UMC203: Artificial Intelligence and Machine Learning

Kintan Saha SR No. 23881

July 5, 2025

Question 1: Fisher Linear Discriminant

Querying the Oracle provided the following two attributes: 'Male' and 'Black hair'. Hence, we have the following classes of data:

1. Class 0: 'Female' and 'Non Black Hair'
2. Class 1: 'Male' and 'Non Black Hair
3. Class 2: 'Female' and 'black hair'
4. Class 3: 'Male' and 'black hair'

The oracle has also provided the `train_images`, `train_labels`, `test_images` and `test_labels`. From the `train_images`, the images are filtered into the 4 classes to which they belong.

Part 1

We wish to find the change in estimates, the l_2 norm of the mean vector of each class, and the Frobenius norm of the covariance matrices of each class, with the number of samples chosen. For each of n = 50, 100, 500, 1000, 2000, 4000, we use `np.random.choice` to randomly choose subsets of the desired size from each of the classes. The results thus obtained are plotted below in Figure 1 and Figure 2.

Part 2(a)

We now wish to develop a multi-class FLD to classify examples from the 4 classes given to us. We have 4 classes; hence we define the between-class scatter matrix and the within-class scatter matrix as the following:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

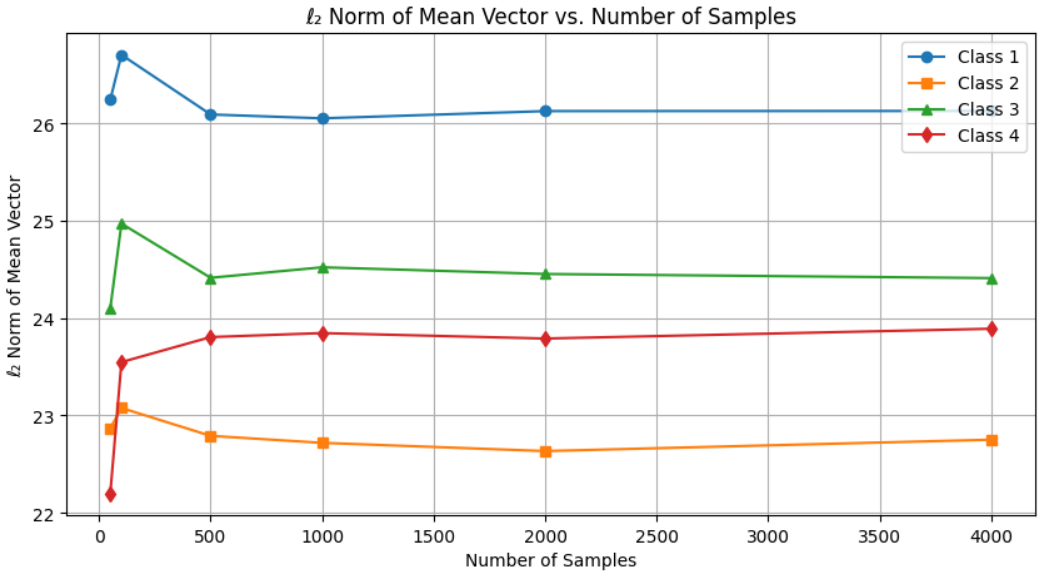

Figure 1: l_2 norm

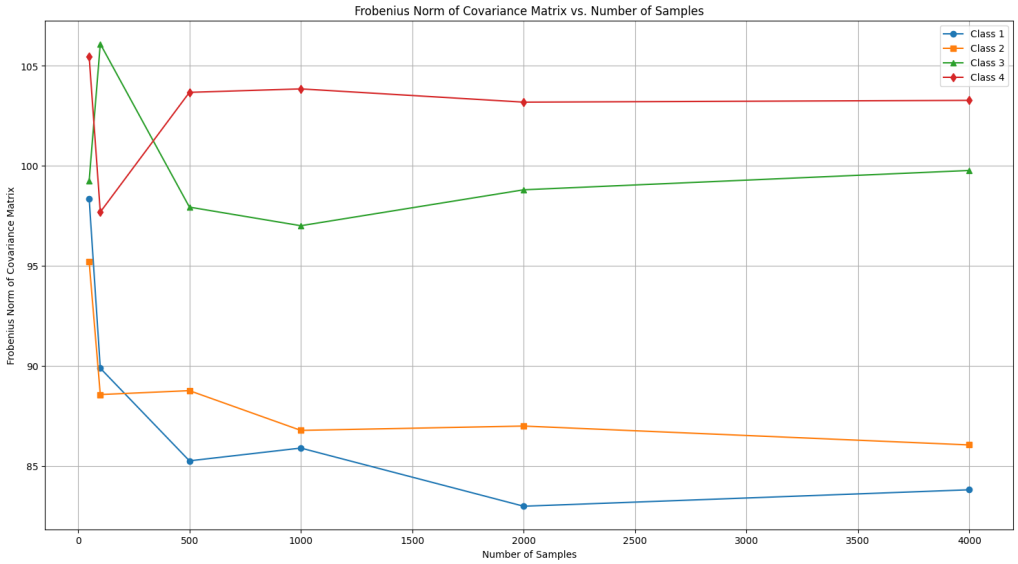

Figure 2: Frobenius Norm

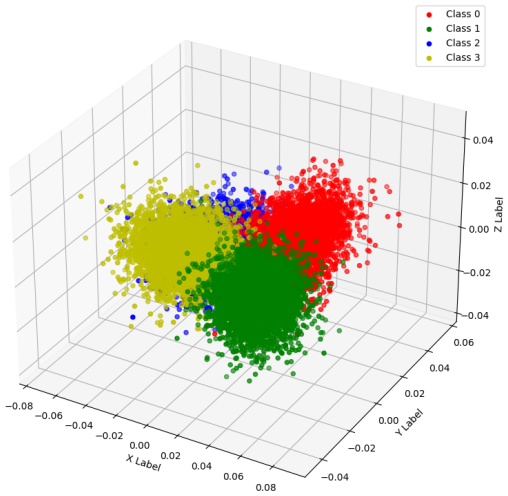

