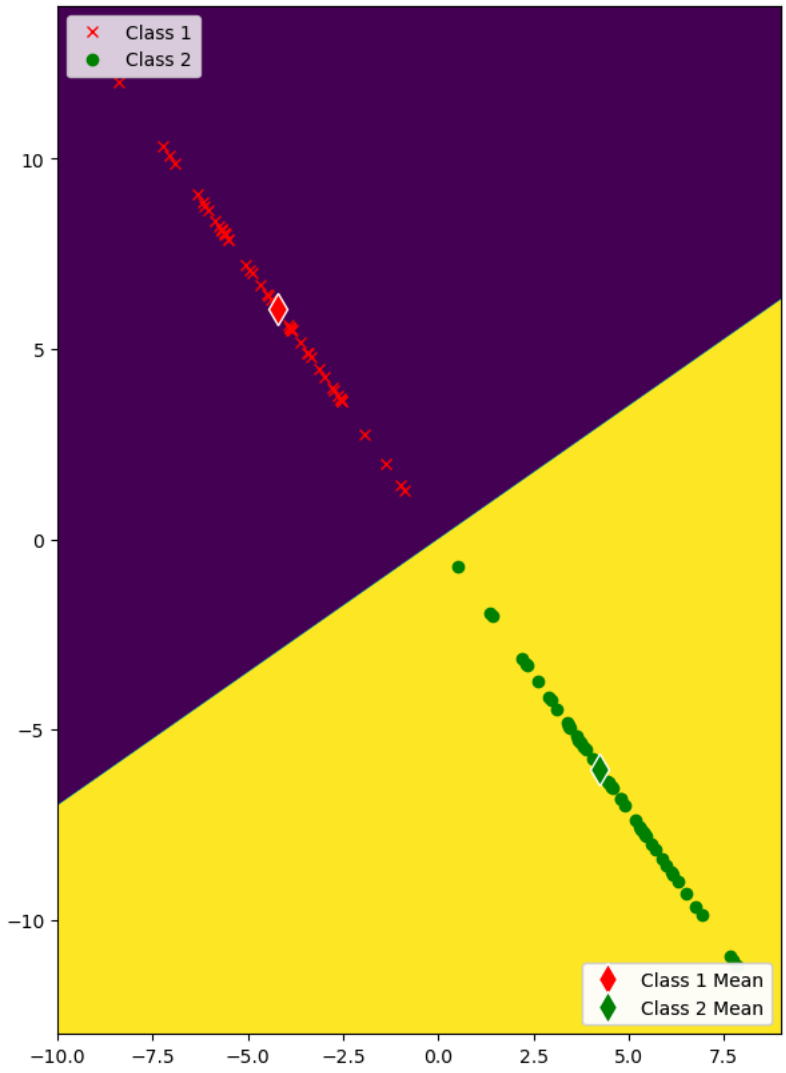
For dataset1_train:
m* = 27 , error rate = 0.000%, rm* = [-7, 10]

```
m =  0 ,rm =  [10, 0] ,errorRate =  36.000%
m =  1 ,rm =  [10, 1] ,errorRate =  40.000%
m =  2 ,rm =  [10, 2] ,errorRate =  41.000%
m =  3 ,rm =  [10, 3] ,errorRate =  43.000%
m =  4 ,rm =  [10, 4] ,errorRate =  45.000%
m =  5 ,rm =  [10, 5] ,errorRate =  45.000%
m =  6 ,rm =  [10, 6] ,errorRate =  47.000%
m =  7 ,rm =  [10, 7] ,errorRate =  47.000%
m =  8 ,rm =  [10, 8] ,errorRate =  48.000%
m =  9 ,rm =  [10, 9] ,errorRate =  52.000%
m =  10 ,rm =  [10, 10] ,errorRate =  52.000%
m =  11 ,rm =  [9, 10] ,errorRate =  52.000%
m =  12 ,rm =  [8, 10] ,errorRate =  51.000%
m =  13 ,rm =  [7, 10] ,errorRate =  50.000%
m =  14 ,rm =  [6, 10] ,errorRate =  50.000%
m =  15 ,rm =  [5, 10] ,errorRate =  49.000%
m =  16 ,rm =  [4, 10] ,errorRate =  49.000%
m =  17 ,rm =  [3, 10] ,errorRate =  49.000%
m =  18 ,rm =  [2, 10] ,errorRate =  47.000%
m =  19 ,rm =  [1, 10] ,errorRate =  42.000%
m =  20 ,rm =  [0, 10] ,errorRate =  34.000%
m =  21 ,rm =  [-1, 10] ,errorRate =  29.000%
m =  22 ,rm =  [-2, 10] ,errorRate =  24.000%
m =  23 ,rm =  [-3, 10] ,errorRate =  17.000%
m =  24 ,rm =  [-4, 10] ,errorRate =  10.000%
m =  25 ,rm =  [-5, 10] ,errorRate =  5.000%
m =  26 ,rm =  [-6, 10] ,errorRate =  2.000%
m =  27 ,rm =  [-7, 10] ,errorRate =  0.000%
m =  28 ,rm =  [-8, 10] ,errorRate =  0.000%
m =  29 ,rm =  [-9, 10] ,errorRate =  0.000%
m =  30 ,rm =  [-10, 10] ,errorRate =  0.000%
m =  31 ,rm =  [-10, 9] ,errorRate =  0.000%
m =  32 ,rm =  [-10, 8] ,errorRate =  0.000%
m =  33 ,rm =  [-10, 7] ,errorRate =  4.000%
m =  34 ,rm =  [-10, 6] ,errorRate =  5.000%
m =  35 ,rm =  [-10, 5] ,errorRate =  6.000%
m =  36 ,rm =  [-10, 4] ,errorRate =  17.000%
m =  37 ,rm =  [-10, 3] ,errorRate =  23.000%
m =  38 ,rm =  [-10, 2] ,errorRate =  26.000%
m =  39 ,rm =  [-10, 1] ,errorRate =  30.000%
```



