EECE 5644 Assignment 1

Ishaan Desai (desai.is@northeastern.edu)

October 16th, 2025

Professor Erdogmus

GitHub Link:

https://github.com/ishaandesai0/EECE-5644/tree/main/Assignment%201

Question 1

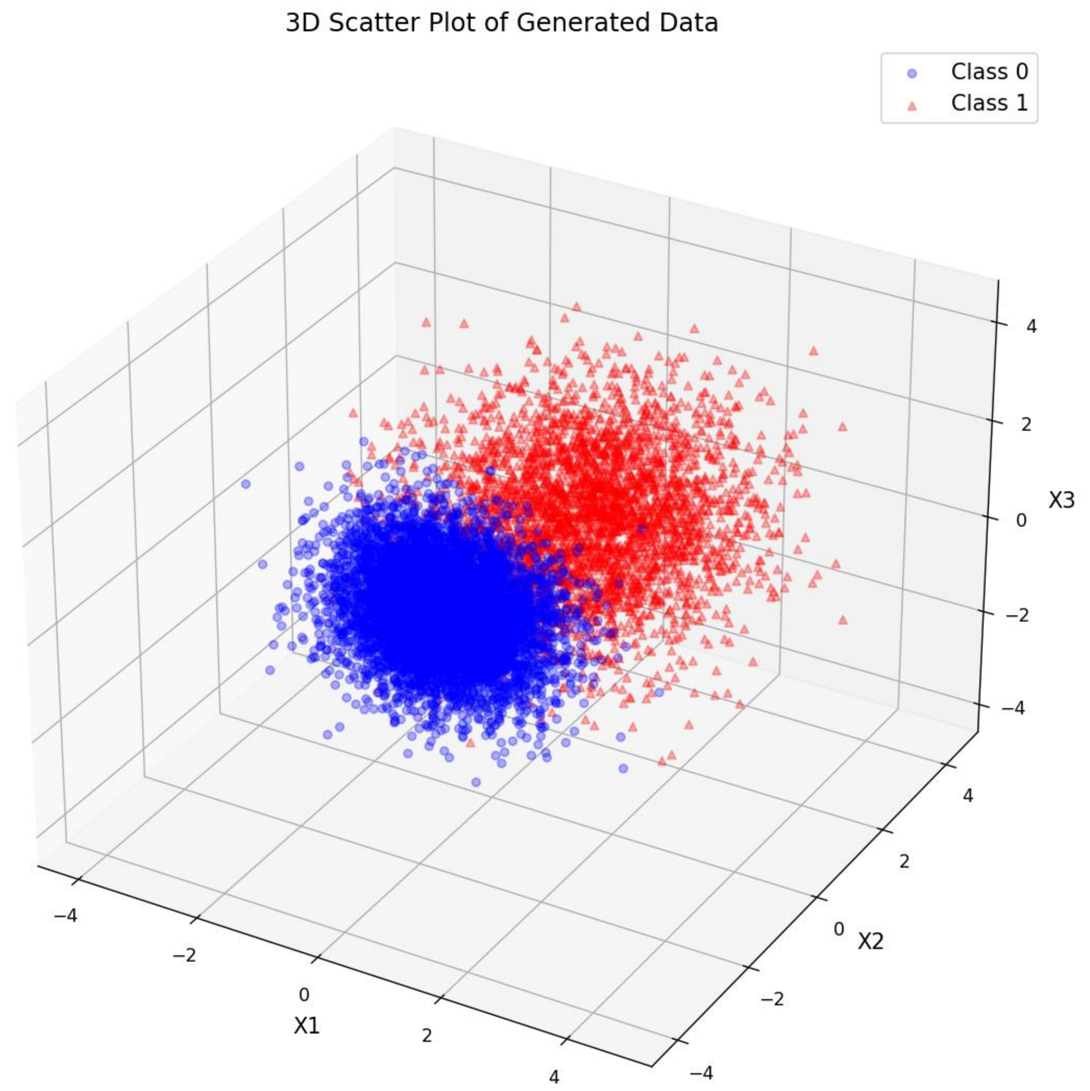

Figure 1: 10,000 sample values from provided Gaussian PDF Distribution

The first part of this question had 10,000 samples from the provided PDF generated. In order to randomly generate, I used the random library from NumPy in Python. The probability of getting a sample from Class 0 was 65% and Class 1 35%. These numbers were stated in the problem, and the sample data used for this question can be seen above in Figure 1.

1.A Expected Risk Minimization Classification

The minimum expected risk classification rule is based on the likelihood ratio test. For binary classification, the optimal decision rule is:

Decide L = 1 if: $p(x|L=1)/p(x|L=0) > y$

Decide L = 0 otherwise

Where the threshold y is determined by the class priors and loss values:

$Y = [P(L=0)/P(L=1)] \times [(\lambda_{01} - \lambda_{00})/(\lambda_{10} - \lambda_{11})]$

The loss function represents the cost of deciding class I when the true class is j:

λ_{ij}	Interpretation
λ_{00}	True Negative
λ_{01}	False Negative
λ_{10}	False Positive
λ_{11}	True Positive

Table 1: Loss Function Components

For 0-1 loss:

- $\lambda_{00} = \lambda_{11} = 0$
- $\lambda_{01} = \lambda_{10} = 1$

Therefore, $y = P(L = 0)/p(L = 1) = 0.65/0.35 = 1.8571$

This is the Maximum A Posteriori (MAP) decision rule, minimizing the probability of error.