Figure 3: Projection of `train_images` on subspace using multi class FLD trained on entire `train_images`

where c is the number of classes, N_i is the number of samples present in class i, μ_i is the mean vector of class i, μ is the mean vector of the entire training set.
We now wish to solve the following optimization problem:

$$\max_W J(\mathbf{W}) = \frac{det(\mathbf{W}^T S_B \mathbf{W})}{det(\mathbf{W}^T S_W \mathbf{W})}$$

From the textbook and the paper, we know that this is equivalent to finding the eigenvectors corresponding to the top 3 eigenvalues of

$$S_W^{-1} S_B$$

(Need to find largest c-1 eigenvalues where c is the number of classes).The weight matrix W is obtained by using these 3 eigenvectors as columns for the matrix. Hence, we will project our data points on the subspace spanned by these 3 eigenvectors. The plot of projection of the training dataset onto the subspace spanned by the 3 eigenvectors is given in Figure 3.
We have also found the weights, that is, the matrix W, for each of the 20 subsets formed taking n = 2500, 3500, 4000, 4500, 5000. We have also computed the multi-class objective function for each of the 20 subsets for each of the n=2500, 3500, 4000, 4500, 5000. We have used this data to plot a box plot of the multi-class objective function. The box plot is shown in Figure 4.
The definition of the multi class objective value is (where W^* is the optimal weight matrix):

$$\frac{det(\mathbf{W}^{*T} S_B \mathbf{W}^*)}{det(\mathbf{W}^{*T} S_W \mathbf{W}^*)}$$

