SUDESH KUMAR SANTHOSH KUMAR
4166249920

Report for Homework - 1 EE 559

(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.

(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.

a] Computing class means :-

Train_data_1 :-

```
--------------------------------------------------
  Shape of Input Data: (100, 3)
  Number of Data Points: 100
  Number of Input Features: 2
  Number of Target Classes: 2
--------------------------------------------------
--------------------------------------------------
  Shape of sample_means: (2, 2)
  Sample Means:

[[0.08893115 1.08956606]
 [1.04128622 0.01994688]]

--------------------------------------------------
```

Train_data_2 :-

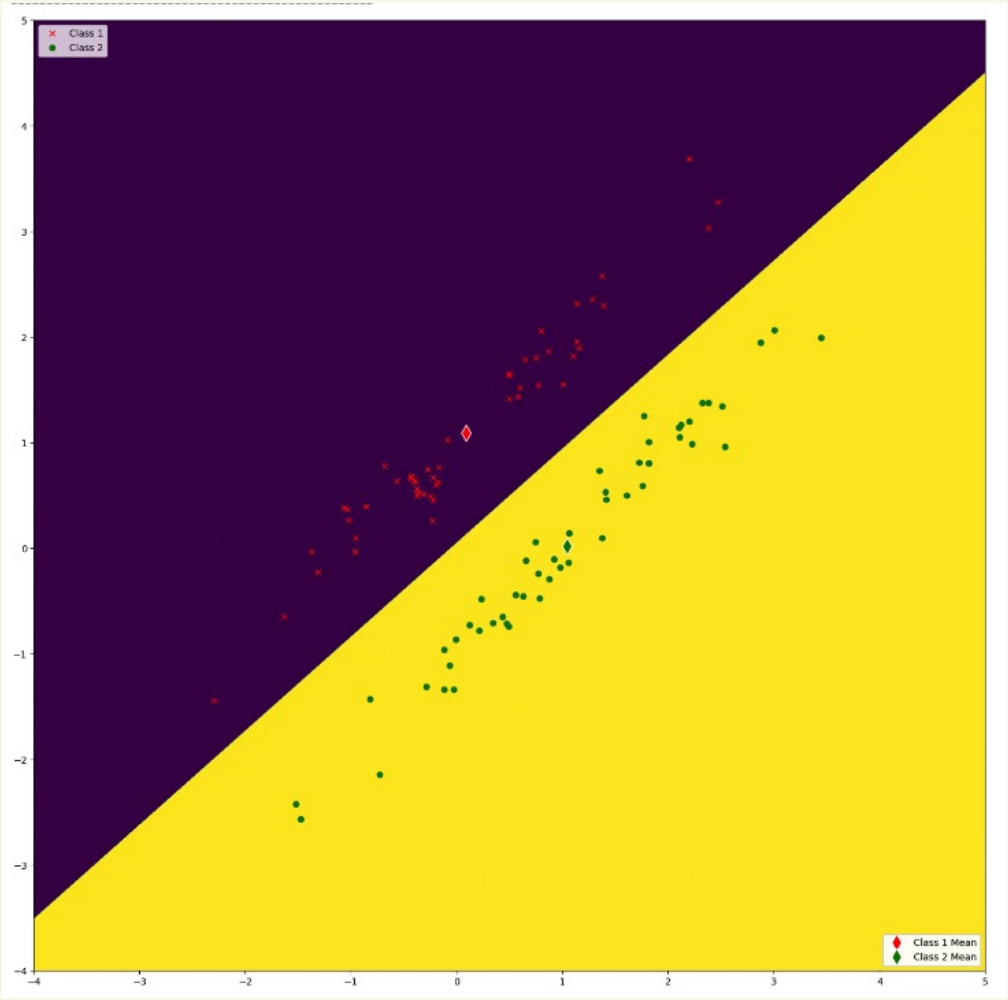
```
--------------------------------------------------
  Shape of Input Data: (100, 3)
  Number of Data Points: 100
  Number of Input Features: 2
  Number of Target Classes: 2
--------------------------------------------------
--------------------------------------------------
  Shape of sample_means: (2, 2)
  Sample Means:

[[-0.3536011   0.99036239]
 [ 1.03652797 -0.03223129]]

--------------------------------------------------
```

Train_data_3 :-

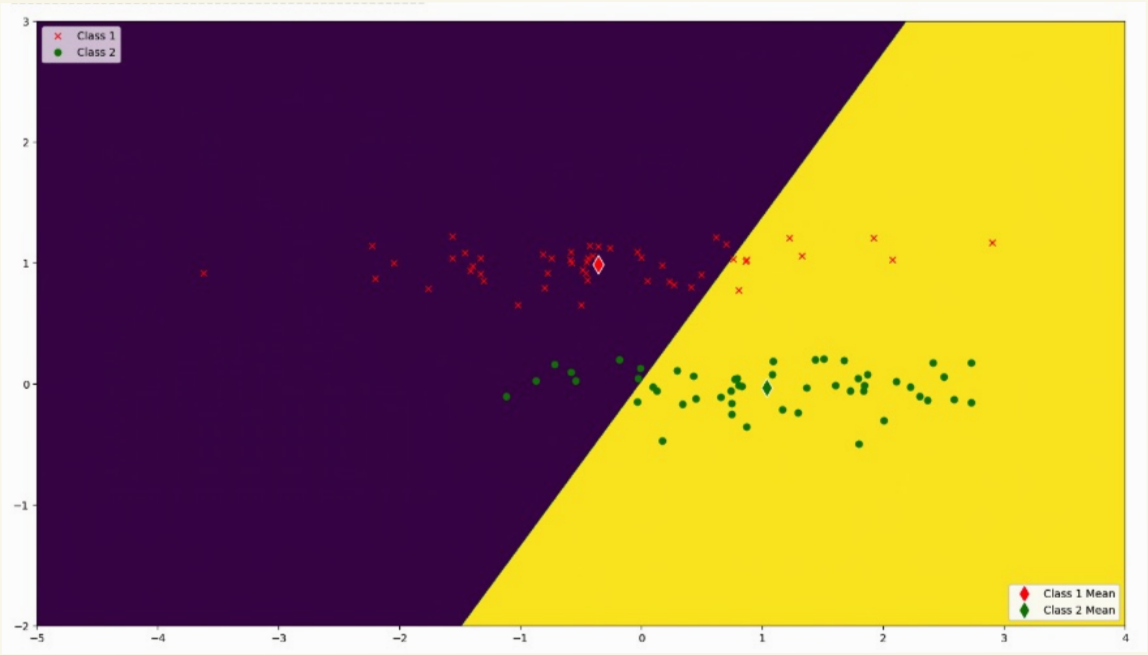
```
--------------------------------------------------
  Shape of Input Data: (100, 3)
  Number of Data Points: 100
  Number of Input Features: 2
  Number of Target Classes: 2
--------------------------------------------------
--------------------------------------------------
  Shape of sample_means: (2, 2)
  Sample Means:

[[-0.06738853  0.21324203]
 [ 0.52230078  0.93267215]]

--------------------------------------------------
```

