Homework 1 – Sontu Narendra Gautam - 002241534

Link: ML_HW1

Q1) A) If neither the priors nor the conditional probabilities are known, the best strategy is to guess the state of nature randomly. Without any information, each state is equally likely, so we choose the state with the highest overall probability, which is just a random guess.

Since we have no information, the probability of error, P(error), depends on the actual (but unknown) priors. If we assume equal priors for simplicity, then P(error) = 0.5, as there's a 50% chance of guessing incorrectly in a two-state system. It's basically like a coin toss.

B) If the priors are known, we would guess the state of nature that has the highest prior probability. This is known as the maximum a priori (MAP) decision rule.

Mathematically, if **p(ω1) > p(ω2)**, we would always guess ω1, and vice versa. The probability of error is then **P(error) = min(p(ω1), p(ω2))**. Since we are now using some information (the priors), this P(error) will always be less than or equal to the P(error) when guessing without any information.

C) When both the priors and the conditional distributions are known, and we can observe the evidence x, the Bayes decision rule is used to minimize the probability of error. For a new observation x, we compute the posterior probability for each state of nature using Bayes' theorem:

p(ωi | x) = p(x | ωi)p(ωi) / p(x), for i = 1, 2

We then choose the state of nature that has the higher posterior probability for the observed evidence. This decision rule minimizes the probability of error, as it always chooses the state of nature that is most likely given the observed evidence.

The probability of error in this case is **P(error) = 1 - ∫ max(p(ω1 | x), p(ω2 | x)) dx** over the space of x, and this is always less than or equal to the error probability in the second question, where the evidence x was not observable.

Q2) A)

To find the decision threshold γ which minimizes the probability of error P(error|x), we can use the likelihood-ratio test. The likelihood-ratio test compares the likelihood of the data under each class hypothesis and decides in favor of the class that has the higher likelihood multiplied by its prior probability.

The likelihood ratio for class 1 to class 0 is given by:

$$\Lambda(x) = \frac{p(x|L=1)}{p(x|L=0)}$$

Here, p(x|L=0) and p(x|L=1) are the class-conditional probability density functions (pdfs) for a multivariate Gaussian distribution.

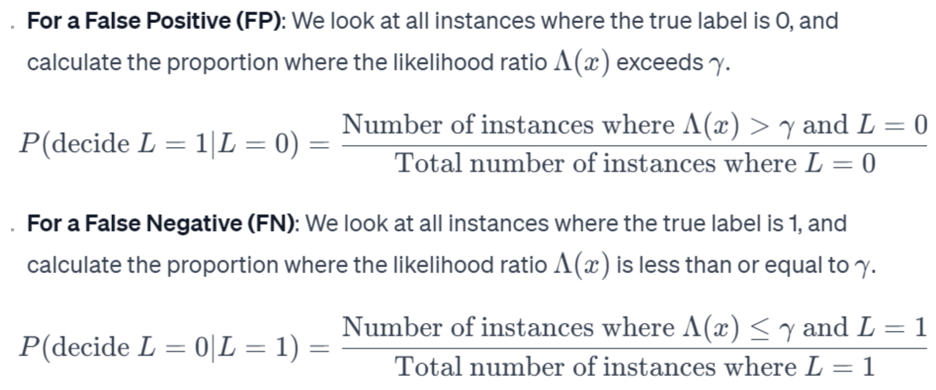
The overall probability of error P(error) is the weighted sum of the probabilities of misclassifying an instance from each class, given by:

P(error)=P(L=0)·P(decide L=1|L=0)+P(L=1)·P(decide L=0|L=1)

Where:

- P(L=0) and P(L=1) are the prior probabilities for class 0 and class 1.
- P(decide L=1|L=0) is the probability of a false positive.
- P(decide L=0|L=1) is the probability of a false negative.

To minimize P(error), we numerically search for the γ that results in the lowest value of this expression. This is done by evaluating P(error) for a range of γ values and selecting the one that yields the minimum error.



- **For a False Positive (FP):** We look at all instances where the true label is 0, and calculate the proportion where the likelihood ratio $\Lambda(x)$ exceeds γ.