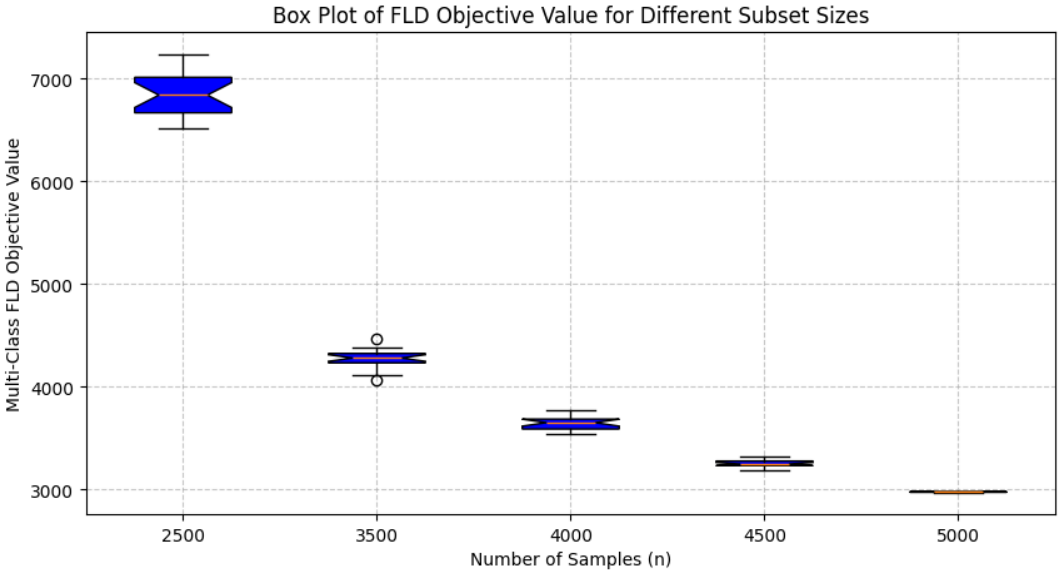

Figure 4: boxplot for multi class FLD objective function corresponding to different subset sizes

We observe that as n increases, the variance in the objective value decreases and the median of the objective value also decreases.
We have also plotted the projection of `train_images` on the 'linear discriminant' for 1 subset for each of n =2500, 3500, 4000, 4500 in Figures 5, 6, 7, 8.
Our classifier has the following structure: Project the data points using the top 3 eigenvectors. Then, for each class find its mean vector and covariance matrix in the projected subspace. Use these to form a multivariate normal for each class. For a given projected test data point find the class for which the test data point has highest likelihood.
As such, this classifier does not have a sharp threshold. Approximate thresholds can be obtained by using the projection of `train_images`. However, this would not be accurate and would be prone to errors. I did not use a linear classifier such as the classifier of minimum distance from mean vector of the projected classes. This was because such linear classifiers led to very low accuracy (0.25).

Part 2(b)

The classifier is designed as follows: Each class is modeled using a multivariate Gaussian distribution. The mean vector and the covariance matrix for each of the multivariate normals is obtained as follows:

$$\mu_i = \frac{\sum_{x_k \in X_i} x_k}{N_i}$$

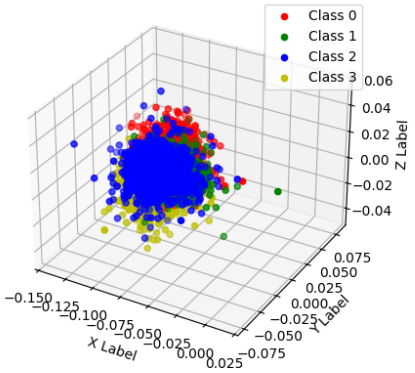

Figure 5: Projection for n = 2500

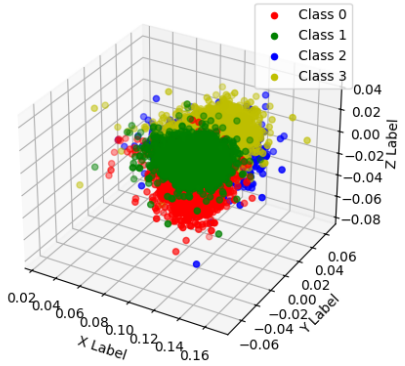

Figure 6: Projection for n = 3500

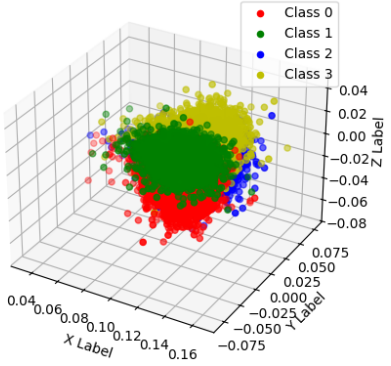

Figure 7: projection for n = 4000

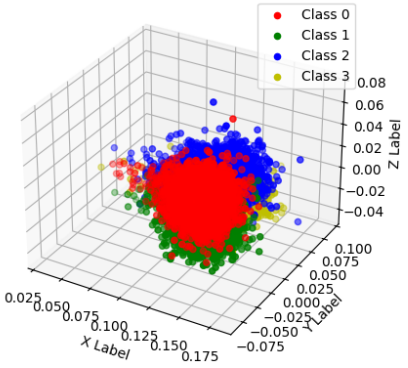

Figure 8: projection for n = 4500

$$\Sigma_i = \frac{\sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T}{N_i - 1}$$