b] Plotting the train data :-

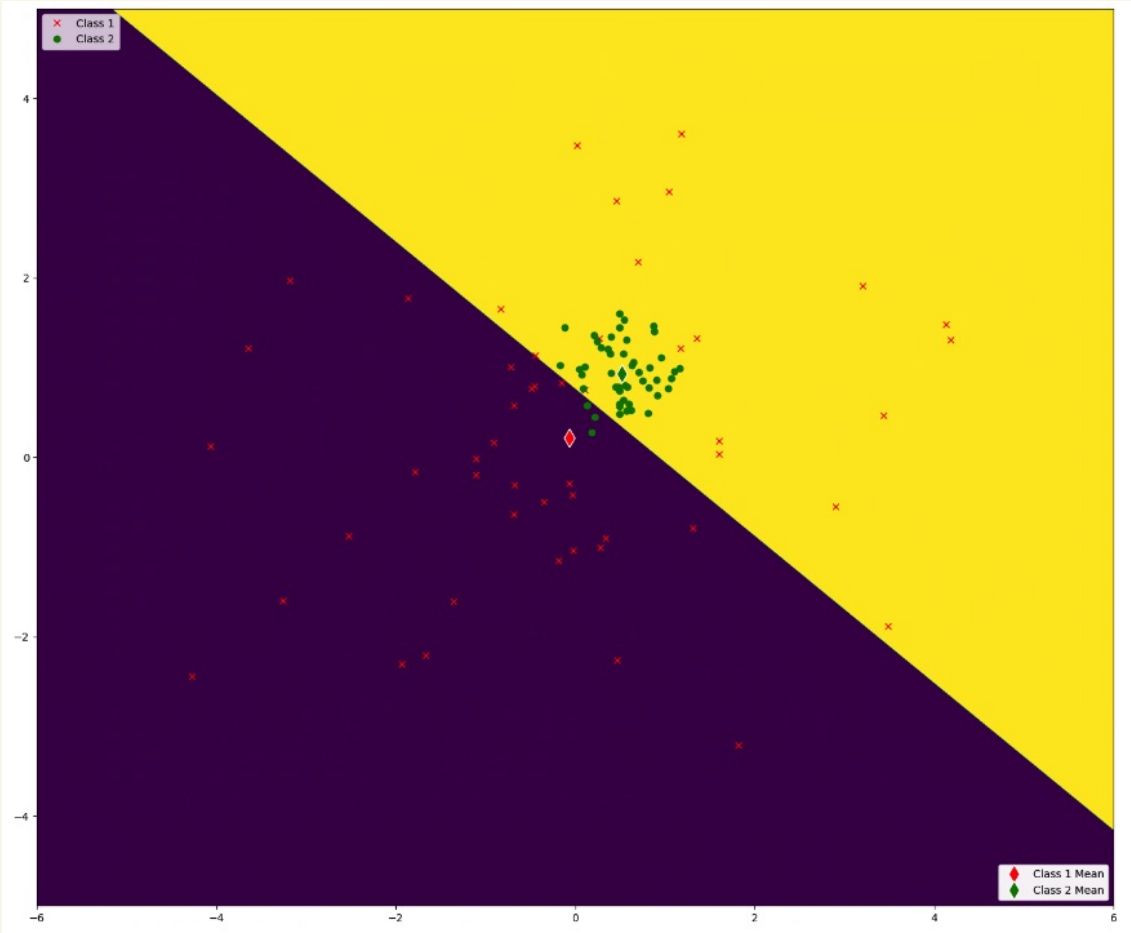
1] Train_data_1 :-



2] Train_data_2 :-



3] Train_data_3 :-



c] Classification error rate on the training set and test set :-

1] Train_data_1 :-

Classification Error Rate for the training set of dataset-1 is: 0.0%
Accuracy for the training set of dataset-1 is: 100.0%

2] Test_data_1 :-

Classification Error Rate for the testing set of dataset-1 is: 0.0%
Accuracy for the testing set of dataset-1 is: 100.0%

3] Train_data_2 :-

Classification Error Rate for the training set of dataset-2 is: 17.0%
Accuracy for the training set of dataset-2 is: 83.0%

4] Test_data_2 :-

Classification Error Rate for the testing set of dataset-2 is: 26.0%
Accuracy for the testing set of dataset-2 is: 74.0%

5] Train_data_3 :-

Classification Error Rate for the training set of dataset-2 is: 23.0%
Accuracy for the training set of dataset-2 is: 77.0%

6] Test_data_3 :-

Classification Error Rate for the testing set of dataset-3 is: 23.0%
Accuracy for the testing set of dataset-3 is: 77.0%

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

From the previous page, it is clear that test error on data set - 1 is 0.1., classification error rate on test data of dataset - 2 is 26.1. whereas on test data of dataset - 3 it is 23.1.

Classification Error Rate (CER) :-

CER is very low (or) zero for test - 1

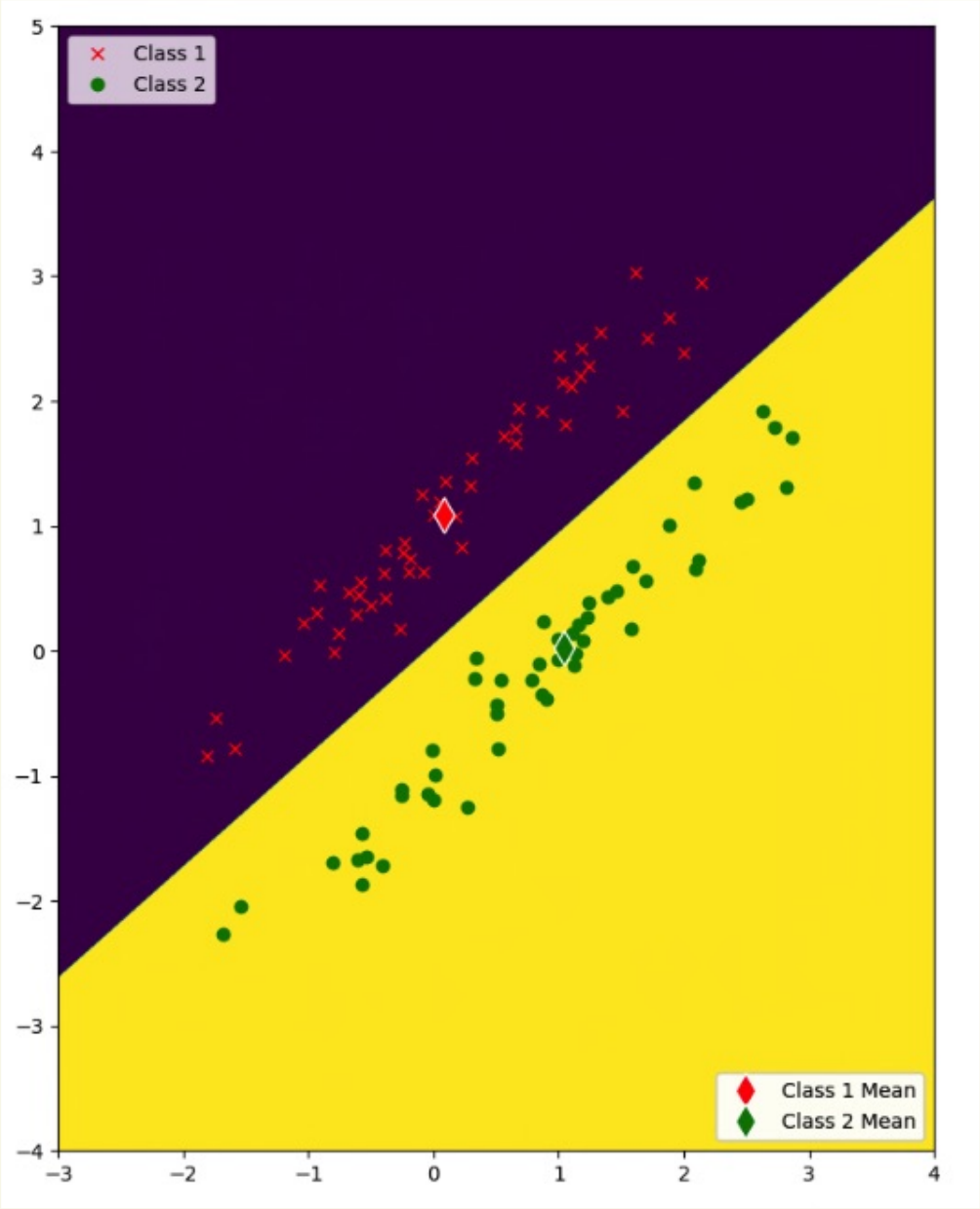
CER is high for test - 2

CER is the highest for test - 3

It's evidently seen that, the data-points in test - 1 are linearly separable and they lie exactly on either side of the perpendicular bisector of the line joining the two sample means. Hence, CER is zero for test - 1