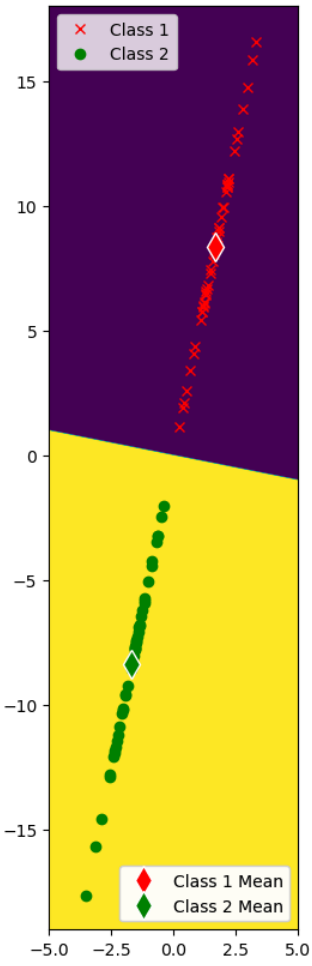


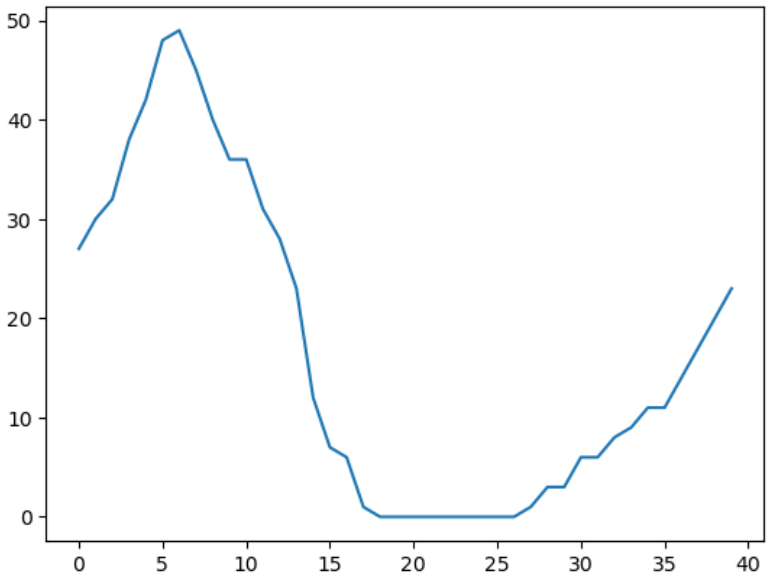
For dataset1_test:
Error rate = 0.000%



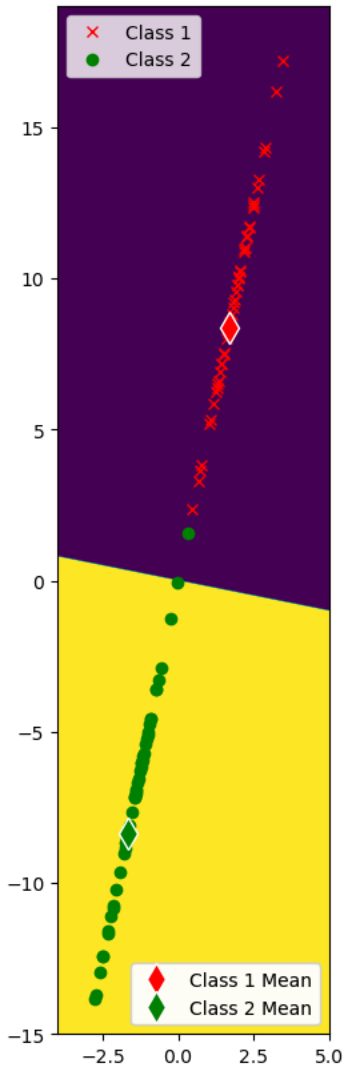
For dataset2_train:
m* = 18 , error rate = 0.000%, rm* = [2, 10]

```
m =  0 ,rm =  [10, 0] ,errorRate =  27.000%
m =  1 ,rm =  [10, 1] ,errorRate =  30.000%
m =  2 ,rm =  [10, 2] ,errorRate =  32.000%
m =  3 ,rm =  [10, 3] ,errorRate =  38.000%
m =  4 ,rm =  [10, 4] ,errorRate =  42.000%
m =  5 ,rm =  [10, 5] ,errorRate =  48.000%
m =  6 ,rm =  [10, 6] ,errorRate =  49.000%
m =  7 ,rm =  [10, 7] ,errorRate =  45.000%
m =  8 ,rm =  [10, 8] ,errorRate =  40.000%
m =  9 ,rm =  [10, 9] ,errorRate =  36.000%
m =  10 ,rm =  [10, 10] ,errorRate =  36.000%
m =  11 ,rm =  [9, 10] ,errorRate =  31.000%
m =  12 ,rm =  [8, 10] ,errorRate =  28.000%
m =  13 ,rm =  [7, 10] ,errorRate =  23.000%
m =  14 ,rm =  [6, 10] ,errorRate =  12.000%
m =  15 ,rm =  [5, 10] ,errorRate =  7.000%
m =  16 ,rm =  [4, 10] ,errorRate =  6.000%
m =  17 ,rm =  [3, 10] ,errorRate =  1.000%