that is, we are using the sample means and sample covariance of the classes as the mean vector and the covariance matrix for the mulitvariate normals for each of the classes. For a test point, the class is assigned using the following rule:

$$\eta^*(x) = \arg\max_j \Pr(Y = j \mid X = x) \tag{1}$$

$$= \arg\max_j \Pr(X = x \mid Y = j) \Pr(Y = j) \tag{2}$$

$$= \arg\max_j \Pr(X = x \mid Y = j) \tag{3}$$

$$\tag{4}$$

This decision rule gives us an accuracy of **72**% when using the FLD trained on the entire `train_images`. We then train the multi class FLD separately on datasets where each class has n = 2500, 3500, 4000, 4500 samples. The following accuracy was obtained for each of them as shown in Table 1.

n	Accuracy
2500	0.673
3500	0.695
4000	0.709
4500	0.713
5000	0.720

Table 1: Accuracy values for different values of n

Question 2: Bayes Classification

Procedure:

I have adopted 2 different approaches to solving this question. This was necessitated by computational issues. The first approach that I have taken is to assume the features of the data to be independent of each other. Theoretically this isn't a good assumption but this assumption gives rise to very good accuracy, low misclassification loss. The other approach that I have taken is to not assume the features are independent of each other.

Approach 1:

The first approach entails the following: For each class, each feature is modeled using a 1D normal distribution with mean as the sample mean of that feature and variance as the sample variance of the feature. Hence, given a new test data point we compute the likelihood of that point being in class 1 as follows:

$$P(x \mid C_1) = P(x_1, x_2, \ldots, x_d \mid C_1)$$

Assuming that the features are conditionally independent given the class:

$$P(x \mid C_1) = \prod_{i=1}^{d} P(x_i \mid C_1)$$

where each feature x_j is modeled as:

$$P(x_j \mid C_i) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp\left(-\frac{(x_j - \mu_{ij})^2}{2\sigma_{ij}^2}\right)$$

Thus, the overall likelihood is:

$$P(x \mid C_i) = \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp\left(-\frac{(x_j - \mu_{ij})^2}{2\sigma_{ij}^2}\right)$$

or equivalently:

$$P(x \mid C_i) = \frac{1}{(2\pi)^{d/2} \prod_{j=1}^{d} \sigma_{ij}} \exp\left(-\sum_{j=1}^{d} \frac{(x_j - \mu_{ij})^2}{2\sigma_{ij}^2}\right)$$

The posterior probability of class C_1 given a data point x is given by Bayes' theorem:

$$P(C_1 \mid x) = \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

where the evidence $P(x)$ is:

$$P(x) = P(x \mid C_1)P(C_1) + P(x \mid C_2)P(C_2)$$

Similarly, for class C_2:

$$P(C_2 \mid x) = \frac{P(x \mid C_2)P(C_2)}{P(x)}$$

he **Modified Bayes Classifier** h_ϵ with a reject option is defined as:

$$h_\epsilon(x) = \begin{cases} 1, & \text{if } \eta(x) \geq \frac{1}{2} + \epsilon \\ 0, & \text{if } \eta(x) \leq \frac{1}{2} - \epsilon \\ \text{reject}, & \text{otherwise} \end{cases}$$

where $\eta(x) = P(Y = 1 \mid X = x)$ is the posterior probability of class 1, and $\epsilon \in \left(0, \frac{1}{2}\right)$ is the threshold parameter.
Querying the oracle, I get images of 2 classes : labels 3 and 35. I have assigned class 1 as the class with label 3 and class 0 as the class with label 35. The reject option is modeled using -1 in my code.