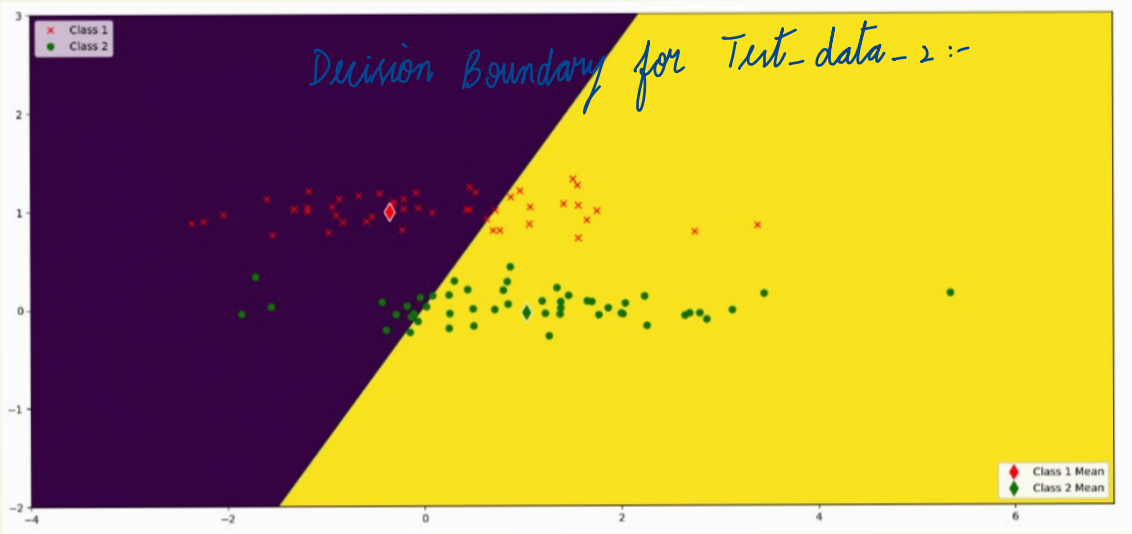
The data-points corresponding to class - 1 lie closer to the sample mean 1 and away from the sample mean - 1.

Decision Boundary for Test_data_1:-

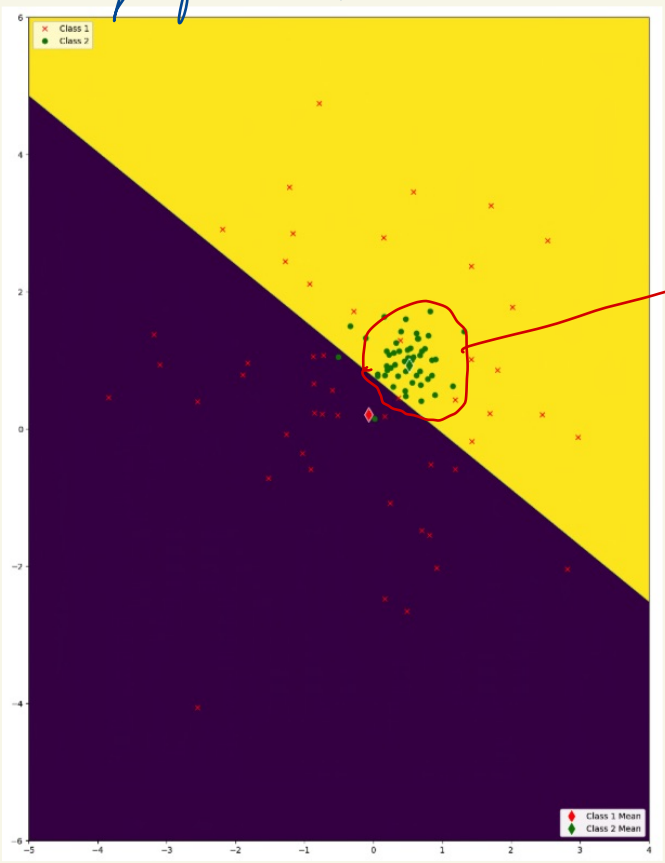


It's also clearly seen that data-points in test-2 are also linearly separable but the data points don't lie exactly on either side of the decision boundary. Also some data-points corresponding to class-1 lie very far away from sample mean-1 and little closer to sample mean-2 and vice versa.

Hence, the CER is very high according to 26% error.



Decision Boundary for Test_data_3 :-



Almost all of the data-points corresponding to label-2 lies very close to the sample mean 2 (Reason-1)

From the above plot, we can infer that, the data-points are not linearly separable and we cannot expect the "Nearest Mean Classifier" which is a linear classifier to perform well on a dataset like this. Hence, the CER is high according to 23.1. error.

We can clearly see, from "Reason 1", the CER is little lesser than the CER for Test_data_2 even though test_data_2 is linearly separable.

The CER of 23.5. arises because some data-points corresponding to label-1 are pretty much scattered and don't lie close to sample mean 1.

(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.

Train_data_1 :-

```
--------------------------------------------------
  Shape of Input Data: (100, 3)
  Number of Data Points: 100
  Number of Input Features: 2
  Number of Target Classes: 2
--------------------------------------------------
------------------------------------------------------------------
  Shape of X_mean: (2,)
  Shape of X_std: (2,)
  Mean of X along columns is: [0.56510868 0.55475647]
  Standard Deviation of X along columns is: [1.17098977
1.18285787]
  Input Train Data, X_train has been Standardized successfully!
------------------------------------------------------------------
--------------------------------------------------
  Shape of sample_means: (2, 2)
  Sample Means:
[[-0.40664534  0.45213344]
 [ 0.40664534 -0.45213344]]
--------------------------------------------------
```

Train_data_2 :-

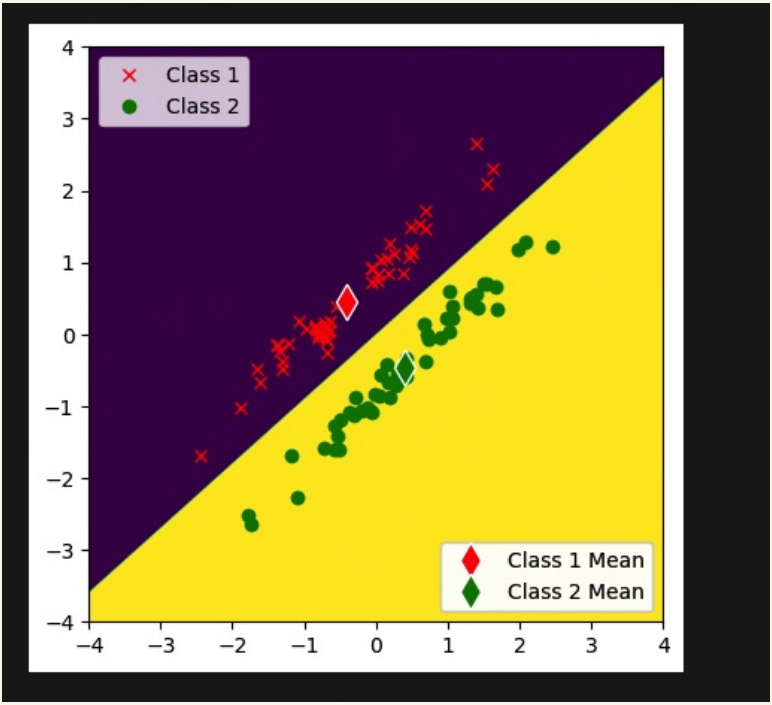
```
--------------------------------------------------
  Shape of Input Data: (100, 3)
  Number of Data Points: 100
  Number of Input Features: 2
  Number of Target Classes: 2
--------------------------------------------------
------------------------------------------------------------
  Shape of X_mean: (2,)
  Shape of X_std: (2,)
  Mean of X along columns is: [0.34146344 0.47906555]
  Standard Deviation of X along columns is: [1.29941135
0.53325431]
  Input Train Data, X_train has been Standardized successfully!
------------------------------------------------------------
--------------------------------------------------
  Shape of sample_means: (2, 2)
  Sample Means:
[[-0.53490724  0.95882365]
 [ 0.53490724 -0.95882365]]
--------------------------------------------------
```