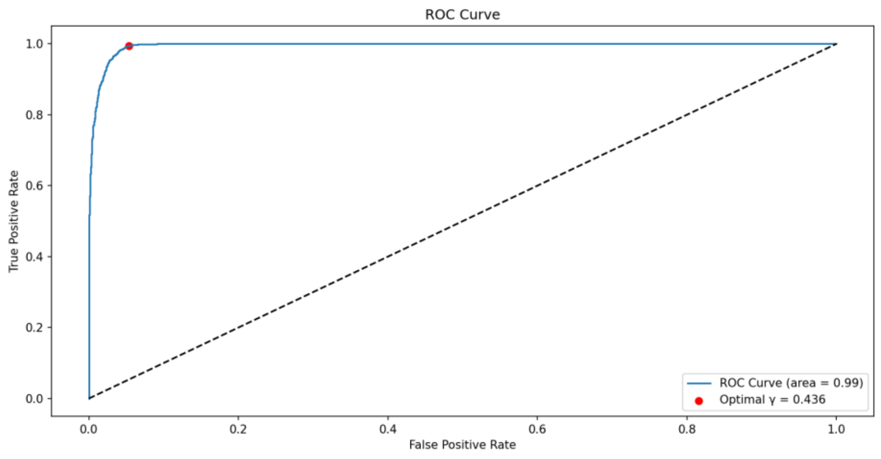
$$P(\text{decide } L=1|L=0) = \frac{\text{Number of instances where } \Lambda(x) > \gamma \text{ and } L=0}{\text{Total number of instances where } L=0}$$

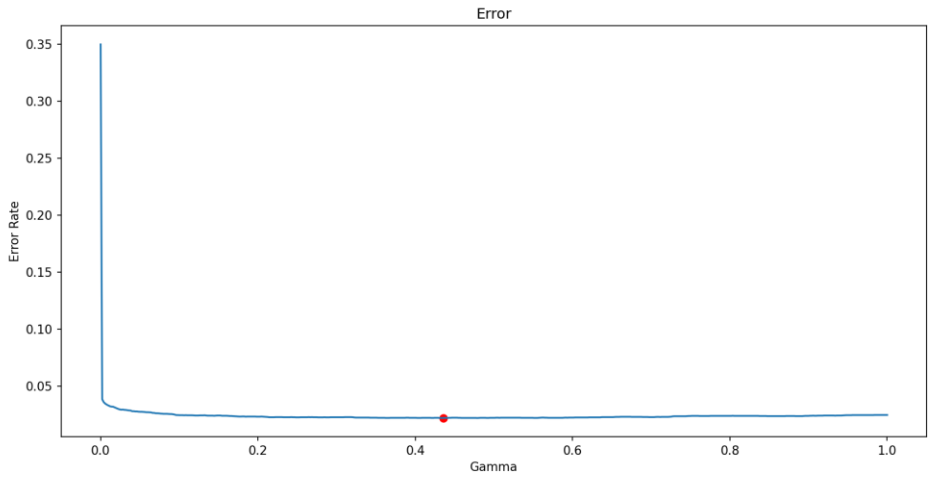
- **For a False Negative (FN):** We look at all instances where the true label is 1, and calculate the proportion where the likelihood ratio $\Lambda(x)$ is less than or equal to γ.

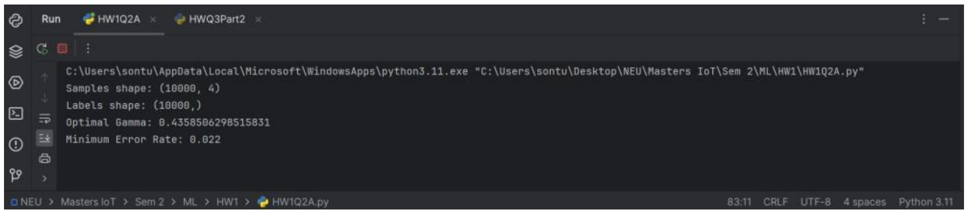
$$P(\text{decide } L=0|L=1) = \frac{\text{Number of instances where } \Lambda(x) \leq \gamma \text{ and } L=1}{\text{Total number of instances where } L=1}$$

Once the optimal γ is found, we can calculate the minimum total P(error) by plugging this γ back into the above expression for P(error).