m =  18 ,rm =  [2, 10] ,errorRate =  0.000%
m =  19 ,rm =  [1, 10] ,errorRate =  0.000%
m =  20 ,rm =  [0, 10] ,errorRate =  0.000%
m =  21 ,rm =  [-1, 10] ,errorRate =  0.000%
m =  22 ,rm =  [-2, 10] ,errorRate =  0.000%
m =  23 ,rm =  [-3, 10] ,errorRate =  0.000%
m =  24 ,rm =  [-4, 10] ,errorRate =  0.000%
m =  25 ,rm =  [-5, 10] ,errorRate =  0.000%
m =  26 ,rm =  [-6, 10] ,errorRate =  0.000%
m =  27 ,rm =  [-7, 10] ,errorRate =  1.000%
m =  28 ,rm =  [-8, 10] ,errorRate =  3.000%
m =  29 ,rm =  [-9, 10] ,errorRate =  3.000%
m =  30 ,rm =  [-10, 10] ,errorRate =  6.000%
m =  31 ,rm =  [-10, 9] ,errorRate =  6.000%
m =  32 ,rm =  [-10, 8] ,errorRate =  8.000%
m =  33 ,rm =  [-10, 7] ,errorRate =  9.000%
m =  34 ,rm =  [-10, 6] ,errorRate =  11.000%
m =  35 ,rm =  [-10, 5] ,errorRate =  11.000%
m =  36 ,rm =  [-10, 4] ,errorRate =  14.000%
m =  37 ,rm =  [-10, 3] ,errorRate =  17.000%
m =  38 ,rm =  [-10, 2] ,errorRate =  20.000%
m =  39 ,rm =  [-10, 1] ,errorRate =  23.000%
```