Train_data_3 :-

```
--------------------------------------------------
  Shape of Input Data: (100, 3)
  Number of Data Points: 100
  Number of Input Features: 2
  Number of Target Classes: 2
--------------------------------------------------
------------------------------------------------------------------
  Shape of X_mean: (2,)
  Shape of X_std: (2,)
  Mean of X along columns is: [0.22745613 0.57295709]
  Standard Deviation of X along columns is: [1.43055836
1.17906206]
  Input Train Data, X_train has been Standardized
successfully!
------------------------------------------------------------------
--------------------------------------------------
  Shape of sample_means: (2, 2)
  Sample Means:
[[-0.2061046  -0.30508577]
 [ 0.2061046   0.30508577]]
--------------------------------------------------
```

Plotting the Decision Boundary for the standardized Training Data :-

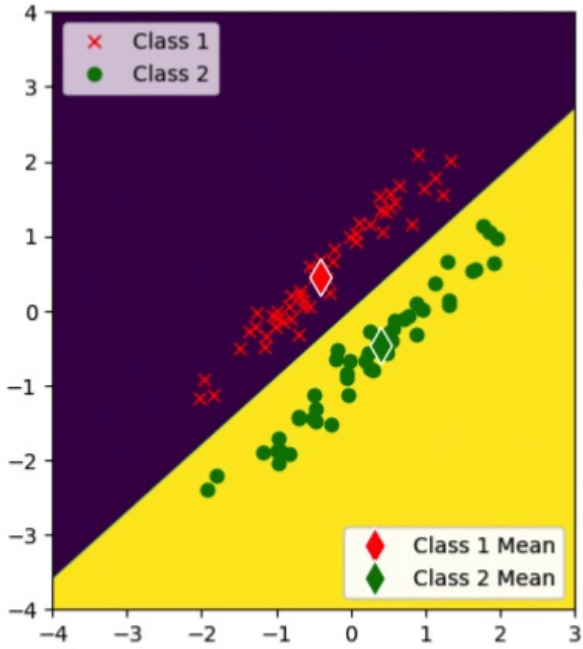
Train_data_1 :-



Classification Error Rate for the standardized training set of dataset-1 is: 0.0%
Accuracy for the standardized training set of dataset-1 is: 100.0%

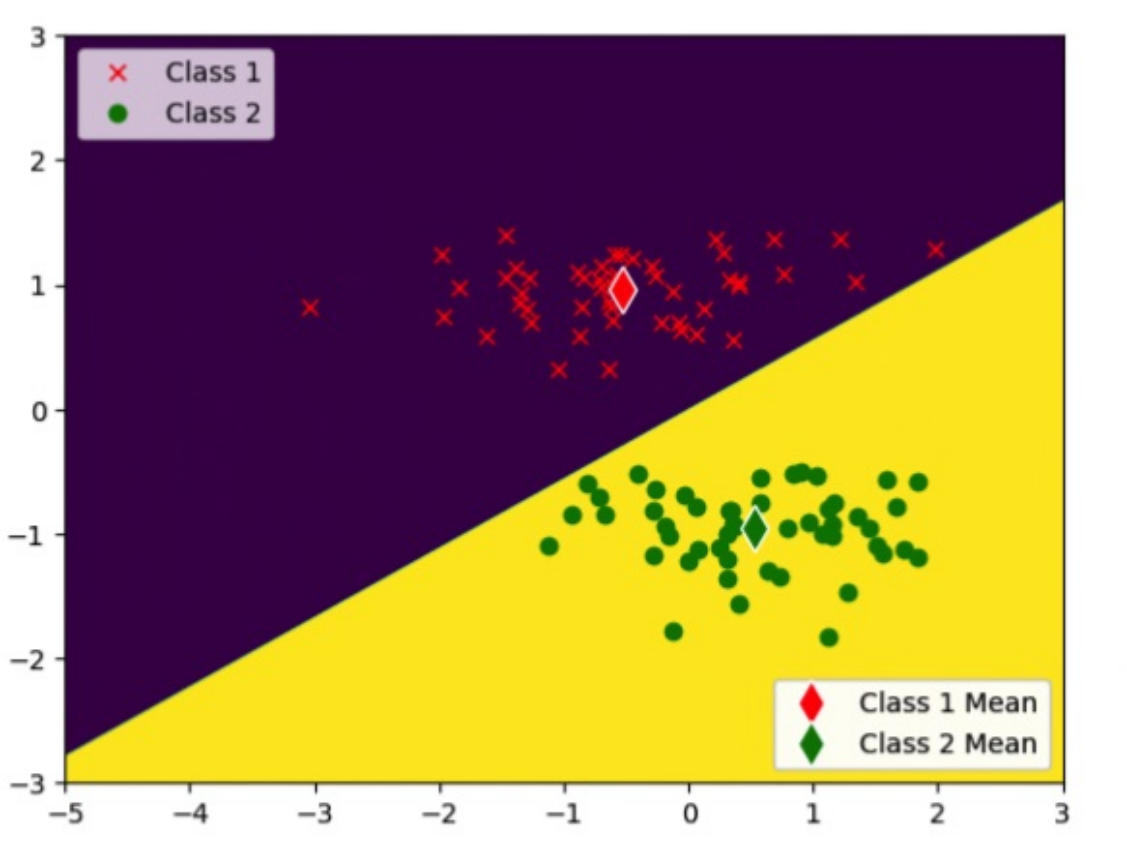
Test_data_1 :-

```
------------------------------------------------------------
  Input Test Data, X_test has been Standardized successfully!
------------------------------------------------------------
```



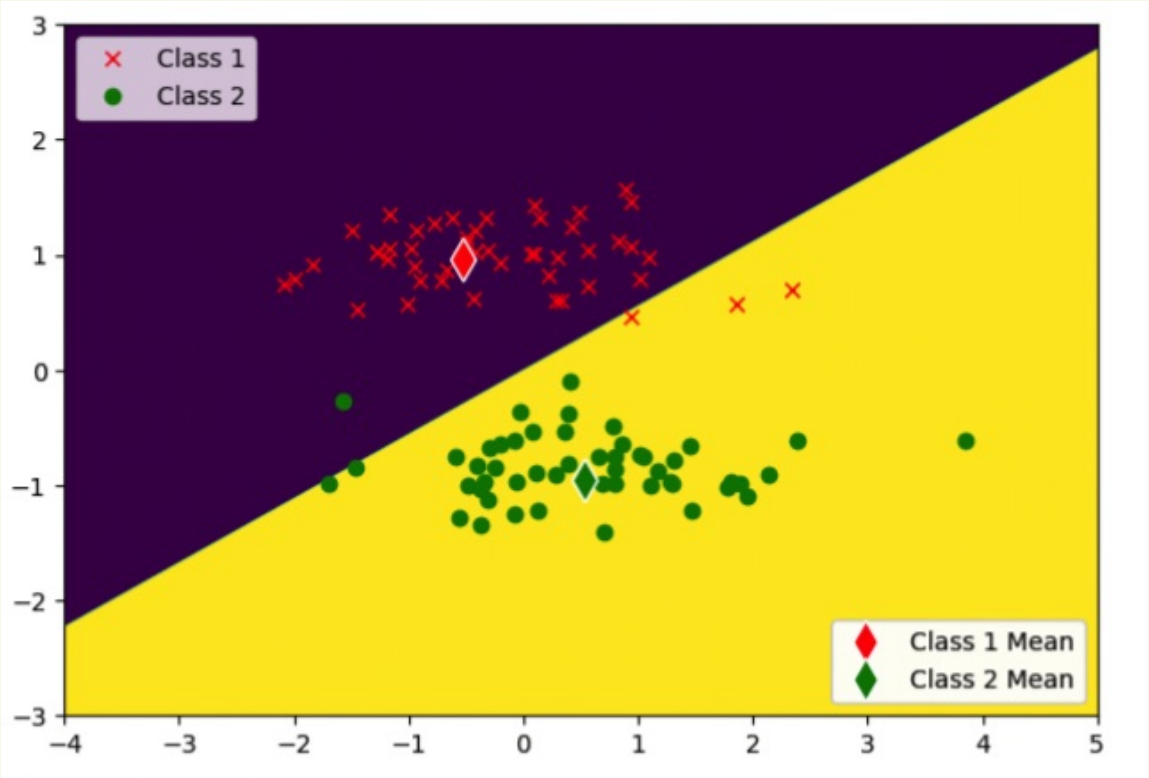

```
Classification Error Rate for the standardized testing set of dataset-1 is: 0.0
Accuracy for the standardized testing set of dataset-1 is: 100.0
```