Got the, Optimal γ = 0.4358506298515831
Minimum Empirical Error: 0.022





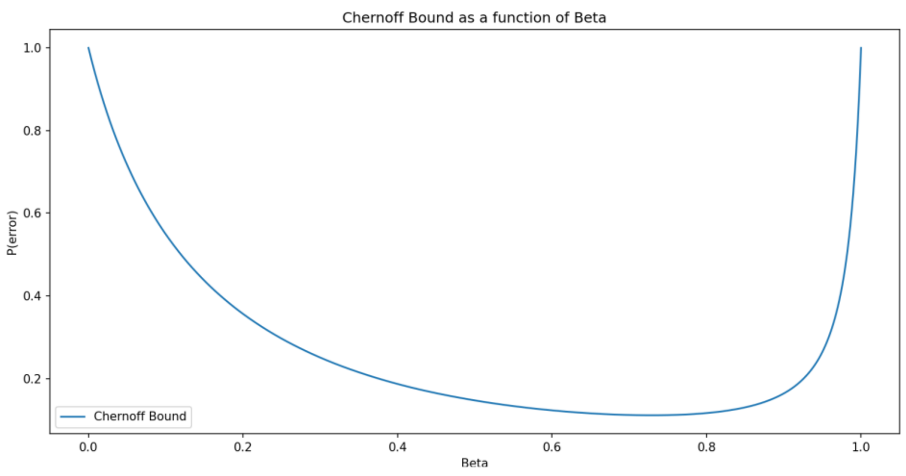

Run
HW1Q2A
HWQ3Part2
NEU > Masters IoT > Sem 2 > ML > HW1 > HW1Q2A.py
83:11 CRLF UTF-8 4 spaces Python 3.11

```
C:\Users\sontu\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:\Users\sontu\Desktop\NEU\Masters IoT\Sem 2\ML\HW1\HW1Q2A.py"
Samples shape: (10000, 4)
Labels shape: (10000,)
Optimal Gamma: 0.4358506298515831
Minimum Error Rate: 0.022
```

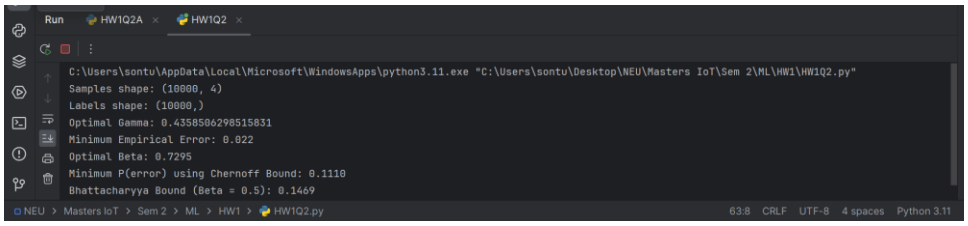
With an optimal γ of 0.4359, the minimum empirical error was 0.022, indicating high classification accuracy when the model accurately reflects the true data distribution.

B)

Plotting the Chernoff bound curve as a function of β ∈ [0, 1],



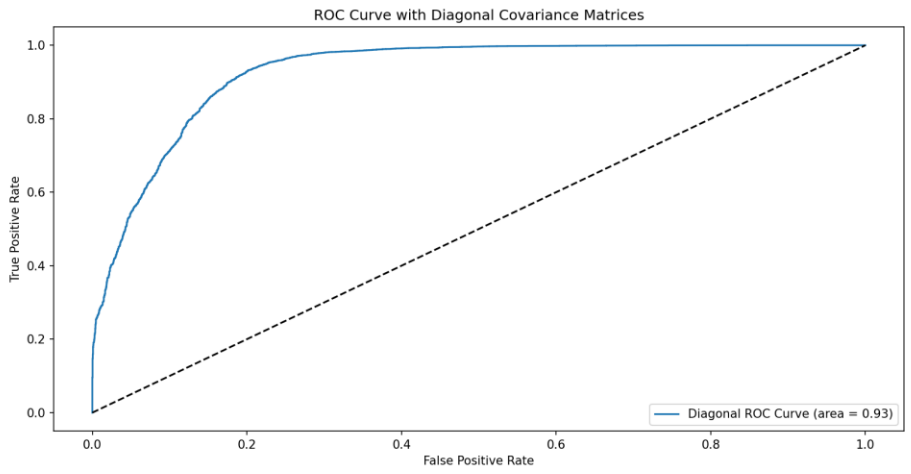
and the β that minimizes it 0.7295


Run
HW1Q2A
HW1Q2
NEU > Masters IoT > Sem 2 > ML > HW1 > HW1Q2.py
63:8 CRLF UTF-8 4 spaces Python 3.11