Results for Approach 1:

Part 1:

We train the modified Bayes Classifier on the train set with different values of epsilon 0.01, 0.1, 0.25, 0.4. We then run it on the test dataset of size 800 and find the misclassification loss and the number of rejected samples. We also calculate the accuracy of the classifier. The results have been tabulated below:

ϵ	0.01	0.1	0.25	0.4
Misclassification Loss	0.08625	0.08625	0.08625	0.085
Rejected Samples	15	15	15	16
Accuracy	0.895	0.895	0.895	0.895

Table 2: Performance metrics for different values of ϵ

I have also plotted misclassification loss versus epsilon in Figure 9.

Part 2:

We modify the dataset by subsampling to create different class priors of 60-40, 80-20, 90-10, 99-1. We then train the Modified Bayes Classifier on the train set under these modified priors for different values of epsilon 0.1, 0.25, 0.4. After training, we evaluate the classifier on the test dataset and report the misclassification loss among the nonrejected samples and the number of rejected samples. The plot of misclassification loss versus epsilon over different splits can be found in Figure 10. The table for the results of misclassification loss and number of rejected samples is Table 3.

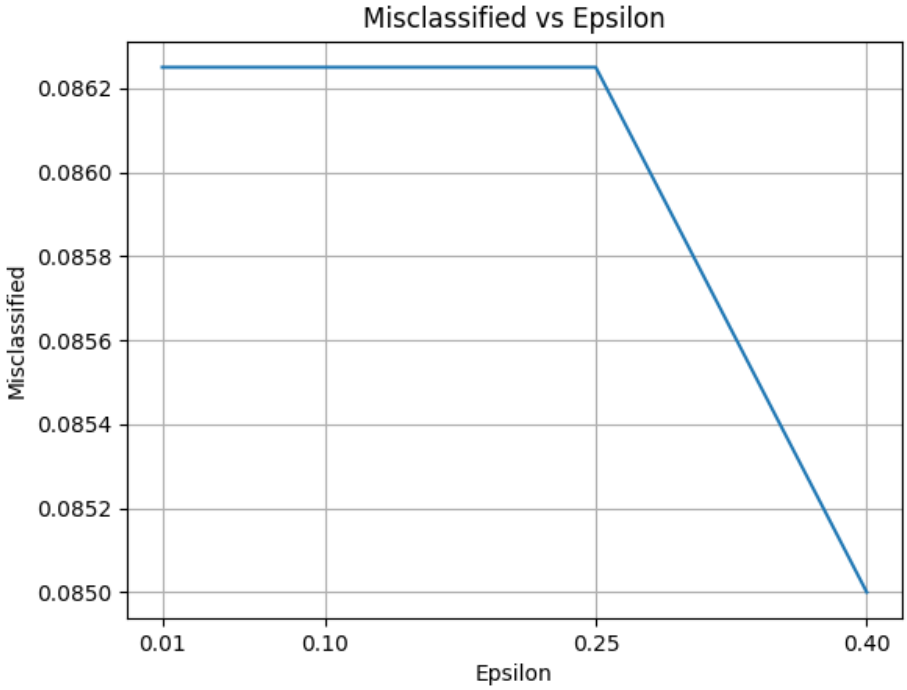

Figure 9: misclassification loss vs epsilon

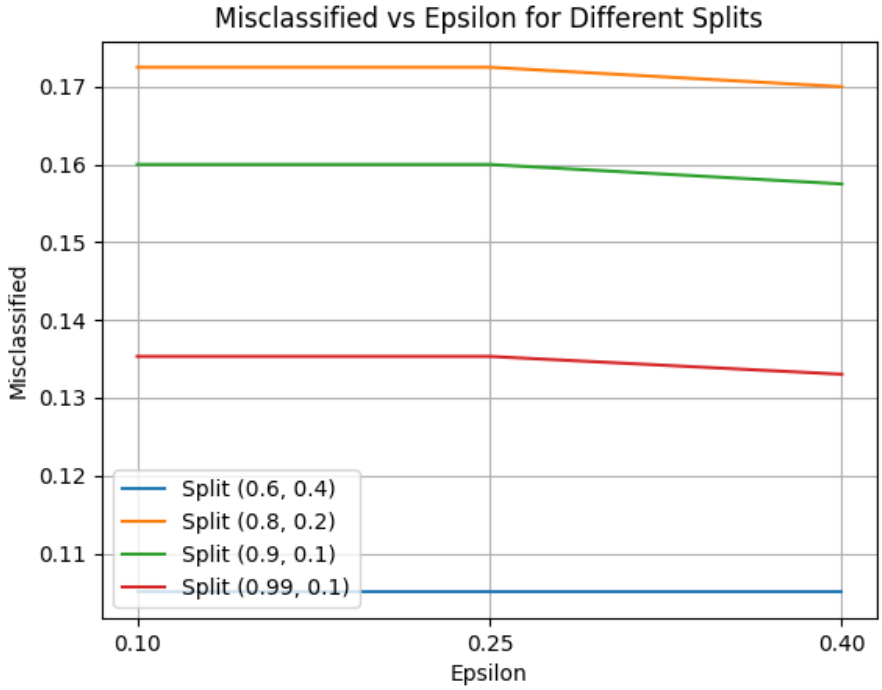

Figure 10: misclassification loss vs epsilon for different splits

Prior Split	Misclassification Loss			Rejected Samples		
	$\epsilon = 0.1$	$\epsilon = 0.25$	$\epsilon = 0.4$	$\epsilon = 0.1$	$\epsilon = 0.25$	$\epsilon = 0.4$
60-40	0.1050	0.1050	0.1050	2	3	3
80-20	0.1725	0.1725	0.1700	10	10	11
90-10	0.1600	0.1600	0.1575	13	14	16
99-1	0.1353	0.1353	0.1330	14	14	15

Table 3: Misclassification loss and number of rejected samples for different prior splits and ϵ values.

Part 3(a):

We now perform a k-fold classification of the data using the number of folds = 5. The results thus obtained are tabulated in Table 4.

Run	Confusion Matrix	Recall	Precision	F1 Score	Accuracy	Rejection Rate