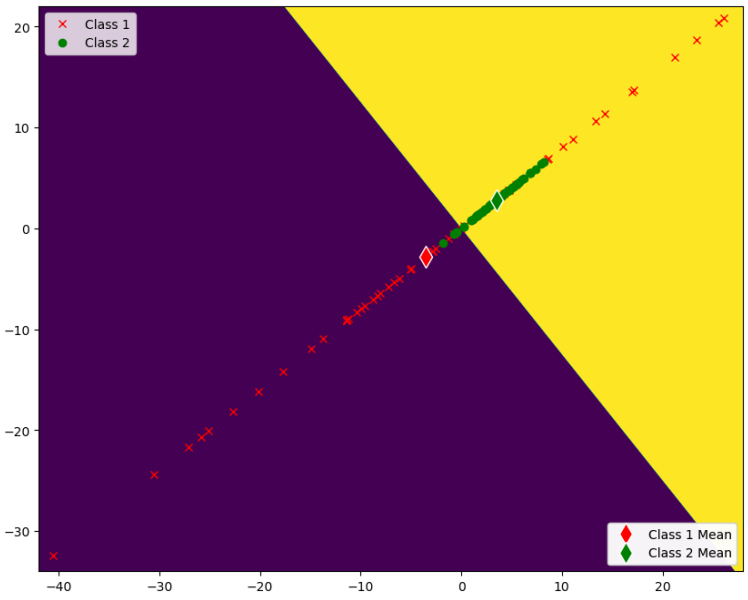


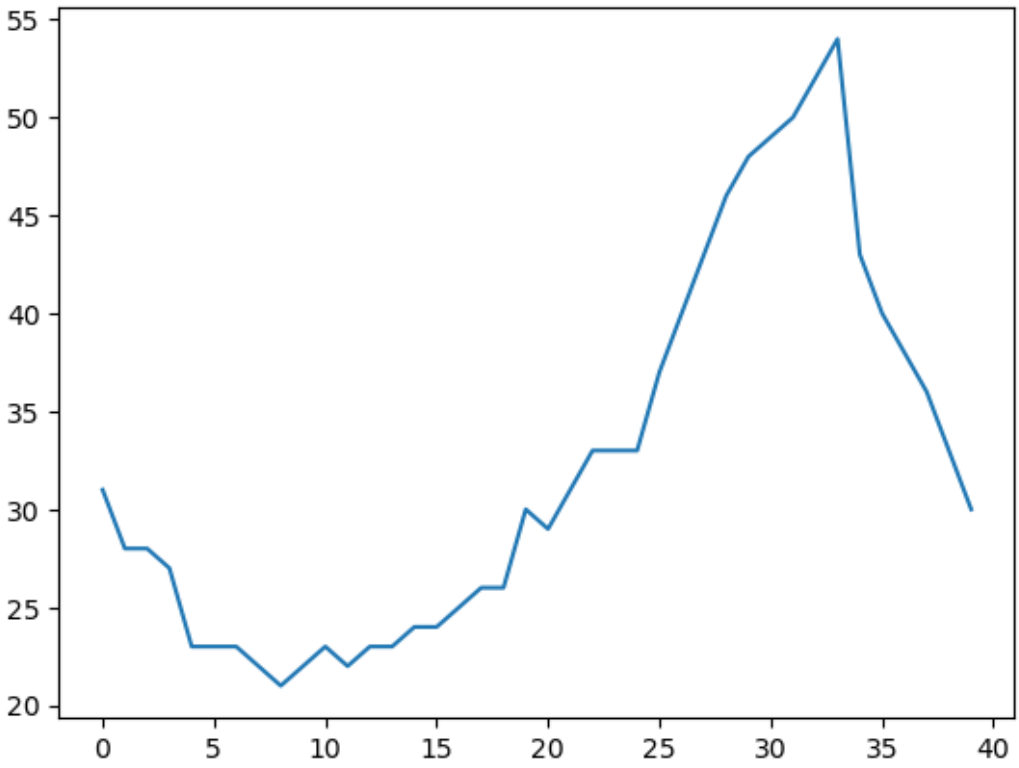
For dataset2_test:
Error rate = 1.000%



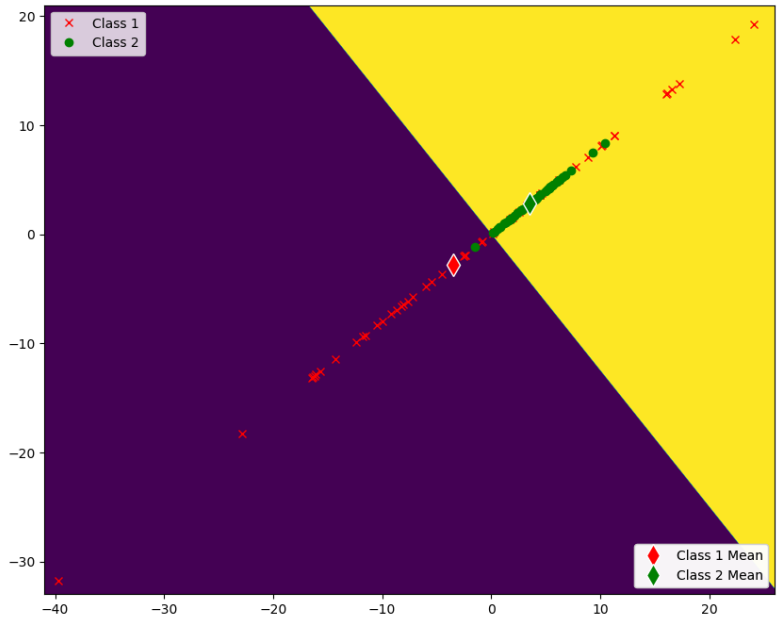
For dataset3_train:
m* = 8 , error rate = 21.000%, rm* = [10, 8]

```
m =  0 ,rm =  [10, 0] ,errorRate =  31.000%
m =  1 ,rm =  [10, 1] ,errorRate =  28.000%
m =  2 ,rm =  [10, 2] ,errorRate =  28.000%
m =  3 ,rm =  [10, 3] ,errorRate =  27.000%
m =  4 ,rm =  [10, 4] ,errorRate =  23.000%
m =  5 ,rm =  [10, 5] ,errorRate =  23.000%
m =  6 ,rm =  [10, 6] ,errorRate =  23.000%
m =  7 ,rm =  [10, 7] ,errorRate =  22.000%
m =  8 ,rm =  [10, 8] ,errorRate =  21.000%
m =  9 ,rm =  [10, 9] ,errorRate =  22.000%
m =  10 ,rm =  [10, 10] ,errorRate =  23.000%
m =  11 ,rm =  [9, 10] ,errorRate =  22.000%
m =  12 ,rm =  [8, 10] ,errorRate =  23.000%
m =  13 ,rm =  [7, 10] ,errorRate =  23.000%
m =  14 ,rm =  [6, 10] ,errorRate =  24.000%
m =  15 ,rm =  [5, 10] ,errorRate =  24.000%
m =  16 ,rm =  [4, 10] ,errorRate =  25.000%
m =  17 ,rm =  [3, 10] ,errorRate =  26.000%
m =  18 ,rm =  [2, 10] ,errorRate =  26.000%
m =  19 ,rm =  [1, 10] ,errorRate =  30.000%
m =  20 ,rm =  [0, 10] ,errorRate =  29.000%
m =  21 ,rm =  [-1, 10] ,errorRate =  31.000%
m =  22 ,rm =  [-2, 10] ,errorRate =  33.000%
m =  23 ,rm =  [-3, 10] ,errorRate =  33.000%
m =  24 ,rm =  [-4, 10] ,errorRate =  33.000%
m =  25 ,rm =  [-5, 10] ,errorRate =  37.000%
m =  26 ,rm =  [-6, 10] ,errorRate =  40.000%
m =  27 ,rm =  [-7, 10] ,errorRate =  43.000%
m =  28 ,rm =  [-8, 10] ,errorRate =  46.000%
m =  29 ,rm =  [-9, 10] ,errorRate =  48.000%