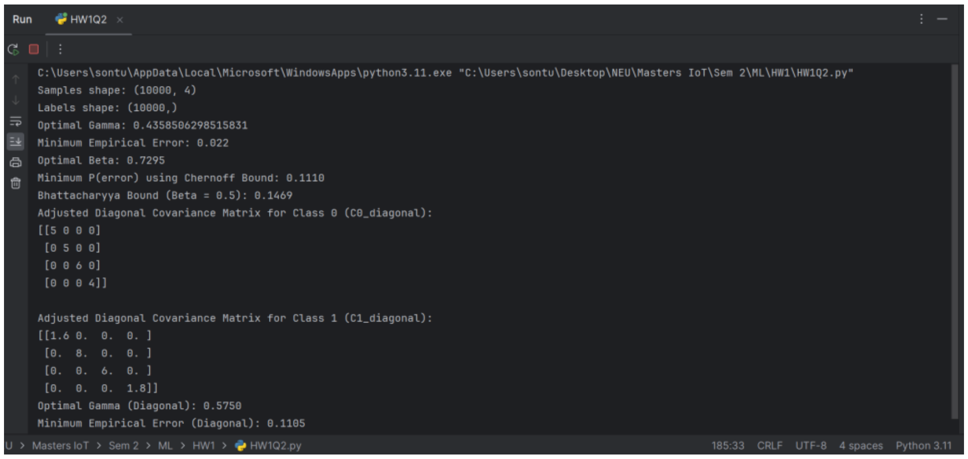
```
C:\Users\sontu\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:\Users\sontu\Desktop\NEU\Masters IoT\Sem 2\ML\HW1\HW1Q2.py"
Samples shape: (10000, 4)
Labels shape: (10000,)
Optimal Gamma: 0.4358506298515831
Minimum Empirical Error: 0.022
Optimal Beta: 0.7295
Minimum P(error) using Chernoff Bound: 0.1110
Bhattacharyya Bound (Beta = 0.5): 0.1469
```

Comparison of P(error) you computed in Part A, the P(error) empirically in Part A, with both the Chernoff (optimal β) and Bhattacharyya bounds (β =1/2), would be,
Minimum Empirical Error in Part A: 0.022
Minimum P(error) using Chernoff Bound: 0.1110
Bhattacharyya Bound (β = 0.5): 0.1469

C)



Optimal Gamma (Diagonal): 0.5750
Minimum Empirical Error (Diagonal): 0.1105

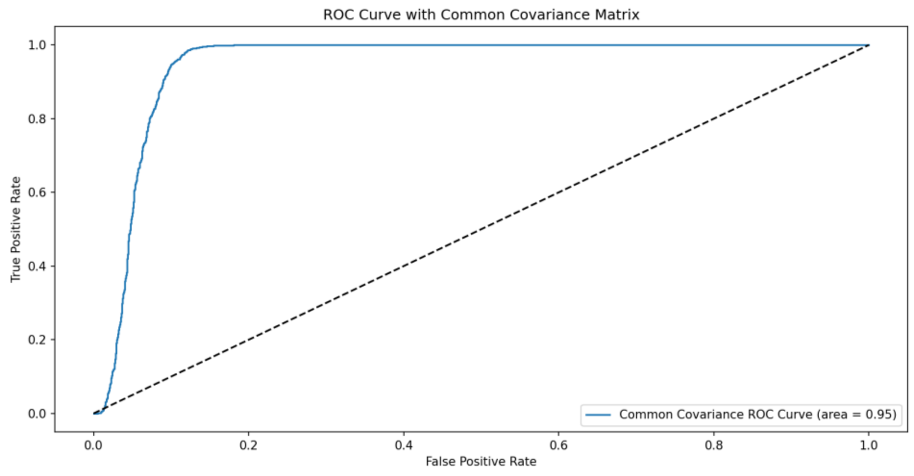

Run
HW1Q2
U > Masters IoT > Sem 2 > ML > HW1 > HW1Q2.py
185:33 CRLF UTF-8 4 spaces Python 3.11

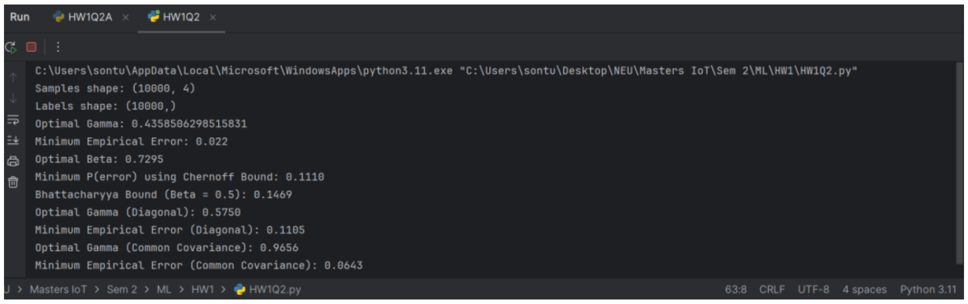
```
C:\Users\sontu\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:\Users\sontu\Desktop\NEU\Masters IoT\Sem 2\ML\HW1\HW1Q2.py"
Samples shape: (10000, 4)
Labels shape: (10000,)
Optimal Gamma: 0.4358506298515831
Minimum Empirical Error: 0.022
Optimal Beta: 0.7295
Minimum P(error) using Chernoff Bound: 0.1110
Bhattacharyya Bound (Beta = 0.5): 0.1469
Adjusted Diagonal Covariance Matrix for Class 0 (C0_diagonal):
[[5 0 0 0]
 [0 5 0 0]
 [0 0 6 0]
 [0 0 0 4]]

Adjusted Diagonal Covariance Matrix for Class 1 (C1_diagonal):
[[1.6 0.  0.  0. ]
 [0.  8.  0.  0. ]
 [0.  0.  6.  0. ]
 [0.  0.  0.  1.8]]
Optimal Gamma (Diagonal): 0.5750
Minimum Empirical Error (Diagonal): 0.1105
```