Train_data_2 :-



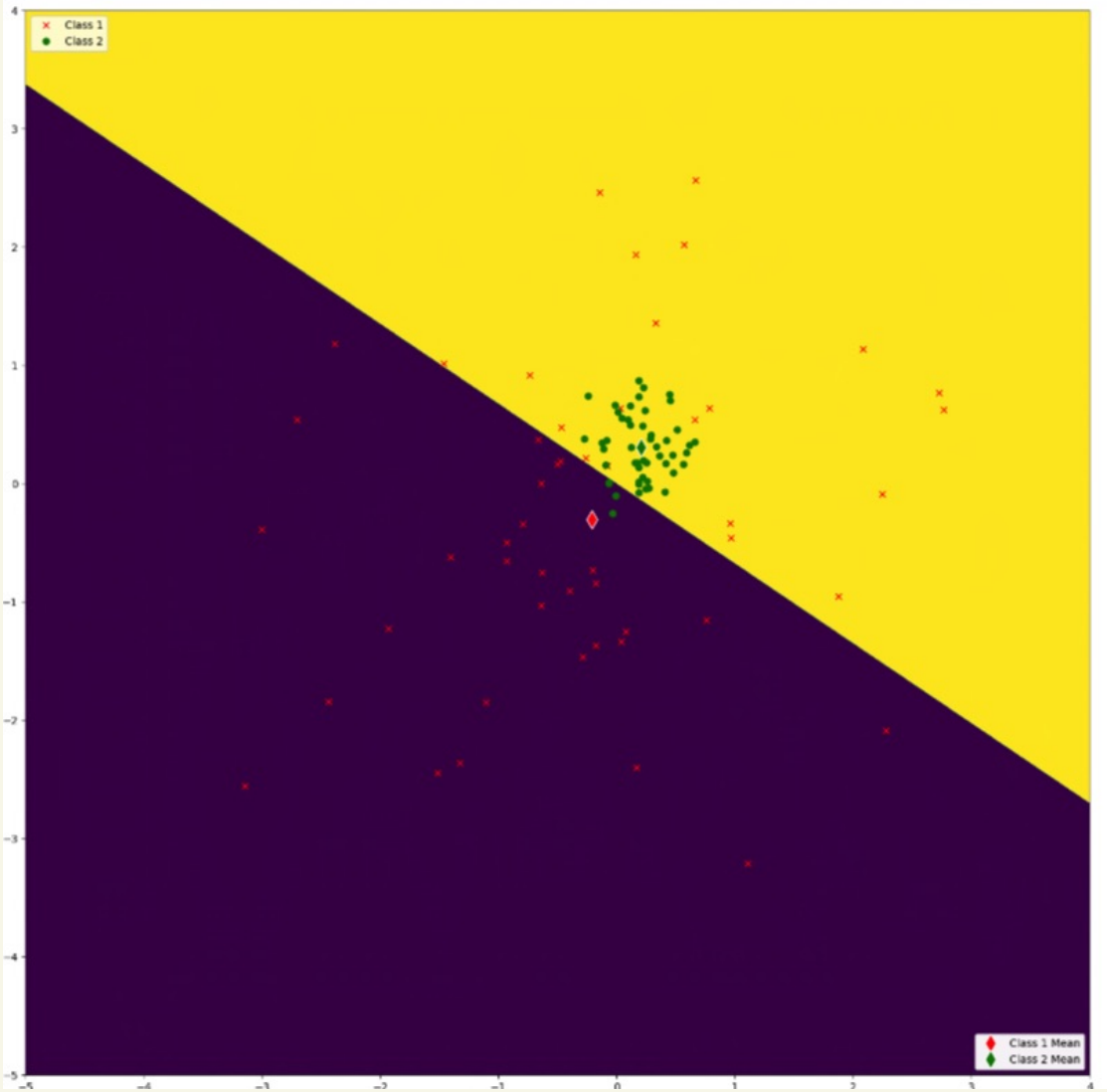
Classification Error Rate for the training set of dataset-2 is: 0.0
Accuracy for the training set of dataset-2 is: 100.0

Test_data_2 :-



Classification Error Rate for the testing set of dataset-2 is: 4.0
Accuracy for the testing set of dataset-2 is: 96.0

Train_data_3 :-



Classification Error Rate for the training set of dataset-3 is: 24.0
Accuracy for the training set of dataset-3 is: 76.0

Test_data_3 :-

Classification Error Rate for the testing set of dataset-3 is: 21.0
Accuracy for the testing set of dataset-3 is: 79.0

How standardization of input data helps us ?

- It helps to ensure that no single feature has too much influence on the distance measurements between samples, which is important for the nearest means classifier which is based on distance calculations.

- Standardizing the data also helps to ensure that all features are on the same scale, which can improve the performance of the classifier.

- Standardizing the data can also improve the numerical stability of the distance calculations, which can be important when working with large datasets.

- The standardization process involves subtracting the mean and dividing by the standard deviation of the data, this can help to reduce the impact of outliers and to make the data more symmetric.

- Since nearest mean classifier relies on distance based measurement, standardizing the data helps in accurate distance measurement between two points.

(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.

Inference :

Classification Error Rates for Unnormalized Data :

Train_1 : 0.1

Test_1 : 0.1

Train_2 : 17.1

Test_2 : 26.1

Train_3 : 23.1

Test_3 : 23.1

Classification Error Rates for Standardized data :

Train_1 : 0.1

Test_1 : 0.1

Train_2 : 0.1

Test_2 : 4.1

Train_3 : 24.1

Test_3 : 23.1

Data Set - 1 :-

It's clear that, the CER for both unnormalized standardized data of both Train and Test datasets account to 0.1. error.

This is because already the data points lied closer around the corresponding sample_means. There weren't any outliers in both Train and Test data and hence standardization doesn't improve the accuracy by any means. Still it is a best practice to standardize the data.

Data Set → 2 :-

Train Data :-

The Classification Error Rate (CER) has drastically decreased from 17.1% to 0.1%. This is because standardizing the data made the outliers little closer to the corresponding sample means thereby all the data-points are correctly classified and the data became symmetrical.

Test Data :-

The Classification Error Rate (CER) has decreased from 26.1% to 4.1% The reason is same as that stated for the training data. We get a few misclassified data points because a few data points lie so far away from the mean such that even standardization couldn't make these outliers move closer towards the mean.