m =  30 ,rm =  [-10, 10] ,errorRate =  49.000%
m =  31 ,rm =  [-10, 9] ,errorRate =  50.000%
m =  32 ,rm =  [-10, 8] ,errorRate =  52.000%
m =  33 ,rm =  [-10, 7] ,errorRate =  54.000%
m =  34 ,rm =  [-10, 6] ,errorRate =  43.000%
m =  35 ,rm =  [-10, 5] ,errorRate =  40.000%
m =  36 ,rm =  [-10, 4] ,errorRate =  38.000%
m =  37 ,rm =  [-10, 3] ,errorRate =  36.000%
m =  38 ,rm =  [-10, 2] ,errorRate =  33.000%
m =  39 ,rm =  [-10, 1] ,errorRate =  30.000%
```





For dataset3_test:
Error rate = 24.000%



(f) Compare and comment on the results: how do the test error rates on (optimized) 1D data of part (e) compare with test error rates on 2D normalized data of part (c), for each given dataset? Try to explain why.

The error rate of both dataset 1 remains 0.00% because the mean value of those two datasets can represent the data distribution. So that the process of feature transformation don't change the relative position of the data sets. Using nearest mean function can divide two cluster greatly.
The error rates of dataset 2 are 0.00% and 1.00% which are smaller than the 0.00% and 4.00% which means the transformation prove the behavior of the classification. Because the process makes the data projected on one line and the midpoint can represent the data distribution which proves the function behavior.
The error rates of dataset 3 remain larger than 20% which are 21.00% and 24.00% since the data distribution of class 1 is dispersed. Therefore, although the feature reduction makes the data projected on one line, the data still distributes on both side of the mean value.