It is evident that the assumption of diagonal covariance matrices—while simplifying the computational aspects of the classification task—leads to a compromise in classification accuracy, as demonstrated by the optimal γ of 0.5750 and the minimum empirical error of 0.1105. This compromise underscores the importance of model selection that accurately reflects the underlying data characteristics to achieve optimal classification performance.

D)




Run
HW1Q2A
HW1Q2
U > Masters IoT > Sem 2 > ML > HW1 > HW1Q2.py
63:8 CRLF UTF-8 4 spaces Python 3.11

```
C:\Users\sontu\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:\Users\sontu\Desktop\NEU\Masters IoT\Sem 2\ML\HW1\HW1Q2.py"
Samples shape: (10000, 4)
Labels shape: (10000,)
Optimal Gamma: 0.4358506298515831
Minimum Empirical Error: 0.022
Optimal Beta: 0.7295
Minimum P(error) using Chernoff Bound: 0.1110
Bhattacharyya Bound (Beta = 0.5): 0.1469
Optimal Gamma (Diagonal): 0.5750
Minimum Empirical Error (Diagonal): 0.1105
Optimal Gamma (Common Covariance): 0.9656
Minimum Empirical Error (Common Covariance): 0.0643
```

The model mismatch impacts the ROC curve and minimum achievable empirical probability of error in the following ways, as indicated by the data from Parts A, C, and D:

Part A (True Model)	Part C (Diagonal Covariance Matrices)	Part D (Common Covariance Matrix)
With an optimal γ of 0.4359, the minimum empirical error was 0.022, indicating high classification accuracy when the model accurately reflects the true data distribution.	Assuming diagonal covariance matrices (a simplification), the minimum empirical error increased to 0.1105 with an optimal γ of 0.5750. This increase in error and change in γ reflects the loss of accuracy due to model mismatch, specifically ignoring off-diagonal elements of the covariance matrices that represent feature correlations.	Under the assumption of a common covariance matrix for both classes, the minimum empirical error was 0.0643 with an optimal γ of 0.9656. This scenario shows a better performance than the diagonal assumption but still underperforms compared to the true model, indicating that assuming a common covariance matrix also introduces inaccuracies but to a lesser extent than the diagonal covariance assumption.

Summary: Model mismatch, whether assuming diagonal covariance matrices or a common covariance matrix, negatively impacts classification performance compared to using the true covariance structures. This is evident from the increased empirical errors and shifts in optimal γ values. The simplifications lead to less accurate estimations of class boundaries, as seen in the increased minimum empirical errors for the mismatched models compared to the true model.

E) To calculate the minimum expected risk, we consider the cost of misclassification and the probabilities of making such errors. Given the cost matrix λ where the cost of deciding class 1 when the true class is 0 is B, and the cost of deciding class 0 when the true class is 1 is 1, the expected risk for a decision threshold gamma is defined as:

R(γ) = λ _21 * P(decide 1 | true 0) * P(true 0) + λ _12 * P(decide 0 | true 1) * P(true 1)

Given that λ _21 = 1 and λ _12 = B, the expected risk becomes:

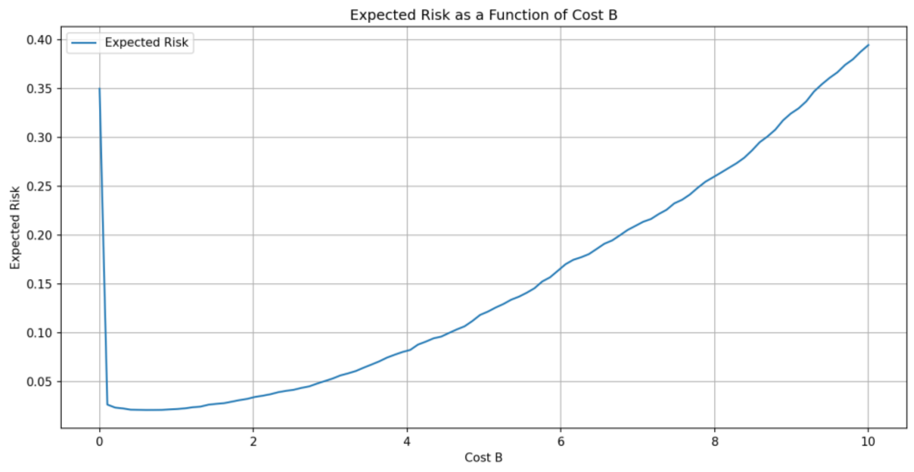
R(γ) = P(decide 1 | true 0) * P(true 0) + B * P(decide 0 | true 1) * P(true 1)