Dataset → 3

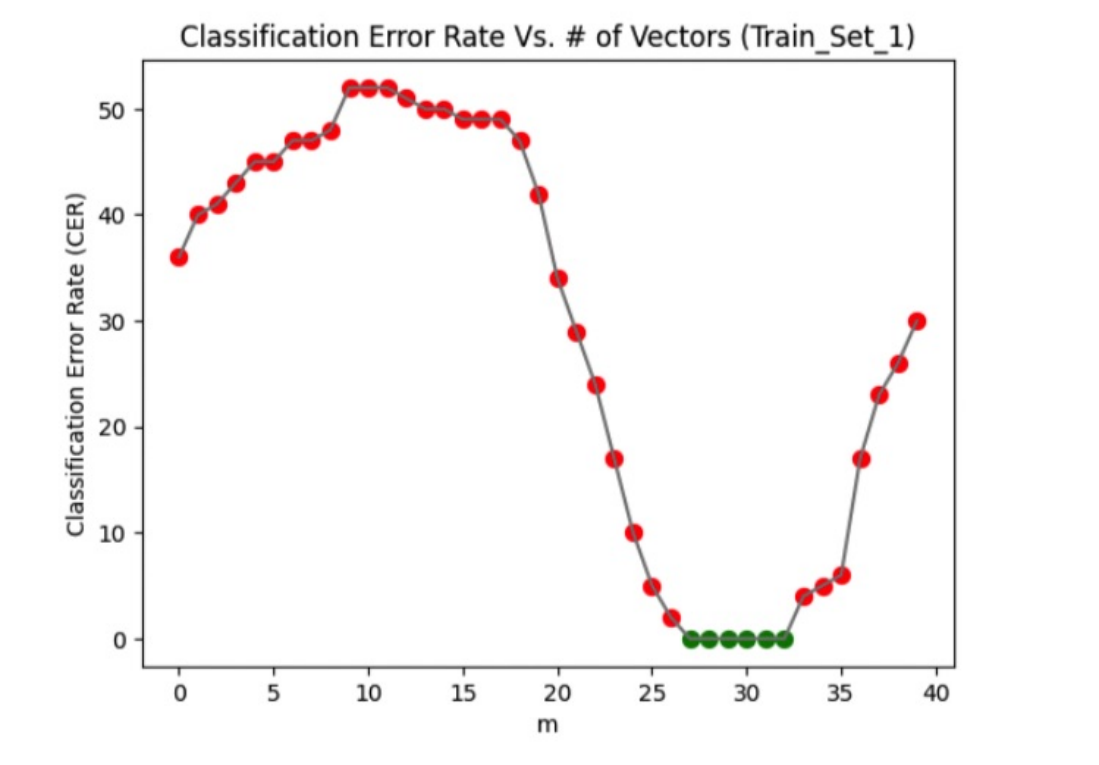
Train and Test data :-

We know that, both train and test data of dataset → 3 are not linearly separable. Standardization works beautifully well for linearly separable data. It's not a hard and fast rule that standardizing the data always accounts to improvement in accuracy. In cases of not linearly separable data, the CER tends to either stay the same or may increase in some cases. Hence, standardizing the data doesn't help us with train data but incase of test data it has accounted in decreasing the error rate to 21%.

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.*

Dataset → 1

Training Error Vs. m



Values of m for which train error is the lowest :

Values of m : 27, 28, 29, 30, 31, 32

Report μ_{m^*} :-

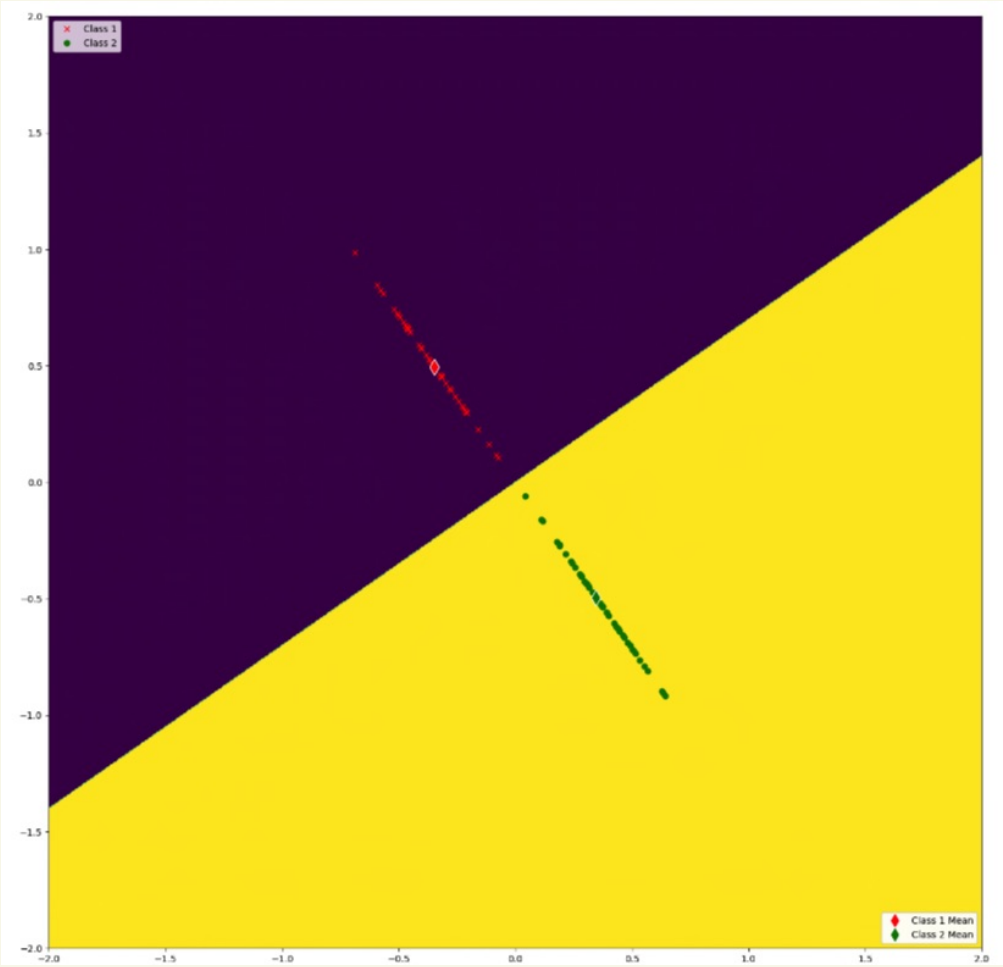
```
    print(nmc_1_std_r.m_star)
    print(nmc_1_std_r.rm_star)
    print(nmc_1_std_r.class_means_rm_star)

    print(f"The least CER Produced at index {nmc_1_std_r.m_star} is {CER_History_1[nmc_1_std_r.m_star]}%")
    print(f"The Highest Accuracy produced at index {nmc_1_std_r.m_star} is {accuracy_History_1[nmc_1_std_r.m_star]}%")
```

✓ 0.5s Python

```
27
[-7. 10.]
[[ 0.04944867]
 [-0.04944867]]
The least CER Produced at index 27 is 0.0%
The Highest Accuracy produced at index 27 is 100.0%
```