0	$\begin{bmatrix} 484 & 74 \\ 22 & 371 \end{bmatrix}$	0.9565	0.8674	0.9098	0.8991	0.1009
1	$\begin{bmatrix} 448 & 81 \\ 22 & 401 \end{bmatrix}$	0.9532	0.8469	0.8969	0.8918	0.1082
2	$\begin{bmatrix} 444 & 70 \\ 35 & 404 \end{bmatrix}$	0.9269	0.8638	0.8943	0.8898	0.1102
3	$\begin{bmatrix} 425 & 76 \\ 33 & 420 \end{bmatrix}$	0.9279	0.8483	0.8863	0.8857	0.1143
4	$\begin{bmatrix} 460 & 95 \\ 25 & 373 \end{bmatrix}$	0.9485	0.8288	0.8846	0.8741	0.1259

Table 4: Performance Metrics for Different folds

Hence, we average these metrics and obtain the following statistics:
Mean Recall: 0.9426
Mean Precision: 0.8510
Mean F1 Score: 0.8944
Mean Accuracy: 0.8881

Part 3(b):

We will take the best classifier obtained in the 5 folds and use it to test on the test dataset. We obtain the following performance metrics as a result:
Accuracy = 0.87375
Number of rejected samples = 7

Approach 2:

In this approach we assume the features to be not be independent. We model both the classes by multivariate normal distributions with mean vector and covariance matrix calculated

as the sample mean and sample covariance matrix. An issue that had arised was that the covariance matrix thus obtained was singular. Thus, to invert it we use the Moore Penrose pseudo-inverse. To calculate the determinant of this matrix we approximate the determinant as the product of the non zero eigenvalues of the matrix. A better approach would be to add a regularization constant such as $10^{-6} * I$. However, even after adding such a regularization constant, my covariance matrices remained non singular. Hence, I am instead approximating the determinnnt using product of non zero eigenvalues.The results thus obtained are displayed in the following section.

Result:

Part 1:

We train the modified Bayes Classifier on the train set with different values of epsilon 0.01, 0.1, 0.25, 0.4. We then run it on the test dataset of size 800 and find the misclassification loss and the number of rejected samples. We also calculate the accuracy of the classifier.