With the likelihood ratios and the decision threshold γ, we can calculate P(decide 1 | true 0) (false positive rate) and P(decide 0 | true 1) (false negative rate).

These calculations involve determining the proportion of times a decision exceeds the threshold γ for samples from each class, thus identifying false positives and false negatives. By integrating these probabilities with their respective costs and class priors, we obtain the overall expected risk associated with a particular decision threshold.

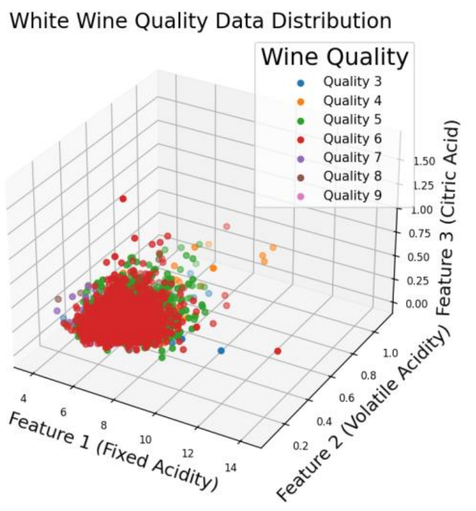
To minimize the expected risk, one would numerically search for the γ value that results in the lowest R(γ), balancing the trade-off between the costs of false positives and false negatives as dictated by the cost matrix λ.

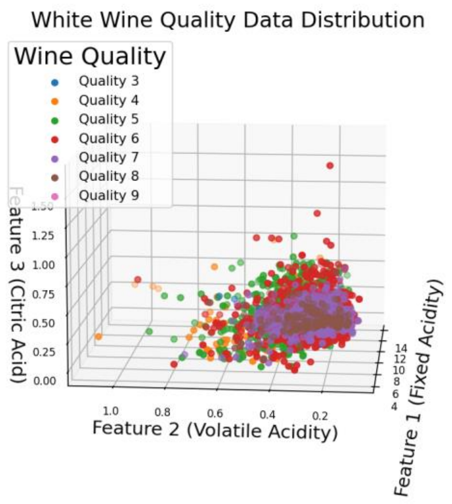
This approach ensures that the classification decision not only aims for high accuracy but also considers the differential costs associated with misclassifying samples from different classes, leading to a more cost-effective classification strategy.



This is the plot for minimum-risk decision boundary γ, as well as the corresponding minimum expected risk, as a function of B

Q3) A) The analysis of the Wine Quality dataset, featuring 4898 samples of white wine across 11 features, reveals a skewed distribution of quality classes, primarily clustering around classes 4, 5, and 6. This skewness, combined with the sparse representation of the extreme quality classes, presents a unique challenge for classification. The dataset's comprehensive feature set eliminates the need for dimensionality reduction, preserving the integrity of the information critical for model training.
Under the assumption that the dataset's samples are independent and identically distributed (i.i.d.), we utilize the central limit theorem to approximate a Gaussian distribution for the class conditional PDFs. This approximation is supported by the substantial sample size, enhancing the model's foundation for applying Gaussian statistics.





Class priors are calculated based on the proportion of samples in each class, using the formula: **P(L) = Number of samples in class L / Total number of samples**.

Class priors, essential for our Bayesian approach, are meticulously calculated to reflect the real distribution of classes within the dataset. However, the regularization of covariance matrices becomes necessary due to their high condition numbers, introducing a small λI term. This term, λ, grounded in the arithmetic average of the covariance matrix's non-zero eigenvalues, ensures a balanced adjustment of variances.

On testing the conditionality of the co-variance matrices, it is identified that most of them yield a large conditional number. Therefore, it is essential to add a small regularization value to broaden the distribution.