''' $\mu_{m^*} = [-7., 10.]$

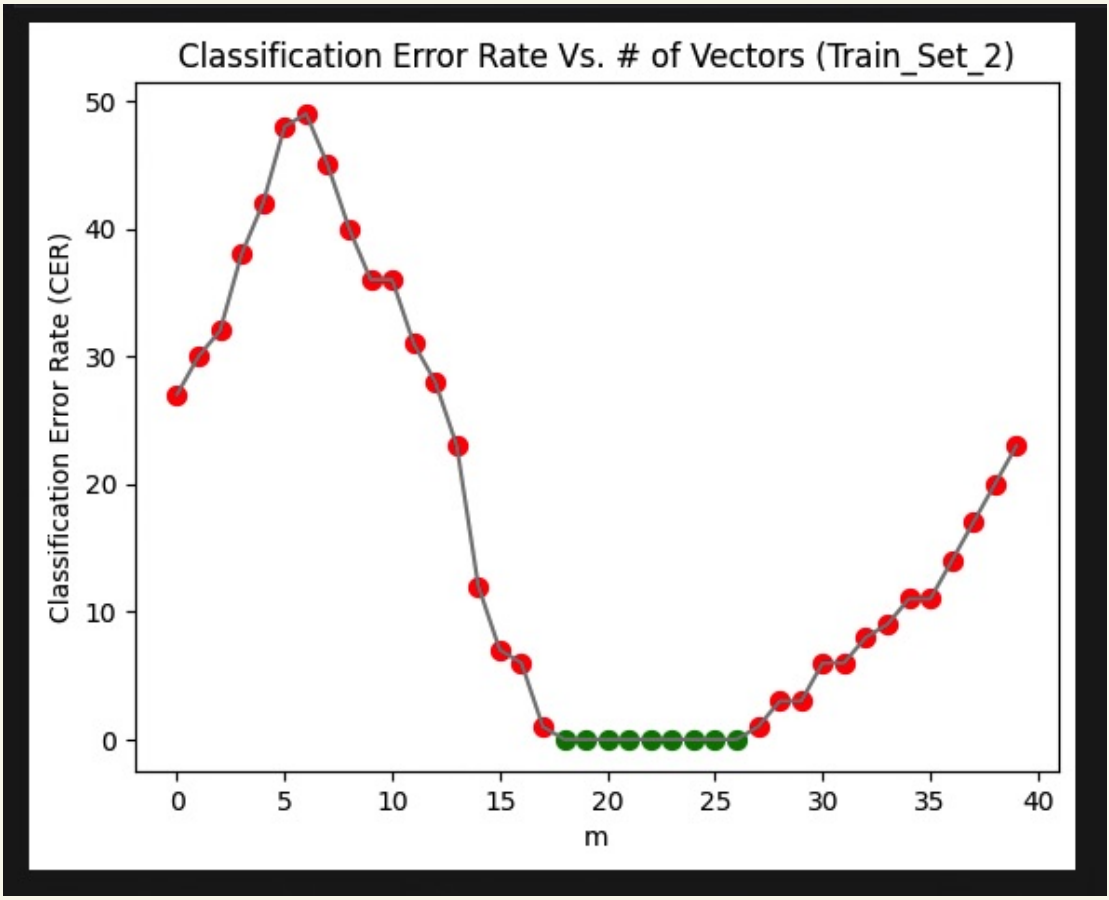
Plot and Resulting Test Error :-



The Classification Error Rate of projected test data of Dataset-1 is: 0.0
The Accuracy of projected test data of Dataset-1 is: 100.0

Dataset → 2

Training Error Vs. m for Train-data-2 :



Value of m for which Training Error is lowest ?

$m = 18, 19, 20, 21, 22, 23, 24, 25, 26$

Report r_{m^*} :-

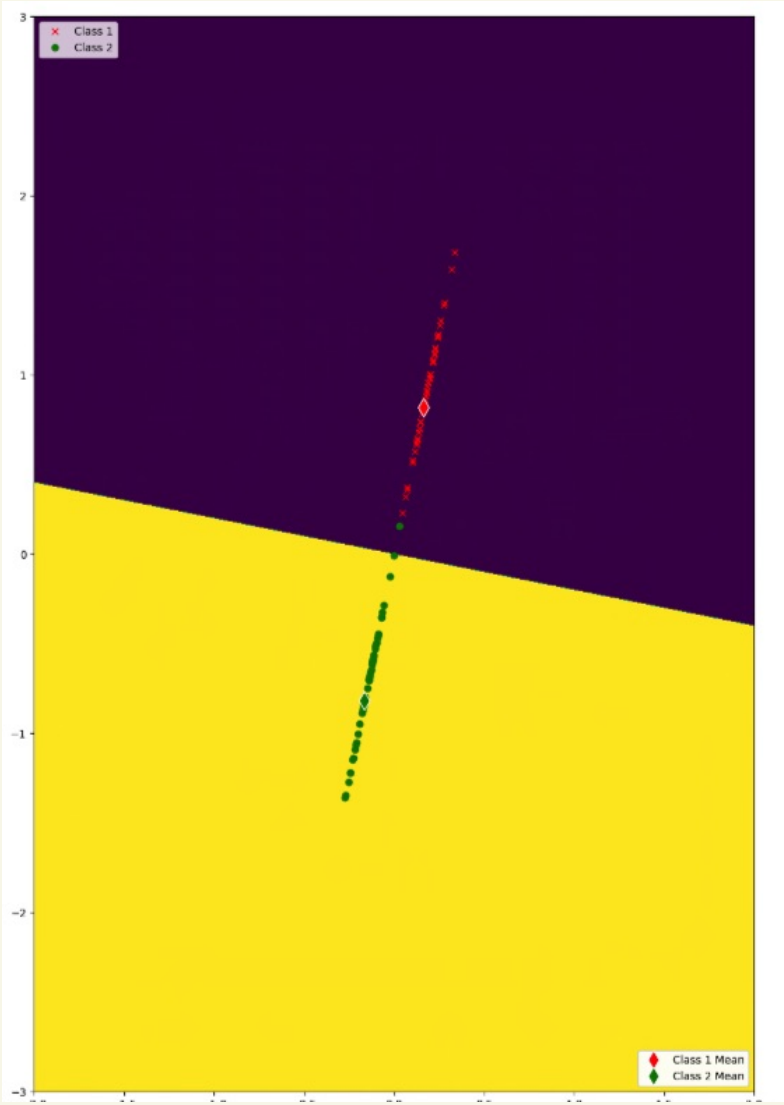
```
    print(nmc_2_std_r.m_star)
    print(nmc_2_std_r.rm_star)
    print(nmc_2_std_r.class_means_rm_star)

    print(f"The least CER Produced at index {nmc_2_std_r.m_star} is {CER_History_2[nmc_2_std_r.m_star]}%")
    print(f"The Highest Accuracy produced at index {nmc_2_std_r.m_star} is {accuracy_History_2[nmc_2_std_r.m_star]}%")
```

✓ 0.5s

```
18
[ 2. 10.]
[[ 0.0819079]
 [-0.0819079]]
The least CER Produced at index 18 is 0.0%
The Highest Accuracy produced at index 18 is 100.0%
```