ϵ	0.01	0.1	0.25	0.4
Accuracy	0.8375	0.8375	0.8375	0.8375
Rejected Samples	115	115	115	115
Misclassification Loss	0.01875	0.01875	0.01875	0.01875

Table 5: Performance metrics for different values of ϵ using approach 2

Part 2:

We modify the dataset by subsampling to create different class priors of 60-40, 80-20, 90-10, 99-1. We then train the Modified Bayes Classifier on the train set under these modified priors for different values of epsilon 0.1, 0.25, 0.4. After training, we evaluate the classifier on the test dataset and report the misclassification loss among the nonrejected samples and the number of rejected samples.

Prior Split	Misclassification Loss			Rejected Samples		
	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0.25$	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0.25$
60-40	0.0225	0.0225	0.0225	179	179	179
80-20	0.0375	0.0375	0.0375	277	277	277
90-10	0.06	0.06	0.06	11	11	11
99-1	0.053	0.053	0.053	11	11	11

Table 6: Misclassification loss and number of rejected samples for different prior splits and ϵ values (approach 2).

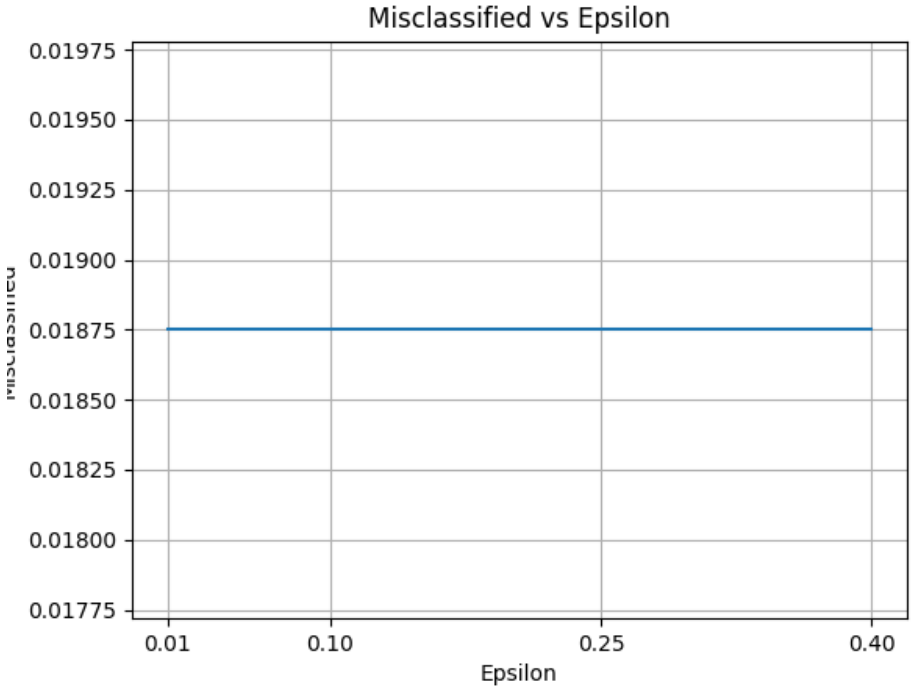

Figure 11: misclassification loss vs epsilon using approach 2

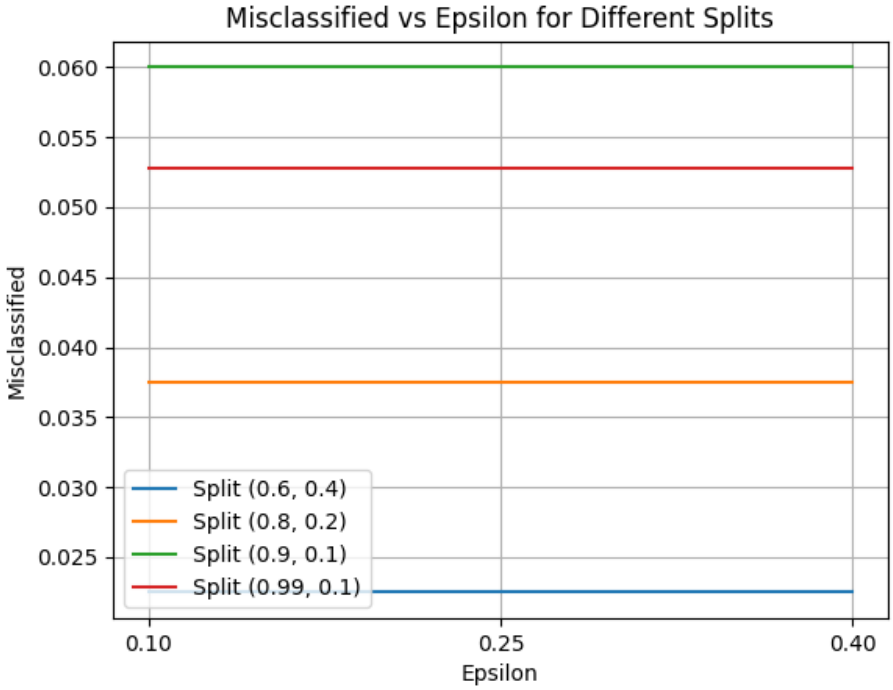

Figure 12: misclassification loss vs epsilon using approach 2 and different splits

Part 3(a):

We now perform a k-fold classification of the data using the number of folds = 5. The results thus obtained are tabulated in Table 7.

Run	Confusion Matrix	Recall	Precision	F1 Score	Accuracy	Rejection Rate
0	$\begin{bmatrix} 405 & 4 \\ 20 & 359 \end{bmatrix}$	0.9529	0.9902	0.9712	0.9695	0.0305
1	$\begin{bmatrix} 368 & 7 \\ 10 & 358 \end{bmatrix}$	0.9735	0.9813	0.9774	0.9771	0.0229
2	$\begin{bmatrix} 363 & 4 \\ 6 & 359 \end{bmatrix}$	0.9837	0.9891	0.9864	0.9863	0.0137
3	$\begin{bmatrix} 351 & 5 \\ 5 & 375 \end{bmatrix}$	0.9860	0.9860	0.9860	0.9864	0.0136
4	$\begin{bmatrix} 379 & 4 \\ 9 & 366 \end{bmatrix}$	0.9768	0.9896	0.9831	0.9828	0.0172

Table 7: Classification Metrics for Different Runs

Part 3(b):

We will take the best classifier obtained in the 5 folds and use it to test on the test dataset. We obtain the following performance metrics as a result:
Accuracy = 0.9695