The classifier was implemented by calculating the likelihood ratio for each sample and varying the threshold y from 0 to infinity. For each threshold value, I computed:

- True Positive Rate $P(D = 1 \mid L = 1)$: fraction of Class 1 Samples correctly identified
- False Positive Rate $P(D = 1 \mid L = 0)$: fraction of Class 0 samples incorrectly called Class 1
- Probability of Error: $FPR*P(L=0) + (1 – TPR) * P(L = 1)$

The ROC curve traces the FPR and TPR operating points as Y varies:

- At $y=0$: Always decide Class 1
- At $y=\infty$, Always decide Class 0

Figure 2 shows the ROC curve below:

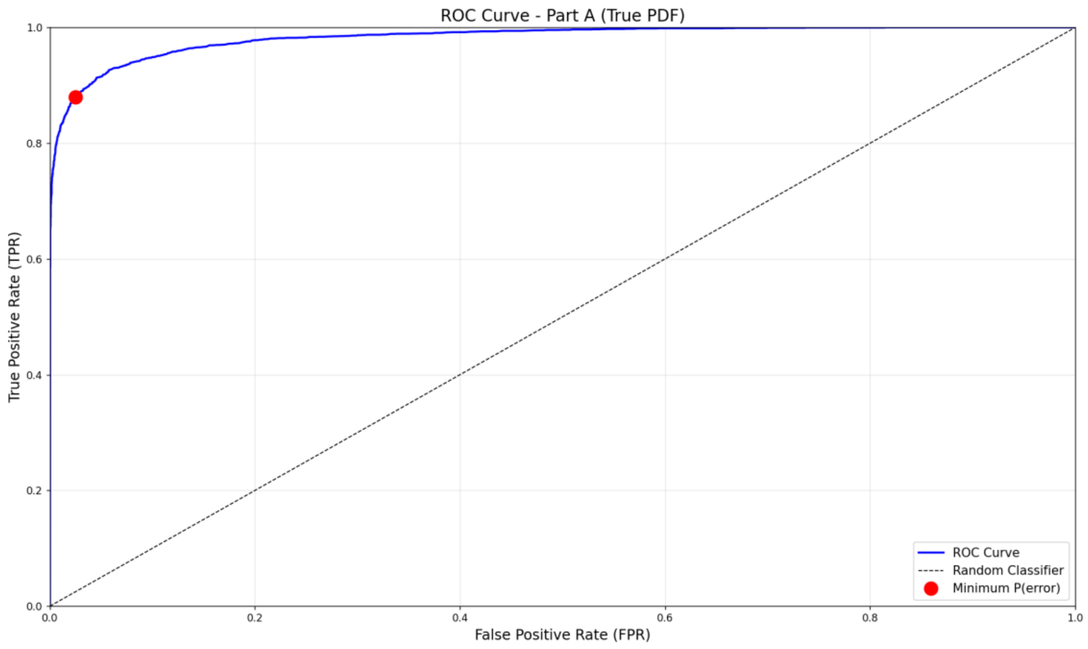

Figure 2: ROC Curve for Part A

This is the ROC curve showing the tradeoff between true positive rate and false positive rate. The red point depicts the minimum probability of error operating point. The curve's position far above the diagonal shows an excellent classifier performance.

Table 2 below shows a more detailed comparison of theoretical and experimental results:

Type	y	P(error)	TPR	FPR
Theoretical	1.8571	~0.057	-	-
Experimental	2.1248	0.057	0.8802	0.0247
Difference	0.2677	0	-	-

Table 2: Comparison of Results

The experimental optimal threshold is slightly more than the theoretical value (0.2677). This discrepancy comes from finite sample effects. With so many samples, the experimental likelihood ratios will approximate but not necessarily match the true distribution.

The minimum probability of error achieved is 5.7%, representing the Bayes error rate. Classifiers can't perform better than this without additional information, as errors in the region where two Gaussian distributions overlap can occur and these samples are ambiguous.

At the optimal operating point:

- TPR = 88.02%: 88% of Class 1 samples correctly identified
- FPR = 2.47%: Only 2.5% of samples are false alarms
- This tradeoff minimizes overall error given class priors

1.B Naive Bayes Classifier

To analyze the impact of incorrect model assumptions, I implemented a Naïve Bayes Classifier. This assumes features are conditionally independent given the class label, which is incorrect for our data

Assumptions:

The Naïve Bayes classifier uses identity covariance matrices:

C0_naive = C1_naive = I3x3 = [[1, 0, 0],

[0, 1, 0],

[0, 0, 1]]

When compared to our true covariances, we can see that the true covariance matrix contains correlations capturing dependencies between features. The naïve assumption sets all correlations to 0, hence treating features as independent.

Figure 3, which is the ROC curve for the Naïve Bayes Classifier, can be seen below:

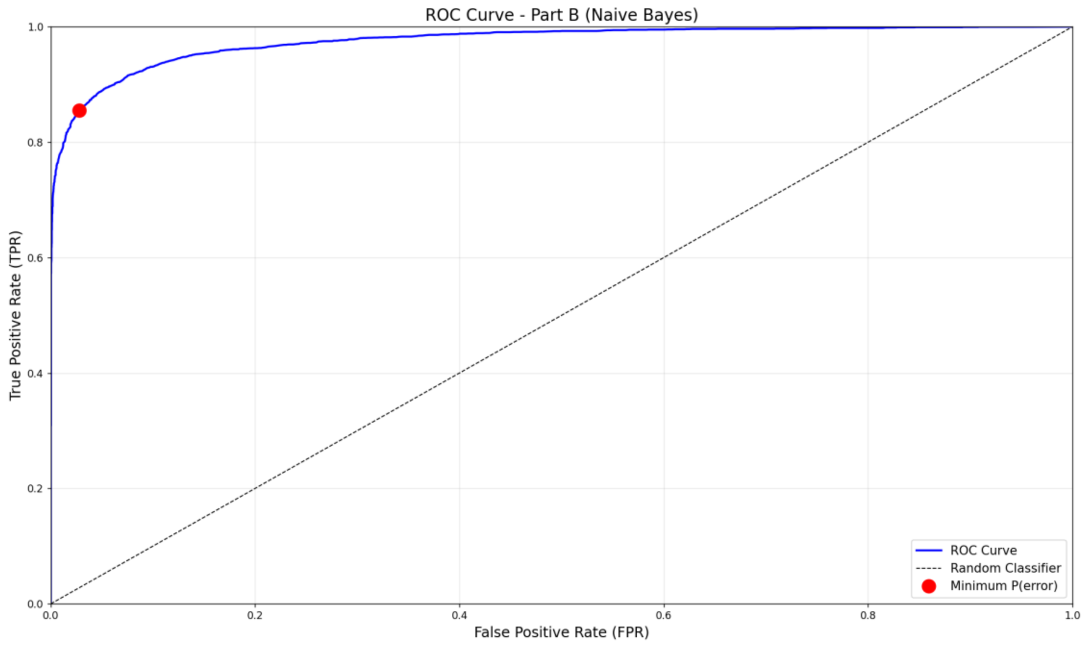

Figure 3: ROC Curve for Part B

As you can see, the curve is slightly lower than Part A, indicating worse performance due to incorrect covariance assumptions. Table 3 below shows a more detailed analysis of the results:

Metric	Value	TPR	FPR
Optimal y	1.2915	0.8552	0.0277
Minimum P(error)	0.0675	-	-

Table 3: Naïve Bayes Performance

The Naïve Bayes classifier achieves a P(error) or 6.75%. This is 1.05% worse than the optimal classifier, showing that incorrect model assumptions worsen performance. On a lighter note, the degradation is not too bad because the class means are well separated. This provides strong discriminative information. Classification also primarily depends on difference in means as opposed to correlation structure. The ignored correlations of (-0.5, 0.3) were also not too extreme. If the classes are closer together or the correlations are stronger, this Naïve Bayes assumption would've performed much worse.

1.C Fisher Linear Discriminant Analysis

Fisher Linear Discriminant Analysis takes a fundamentally different approach. Instead of modeling the full 3D distributions, it finds an optimal 1D projection that maximally separates the classes.

In order to do this, I first estimated the parameters from the data. Our estimated means were:

M0 = [-0.509, -0.479, -0.524]

M1 = [0.978, 1.005, 0.978]

These estimates were close to our true values, which validated the 10K sample dataset. Following this, the scatter matrices were calculated. S_B measures separation between class means, and was calculated as such:

$$S_b = (m_1 - m_0) * (m_1 - m_0)^T$$

S_W measures spread within each class and was the sum of the estimated covariances.

After solving the generalized eigenvalue problem to find the optimal projection, I got a w_LDA of [0.541, 0.624, 0.564] $^{T.}$ Following the projection and classification, Figure 4 shows the ROC curve I got for Fisher LDA:

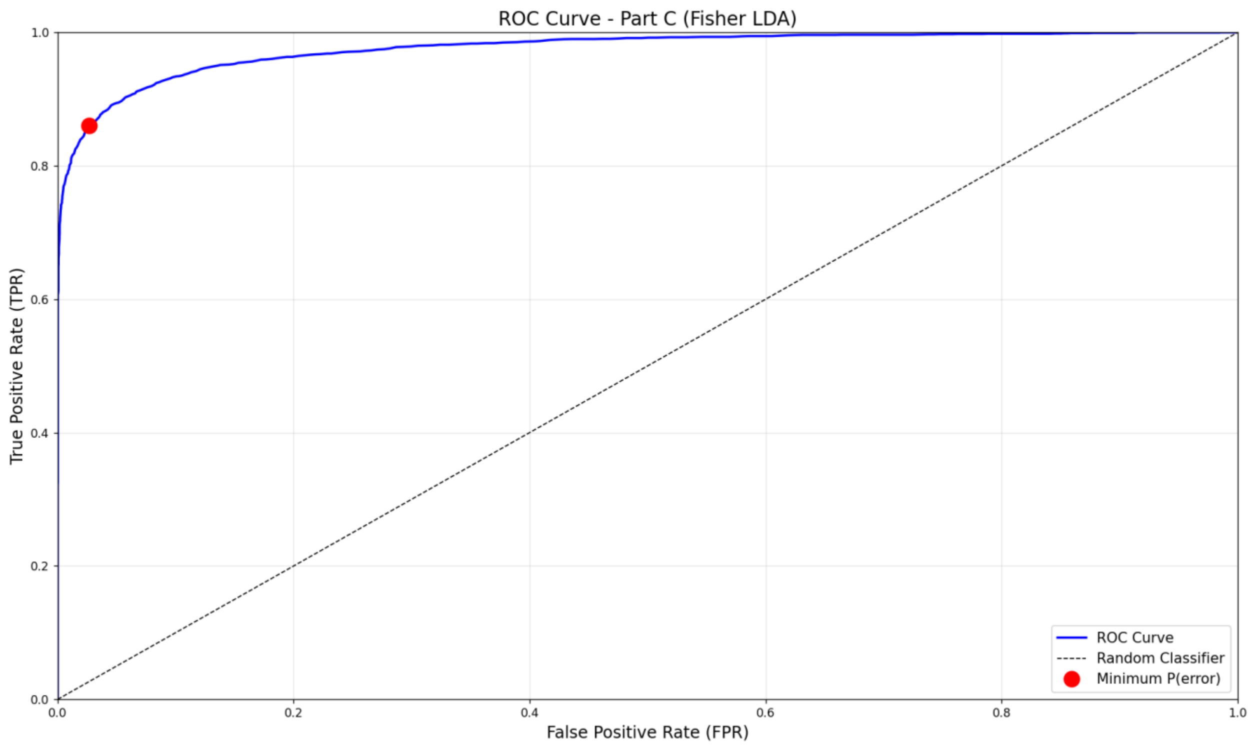

Figure 4: ROC Curve for Part C

Table 4 below shows the performance of the Fisher LDA:

Metric	Value	TPR	FPR
Optimal τ	0.4929	0.8602	0.0267
Minimum P(error)	0.0651	-	-

Table 4: Fisher LDA Performance

Fisher LDA achieves P(error) of 6.51%, which is only 0.81% worse than the optimal Bayes classifier. This shows that

- The optimal discriminant direction captured nearly all of the classification relevant information
- Dimensionality reduction incurs minimal performance loss
- Parameter estimation from samples is reliable with sufficient data
- Fisher LDA is highly effective for Gaussian data with similar covariance structures.

The projection vector w indicated that all three dimensions contribute approximately equally to optimal separation, with the second dimension weighed slightly higher.

When comparing the three methods, the True PDF achieved the theoretical minimum error rate, validating the implementation. Fisher LDA performs extremely well despite using estimated parameters and reducing dimensionality. Naïve Bayes suffered from incorrect covariance assumptions but remained competitive. The final ranking in terms of performance comes in at

1. True PDF
2. Fisher LDA
3. Naïve Bayes

All three methods achieved very high accuracy (>93%), showing that the classes are well-separated despite the overlap region producing the ~6% error rate.