CRegularized = CSampleAverage+λI

Here, λis a hyper-parameter which decides the amount of regularization added to the original variance values. To calculate λ, I considered finding out the arithmetic average of the non-zero eigen values of the matrix. The trace (sum of diagonal elements) is equal to the sum of eigen values of a matrix and rank is number of nonzero eigen values. Hence,

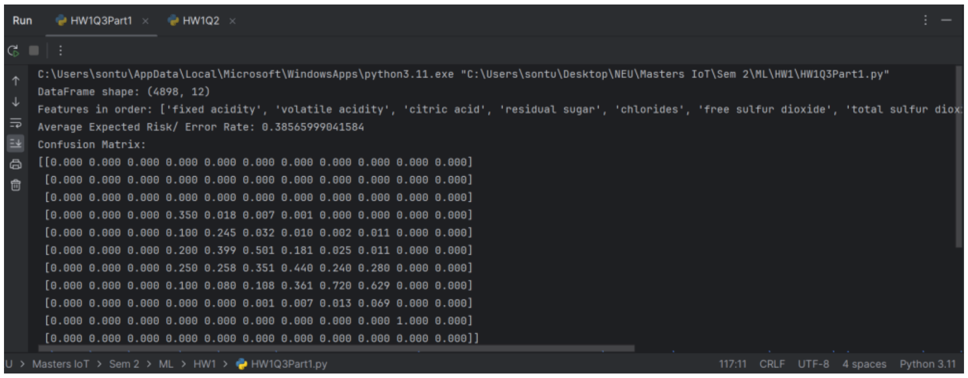
Arithmetic Average = trace (CSampleAverage) / rank (CSampleAverage)

λ = α (Arithmetic Average)

where α is a small real number between 0 and 1. This is a hyperparameter controlling the main hyperparameter λ, which can be tuned in the range of 10-3 to 10-9 to check for maximum.

```
regularization_term = 0.000000005 * (np.trace(covariance_matrices[i, :, :]) /
LA.matrix_rank(covariance_matrices[i, :, :])) * np.eye(num_features)
```

Despite the theoretical grounding in Gaussian distribution and the central limit theorem, the model exhibits an average expected error rate of approximately 38.6%. This rate highlights the model's limitations in accurately classifying wine quality, particularly for classes with fewer samples.

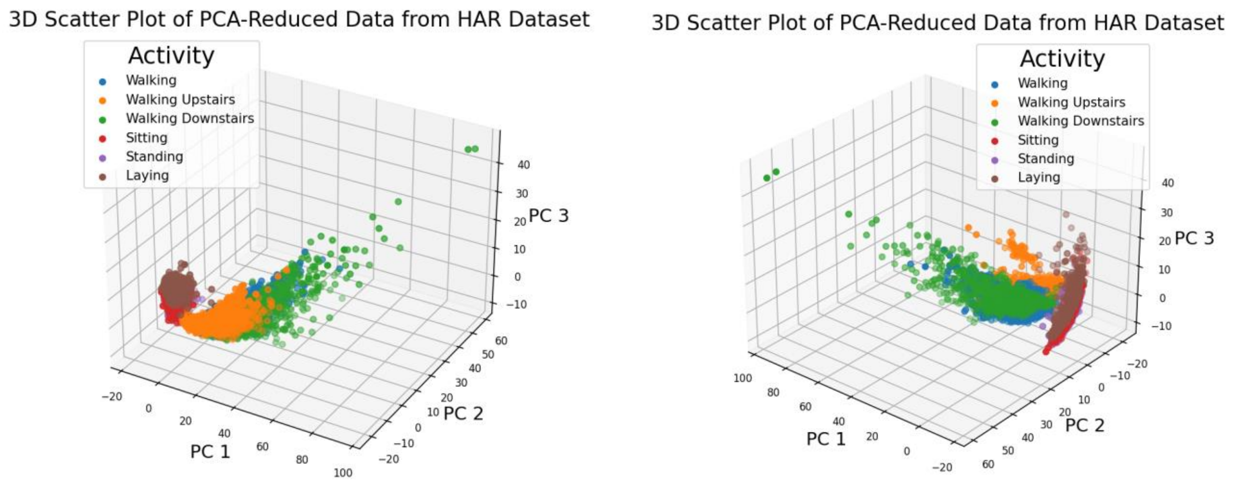


```
C:\Users\sontu\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:\Users\sontu\Desktop\NEU\Masters IoT\Sem 2\ML\HW1\HW1Q3Part1.py"
DataFrame shape: (4898, 12)
Features in order: ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur diox
Average Expected Risk/ Error Rate: 0.38565999041584
Confusion Matrix:
[[0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000]
 [0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000]
 [0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000]
 [0.000 0.000 0.000 0.350 0.018 0.007 0.001 0.000 0.000 0.000 0.000]
 [0.000 0.000 0.000 0.100 0.245 0.032 0.010 0.002 0.011 0.000 0.000]
 [0.000 0.000 0.000 0.200 0.399 0.501 0.181 0.025 0.011 0.000 0.000]
 [0.000 0.000 0.000 0.250 0.258 0.351 0.440 0.240 0.280 0.000 0.000]
 [0.000 0.000 0.000 0.100 0.080 0.108 0.361 0.720 0.629 0.000 0.000]
 [0.000 0.000 0.000 0.000 0.000 0.001 0.007 0.013 0.069 0.000 0.000]
 [0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000]
 [0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000]]
```