$r_{m^*} = [2., 10.]$

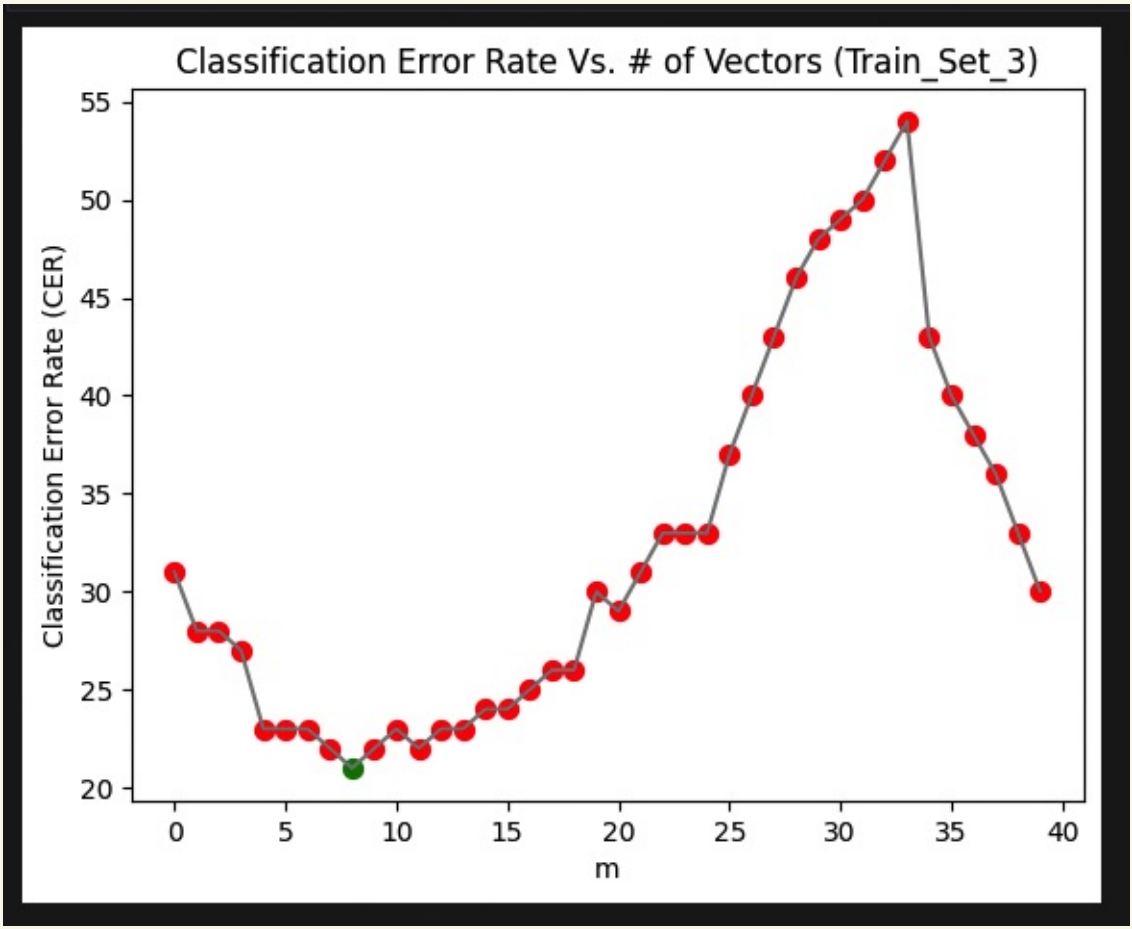
Plot and Resulting Test Error :-



The Classification " Error Rate of projected test data of Dataset-2 is: 1.0
The Accuracy of projected test data of Dataset-2 is: 99.0

Dataset → 3

Training Error Vs. m for Train_data_3



Value of m for which training error is the lowest :

$m = 8$

Report r_{m^t} :-

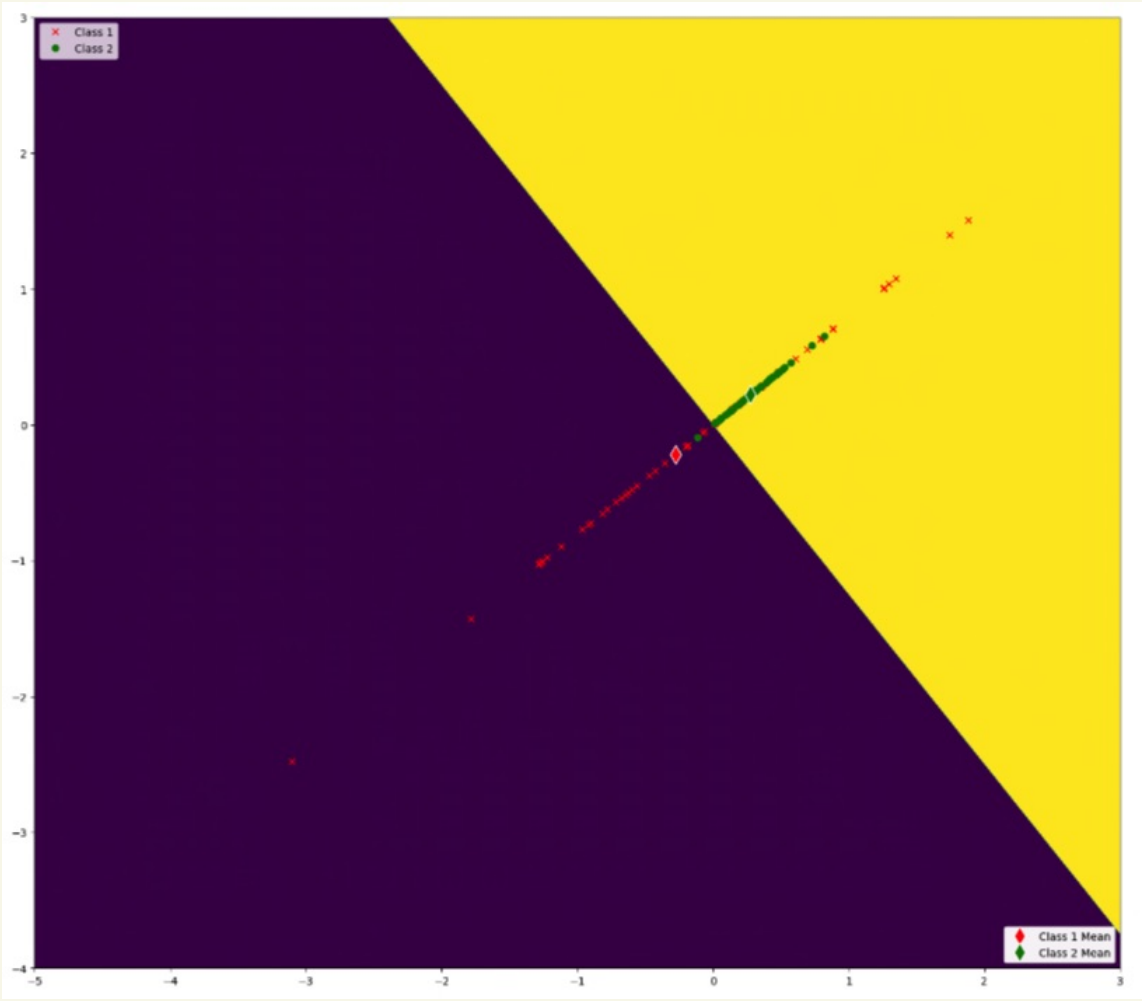
```
print(nmc_3_std_r.m_star)
print(nmc_3_std_r.rm_star)
print(nmc_3_std_r.class_means_rm_star)

print(f"The least CER Produced at index {nmc_3_std_r.m_star} is {CER_History_3[nmc_3_std_r.m_star]}%")
print(f"The Highest Accuracy produced at index {nmc_3_std_r.m_star} is {accuracy_History_3[nmc_3_std_r.m_star]}%")
```

✓ 0.5s

```
8
[10.  8.]
[[-0.02744959]
 [ 0.02744959]]
The least CER Produced at index 8 is 21.0%
The Highest Accuracy produced at index 8 is 79.0%
```

$r_{m^t} = [10., 8.]$

Plot and Resulting Test Error :-



The Classification Error Rate of projected test data of Dataset-3 is: 24.0
The Accuracy of projected test data of Dataset-3 is: 76.0

(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.

Classification Error Rate after Normalization :-

Train_1 : 0.1
Test_1 : 0.1

Train_2 : 0.1
Test_2 : 4.1

Train_3 : 24.1
Test_3 : 21.1

Classification Error Rate after Feature Reduction :-

Train_1 : 0.1
Test_1 : 0.1

Train_2 : 0.1
Test_2 : 1.1

Train_3 : 21.1

Test_3 : 24.1

Train Data and Test Data of Dataset → 1

We can see even after feature reduction, the CER stays at 0.50, no drastic changes in error rate when compared to standardized data Dataset → 1 is a piece of cake for Nearest Means Classifier.

Train Data and Test Data of Dataset → 2 :

After performing Standardization, the CER of training data reached 0.50, even after feature reduction, it remains the same.

Incase of Test Data, after standardization, error rate reduced to value of 4.1.

After performing feature reduction, the performance improved further resulting in a error rate of 1.1.

Train Data and Test Data for Dataset → 3

Dataset - 3 is a linearly non-separable data. Hence, standardization didn't do any justice to the training data resulting in an increase in error rate to 24%.

After performing feature reduction, Train Data error reduced to 21%.

Incase of Test data, the performance became worse where CER increased from 21% to 24%.

Nearest Means Classifier doesn't perform well on Dataset → 3