Question 2

This problem involved designing a 4-class classification scenario using a mixture of four 2-dimensonal Gaussians. The layout was intentionally designed with Class 4 at the origin and overlapping it with the other three classes in a triangular arrangement. The design can be seen below, with all class priors being set to 0.25:

```
# Class 1: Top
    m1 = np.array([0, 3])
    c1 = np.array([[1.5, 0.5],
                   [0.5, 1.5]])
    # Class 2: Bottom-Left
    m2 = np.array([-3, -1.5])
    c2 = np.array([[1.2, -0.3],
                   [-0.3, 1.8]])
    # Class 3: Bottom-Right
    m3 = np.array([3, -1.5])
    c3 = np.array([[1.8, 0.4],
                   [0.4, 1.2]])
    # Class 4: Center
    m4 = np.array([0, 0])
    c4 = np.array([[2.0, 0.5],
                   [0.5, 2.0]])
```

Labels were randomly selected by prior distribution. Of the 10K samples, the breakdown can be seen below, accompanied by the 2D scatterplot of the four classes:

Class 1: 2547 samples (25.5%)

Class 2: 2529 samples (25.3%)

Class 3: 2511 samples (25.1%)

Class 4: 2413 samples (24.1%)

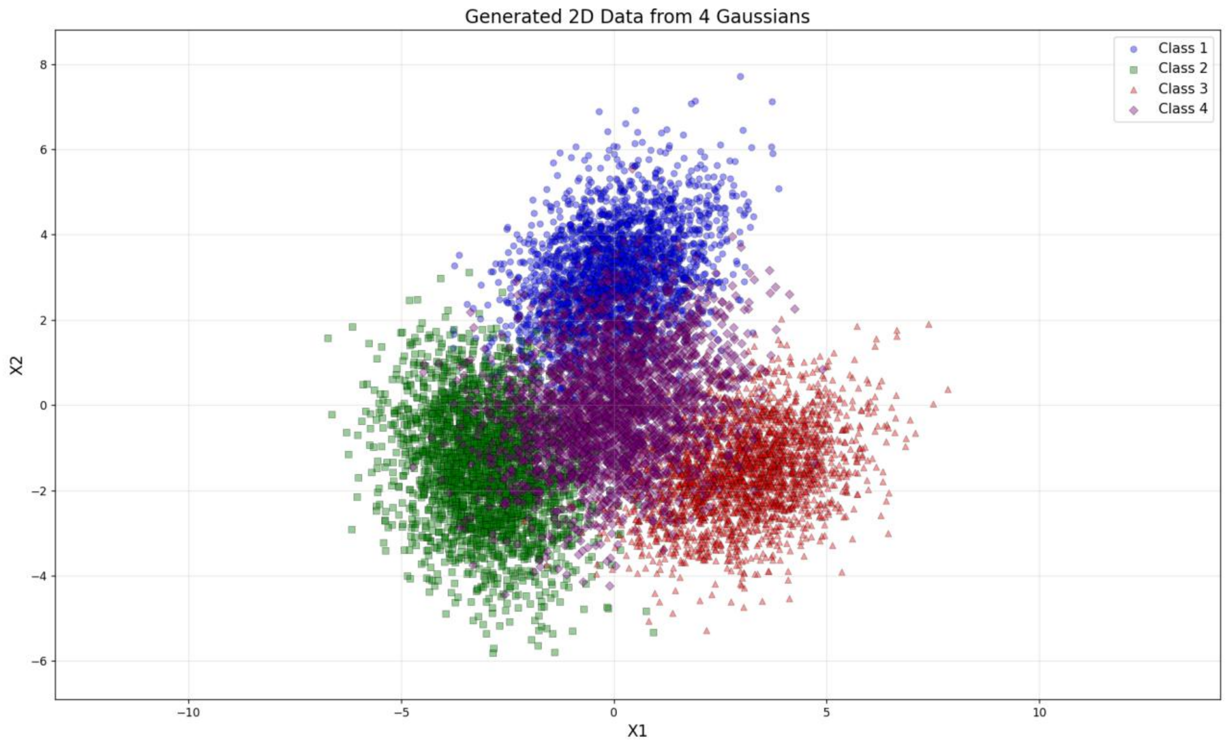

Figure 5: Data Distribution

2.A MAP Classification

Part A: Maximum A Posteriori Classification

The MAP decision rule for multi-class classification chooses the class with the highest posterior probability.

$$D = argmax_j\ P(L = j|x) = argmax_j\ [p(x|L = j)x\ P(L = j)]$$

For equal priors and 0-1 loss, this is the same as choosing the class whose Gaussian distribution assigns the highest likelihood to the observation x. The idea here is that for each sample x, the posterior for each class would be computed, normalized, then decided. Results can be seen in Table 5 below:

	L = 1	L = 2	L = 3	L = 4
D = 1	0.9034	0.0091	0.0000	0.1227
D = 2	0.0106	0.9174	0.0028	0.1181

D = 3	0.0000	0.0036	0.9307	0.0796
D = 4	0.0860	0.0700	0.0665	0.6797

Table 5: Confusion Matrix Part A

I had an overall accuracy of 85.98% with a probability of error of 14.02%. Class 1 had 90.34% accuracy, with classes 2, 3, and 4 having 91.74%, 93.07%, and 67.97% respectively. Classes 1, 2, and 3 have great accuracy since they are positioned at the periphery with minimal overlap. Class 4 has a lower accuracy because of its central position, overlapping with the other classes. Looking at the last column, Class 4 had a total 32.03% misclassification rate, occurring in the overlap regions. Figure 6 depicts the classification results:

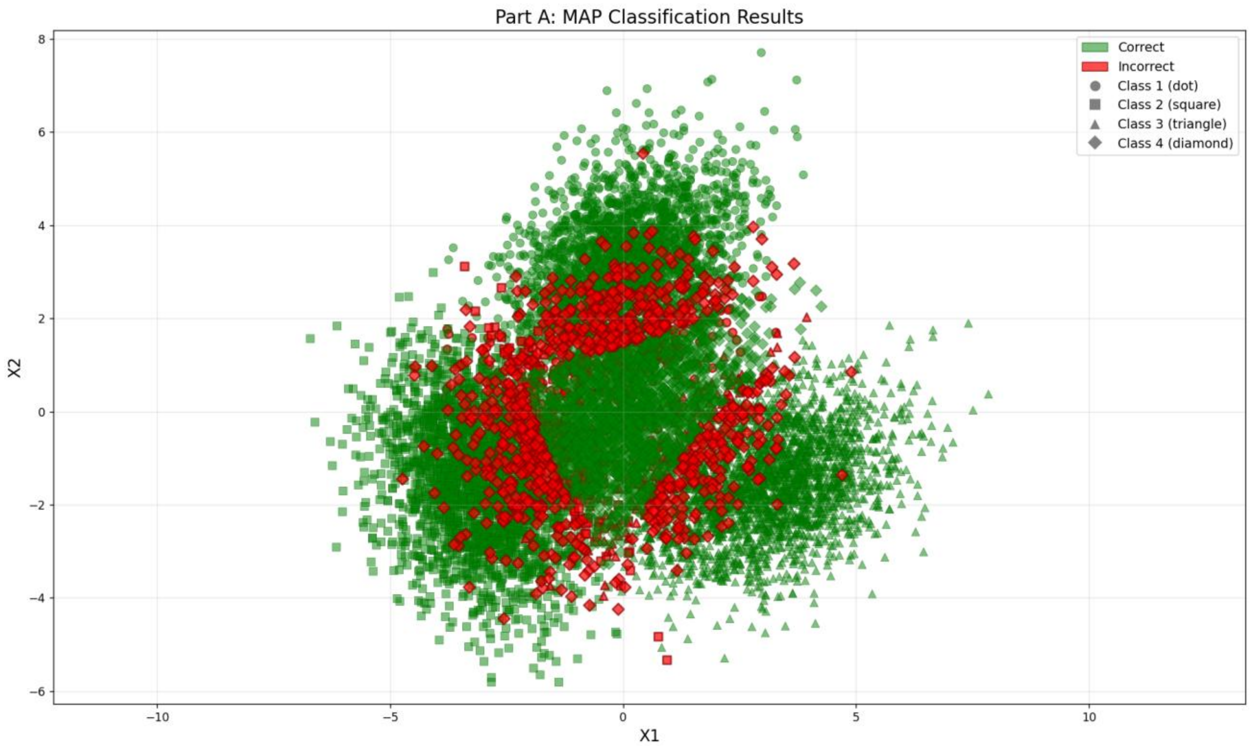

Figure 6: MAP Classification Results

Part B: Expected Risk Minimization with Asymmetric Loss

To demonstrate the impact of asymmetric loss, we modified the classification objective to heavily penalize errors on Class 4. Table 6 below shows the Asymmetric Loss Matrix:

	L = 1	L = 2	L = 3	L = 4
D = 1	0	10	10	100
D = 2	1	0	10	100
D = 3	1	1	0	100
D = 4	1	1	1	0

Table 6: Asymmetric Loss Matrix

The key feature here is that misclassifying a true Class 4 sample costs 100 units while others cost 1-10. It creates strong incentive to protect Class 4.

Results can be seen below in Table 7 and 8:

	L = 1	L = 2	L = 3	L = 4
D = 1	0.1103	0.0000	0.0000	0.0004
D = 2	0.0000	0.0522	0.0000	0.0004
D = 3	0.0000	0.0000	0.3744	0.0008
D = 4	0.8897	0.9478	0.6256	0.9983

Table 7: Confusion Matrix

Expected Risk: 0.6634

Decision	Part A (0-1)	Part B (Asymmetric)
Class 1	~25%	2.8%
Class 2	~25%	1.3%
Class 3	~25%	9.4%
Class 4	~25%	86.4%

Table 8: Decision Distribution Chart

The classifier now decided 86.4% of all samples are Class 4, which is much more than the 25% before. The extreme bias is the risk minimization strategy. If the model picks anything besides Class 4, 100-unit penalty.

Table 9 below goes deeper into the accuracy comparison:

Class	Part A	Part B	Change
1	90.34%	11.03%	-79.31%
2	91.74%	5.22%	-86.52%
3	93.07%	37.44%	-55.63%
4	67.97%	99.83%	+31.86%

Table 9: Accuracy Comparison between Parts A and B

As one can see, the Class 4 errors reduced from 32.03% to 0.17%. This came at a big cost though, as the accuracy for Classes 1, 2, and 3 dropped significantly. Overall accuracy was sacrificed to protect Class 4.

While Part B's expected risk appears higher than Part A's (0.1402), they measure different objectives. Part A is minimizing unweighted error count, while Part B is minimizing weighted risk with asymmetric penalties. With Part B's loss matrix, the Class 4 decision rate of 86% is optimal because avoiding the 100-unit penalty justifies the increased 1-unit penalties on other classes. The Part B results are visualized and can be seen below in Figure 7:

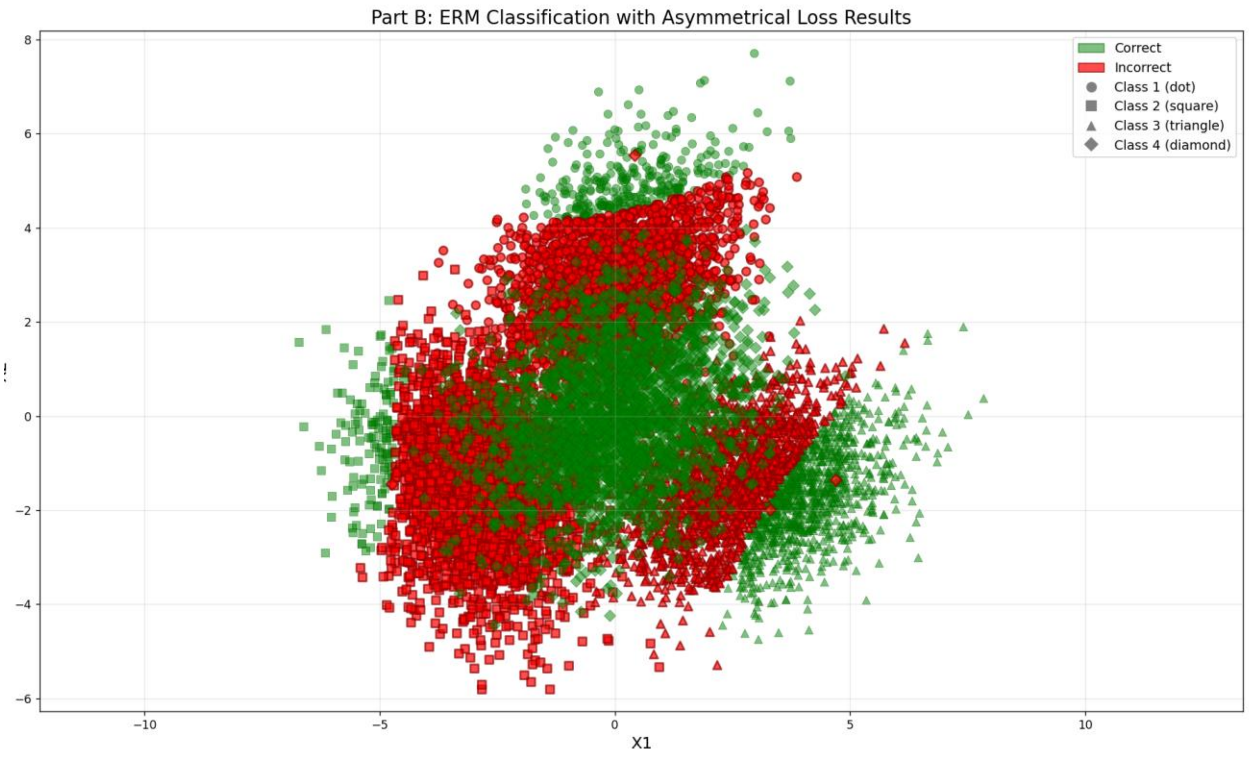

Figure 7: Part B Visualization

Question 3

This question had me apply Gaussian classification models to two real-world datasets from the UCI ML Repository to evaluate whether the Gaussian assumption holds for practical problems.

The method applied for this problem went as follows:

- Load datasets
- Estimate class-conditional parameters: means, covariances, priors
- Apply covariance regularization: C_reg = C_sample + λI
- Classify using MAP rule
- Compute confusion matrices and accuracy
- Visualize high-dimensional data with PCA
- Discuss appropriateness of Gaussian model

Both datasets produced ill-conditioned covariance matrices requiring regularization. To do this, I used the following equation:

$$\lambda \; = \; \alpha \; x \; mean(positive \; eigenvalues$$

where α = 0.01. This made sure all eigenvalues of C_reg are greater than λ, preventing numerical instability during matrix inversion.

3.1 Wine Quality Dataset

Characteristics:

- UCI ML Repository (White Wine only)
- Samples: 4,898
- Features: 11
- Classes: 7 quality scores
- Distribution:
    - Quality 6: 2,198 samples
    - Quality 5: 1,457 samples
    - Quality 7: 880 samples
    - Quality 3: 20 samples
    - Quality 9: 5 samples

The 2D and 3D Wine Quality PCA visualizations can be seen below with Figures 8 and 9:

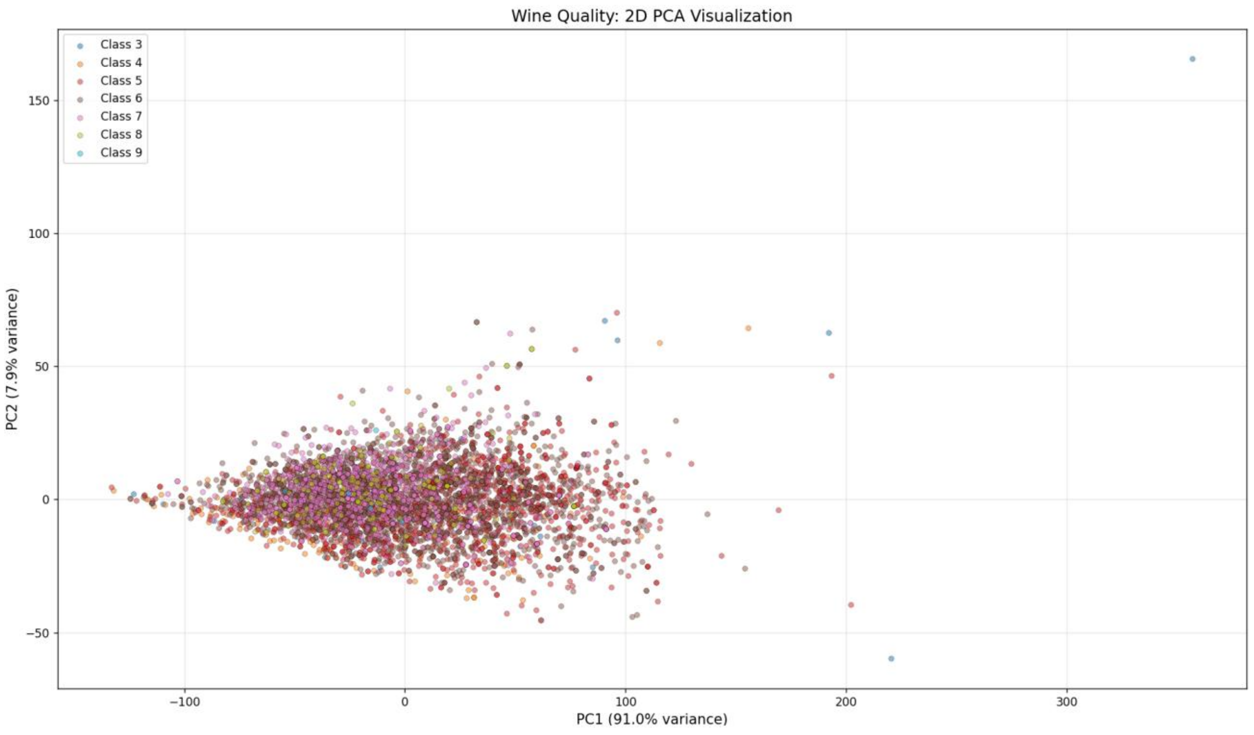

Figure 8: 2D Wine Quality PCA

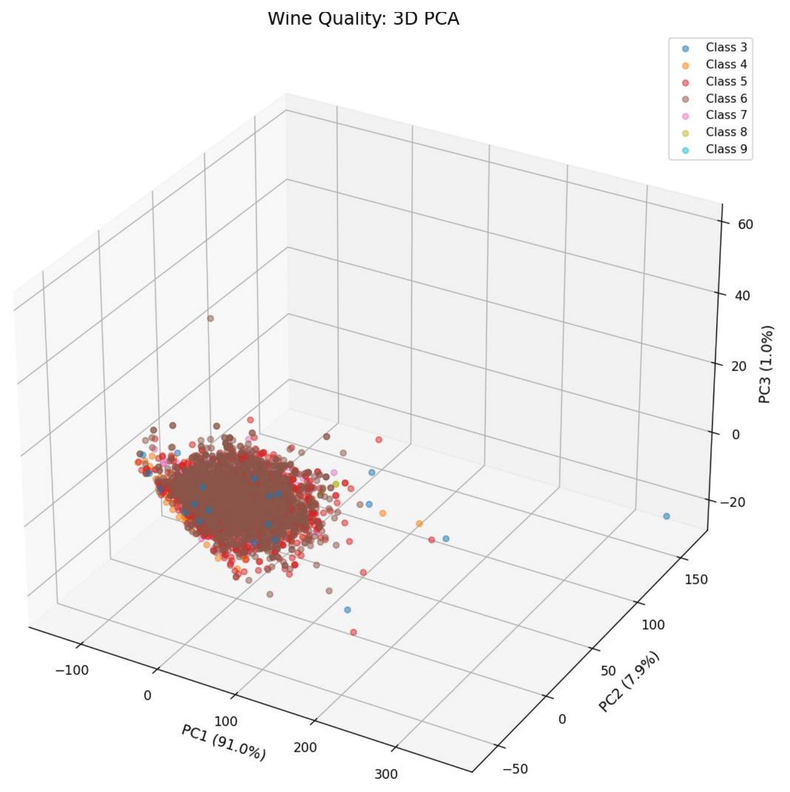

Figure 9: 3D Wine Quality PCA

When looking at the results of the 3D PCA, PC1 contained 90.97% of variance, PC2 contained 7.93%, and PC3 contained 1.02%, giving us 99.91% coverage. Even though we covered this much variance, the visualizations show complete overlap between classes. This indicated that the wine quality cannot be separated in this feature space.

Table 10 below shows the regularization results:

Class	Samples	Cond(C)	Λ
3	20	5.81*10^10	14.967
4	163	7.58*10^9	2.926
5	1457	1.05*10^10	2.093
6	2198	4.24*10^9	1.801
7	880	8.13*10^9	1.153
8	175	1.36*10^10	1.248
9	5	2.94*10^23	1.474

Table 10: Covariance Condition Numbers

The condition numbers show severe ill-conditioning. Even with regularization, the insufficient sample sizes of Classes 3 and 9 cause problems. The classification results can be seen below in Table 11:

	L=3	L=4	L=5	L=6	L=7	L=8	L=9
D=3	0.200	0.012	0.001	0.000	0.000	0.000	0.000
D=4	0.000	0.000	0.001	0.000	0.000	0.000	0.000
D=5	0.150	0.067	0.051	0.020	0.002	0.000	0.000
D=6	0.400	0.466	0.422	0.291	0.106	0.154	0.000
D=7	0.200	0.454	0.524	0.686	0.885	0.817	1.000
D=8	0.050	0.000	0.002	0.002	0.007	0.029	0.000
D=9	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 11: Wine Quality Confusion Matrix

The accuracy was 30.67% and the error rate was 69.33%. The confusion matrix showed a massive failure. The classifier predicted mostly Classes 6 and 7, with classes 3, 4, 5, 8, and 9 having <20% accuracy. The Gaussian Model was not appropriate for this dataset. This was an ordinal problem being treated as a categorical problem. The Gaussian model treats all non-matching classes equally. The PCA also shows that all classes occupy the same region of feature space. The large gaps in sample sizes per class couldn't reliably estimate an 11x11 covariance matrix. The classifier also predicted Classes 6 and 7 the most showed that the model was not learning meaningful class boundaries. The accuracy of 30.67% is marginally better than always predicting Class 6, so this model was not suitable.

3.2 Human Activity Dataset

Characteristics:

- UCI ML Repository
- Samples: 10,299
- Features: 561
- Classes: 6 activities
- Distribution:
    - Walking: 1,722 samples
    - Walking Upstairs: 1,544 samples
    - Walking Downstairs: 1,406 samples
    - Sitting: 1,777 samples
    - Standing: 1,906 samples
    - Laying: 1,944 samples

The 2D and 3D Human Activity PCA visualizations can be seen below with Figures 10 and 11:

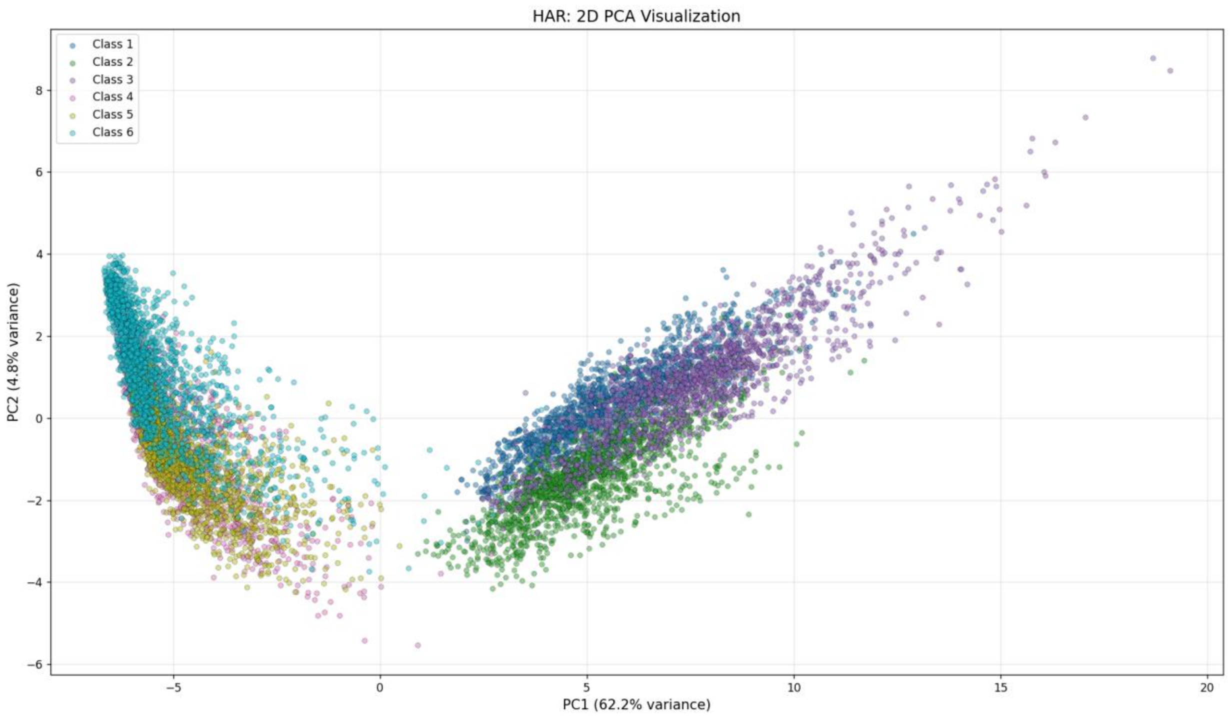

Figure 10: HAR 2D PCA

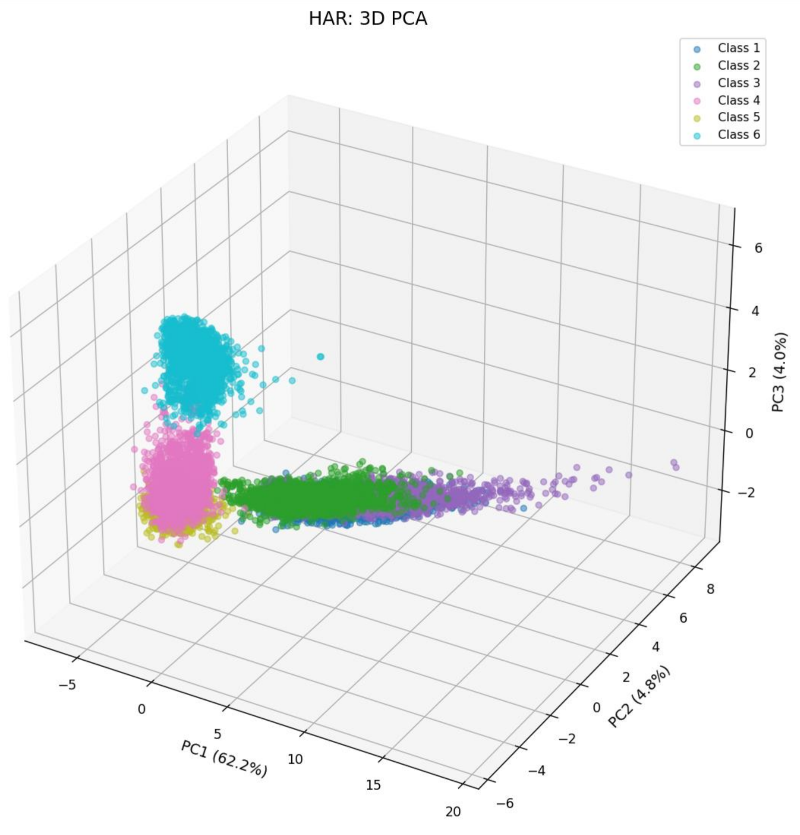

Figure 11: HAR 3D PCA

When looking at the results of the 3D PCA, PC1 contained 62.23% of variance, PC2 contained 4.77%, and PC3 contained 4.02%, giving us 71.02% coverage. The HAR PCA plots show clear separation between clusters. Different activities produce distinct patterns in the sensor feature space, with Laying completely separated from the rest.

Table 12 below shows the regularization results:

Activity	Samples	Cond(C)	Λ
Walking	1,722	1.80*10^21	0.000430
Walk Up	1,544	7.53*10^20	0.000441
Walk Down	1,406	4.45*10^20	0.000605
Sitting	1,777	1.68*10^22	0.000344
Standing	1,906	1.19*10^22	0.000311
Laying	1,944	1.55*10^22	0.000431

Table 12: Covariance Condition Numbers

Even with high dimensionality and large condition numbers, the regularization function successfully stabilized covariance estimation. Small lambda values show that minimal regularization was needed. Table 13 below contains classification results:

	L=1	L=2	L=3	L=4	L=5	L=6
D=1	0.9994	0.0000	0.0000	0.0000	0.0000	0.0000
D=2	0.0001	1.0000	0.0340	0.0000	0.0000	0.0000
D=3	0.0000	0.0000	0.9659	0.0000	0.0000	0.0000
D=4	0.0000	0.0000	0.0000	0.8137	0.0000	0.0000
D=5	0.0000	0.0000	0.0000	0.1863	1.0000	0.0000
D=6	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

Table 13: HAR Confusion Matrix

The accuracy was 96.31% and the error rate was 3.69%. The confusion matrix is almost perfectly diagonal with five of the six classes achieving near perfect classification. The only source of confusion was sitting and standing. Of the 1,777 sitting samples, 81.37% were correctly classified. Confusion about this makes sense, as both are stationary positions where the person isn't moving. This model is definitely Gaussian appropriate. The extremely high accuracy validates the model. The PCA visualizations showed distinct and well-separated clusters for each activity. The different activities all show different motion patterns, and these differences are manifested as separable clusters in the feature space. The balance in the dataset helped provide sufficient data for reliable estimation. The diagonal confusion matrix also indicated clean separation, another characteristic of a well-fit Gaussian model.

Citations

- Lecture Notes
- Code Folder
- Repo Link: https://github.com/ishaandesai0/EECE5644/tree/main/Assignment%201