U > Masters IoT > Sem 2 > ML > HW1 > HW1Q3Part1.py
117:11 CRLF UTF-8 4 spaces Python 3.11

The observed outcomes hint at the Gaussian classification method's limitations, especially for datasets with uneven class distributions. The precision in estimating class priors becomes paramount, as any inaccuracies directly impact the class posteriors and, consequently, the model's overall performance. This analysis underscores the critical need for exact prior estimation to enhance classification accuracy, reminding us of the method's inherent constraints and the careful considerations required when applying it to diverse classification problems.

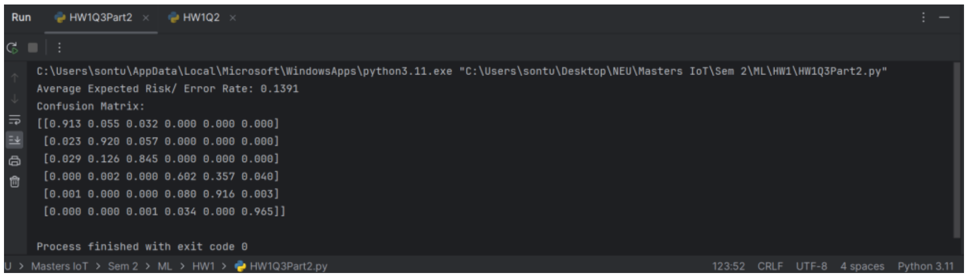
B) Given the dataset comprises 561 features, this high dimensionality presents challenges such as redundant or correlated features that do not significantly contribute to classification accuracy and increased computational complexity. To address these issues, Principal Component Analysis (PCA) is employed as a dimensionality reduction technique. PCA effectively transforms the original high-dimensional feature space into a smaller set of uncorrelated variables that capture the most significant variance, thereby simplifying the dataset while retaining essential information for effective classification.



From the plots, 'Walking', 'Walking Upstairs', and 'Walking Downstairs' are dynamic activities and are somewhat overlapped, they are still reasonably distinguishable from the static activities. This separation likely reflects differences in the sensor data due to movement. 'Sitting', 'Standing', and 'Laying' are closely grouped together, which could be due to the similar nature of the sensor readings during these static activities. However, within this cluster, 'Standing' and 'Sitting' might be more challenging to distinguish, as indicated by their overlapping points.

In this scenario, Principal Component Analysis (PCA) was applied to condense the feature space from 561 down to 10 principal components, focusing on preserving the most critical variance within the data. Analysis of the class priors indicated a relatively even distribution across all six activity labels, suggesting a balanced dataset.

Contrary to the challenges faced in the Wine Quality Classification, the Gaussian classifier demonstrated improved performance in this context. The balanced nature of the dataset ensured class priors were within a similar range, allowing the model to more effectively rely on class-conditional distributions for classification. This approach yielded an Average Expected Risk/Error Rate of approximately 13.91%, illustrating the model's efficacy in identifying meaningful relationships between the reduced features and activity label


Run
HW1Q3Part2
HW1Q2
U > Masters IoT > Sem 2 > ML > HW1 > HW1Q3Part2.py
123:52 CRLF UTF-8 4 spaces Python 3.11

```
C:\Users\sontu\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:\Users\sontu\Desktop\NEU\Masters IoT\Sem 2\ML\HW1\HW1Q3Part2.py"
Average Expected Risk/ Error Rate: 0.1391
Confusion Matrix:
[[0.913 0.055 0.032 0.000 0.000 0.000]
 [0.023 0.920 0.057 0.000 0.000 0.000]
 [0.029 0.126 0.845 0.000 0.000 0.000]
 [0.000 0.002 0.000 0.602 0.357 0.040]
 [0.001 0.000 0.000 0.080 0.916 0.003]
 [0.000 0.000 0.001 0.034 0.000 0.965]]

Process finished with exit code 0
```

This outcome underscores PCA's utility in extracting pertinent features from a vast dataset, enhancing model accuracy by highlighting significant relationships between features and activity labels.