Number of rejected samples = 57

Question 3: Decision Trees

Procedure:

The hyperparameters for the Decision Tree classifier were obtained by querying an oracle. The hyperparameters were:

- **Criterion**: Entropy
- **Splitter**: Best
- **Max Depth**: 5

We use the `processed.cleveland.data` dataset from the UCI Heart Disease dataset. This is the only dataset among the UCI Heart Disease datasets that contains 303 data points, each with 14 features.

Dataset Description

The dataset features are as follows:

Feature	Description
age	Age of the patient
sex	Gender (0 = female, 1 = male)
cp	Chest pain type
trestbps	Resting blood pressure
chol	Serum cholesterol level
fbs	Fasting blood sugar
restecg	Resting electrocardiographic results
thalach	Maximum heart rate achieved
exang	Exercise-induced angina
oldpeak	ST depression induced by exercise
slope	Slope of the peak exercise ST segment
ca	Number of major vessels colored by fluoroscopy
thal	Thalassemia status
goal	Target variable (0 = no heart disease, 1-4 = increasing severity)

Since our objective is to classify whether a patient has heart disease or not, we modify the `goal` column such that all non-zero values are changed to 1, while zero values remain unchanged. This effectively transforms the problem into a binary classification task.

Handling Missing Values

The dataset contains missing values, represented by '?', which appear only in columns 11 and 12. We handle these missing values as follows:

- If the missing value corresponds to a categorical column, we replace it with the mode of the column.
- If the missing value corresponds to a numerical column, we replace it with the median of the column.

Data Splitting and Model Training

After preprocessing the dataset, we split the features from the labels and create a training and test dataset. To ensure reproducibility, we specify a random seed.
The training data is then used to fit a Decision Tree classifier initialized with the hyper-parameters provided by the oracle. Finally, we evaluate the model by computing accuracy, F1-score, recall, precision.

Experimental results:

The decision tree visualised using *dtreeviz* is as shown in Figure 13:

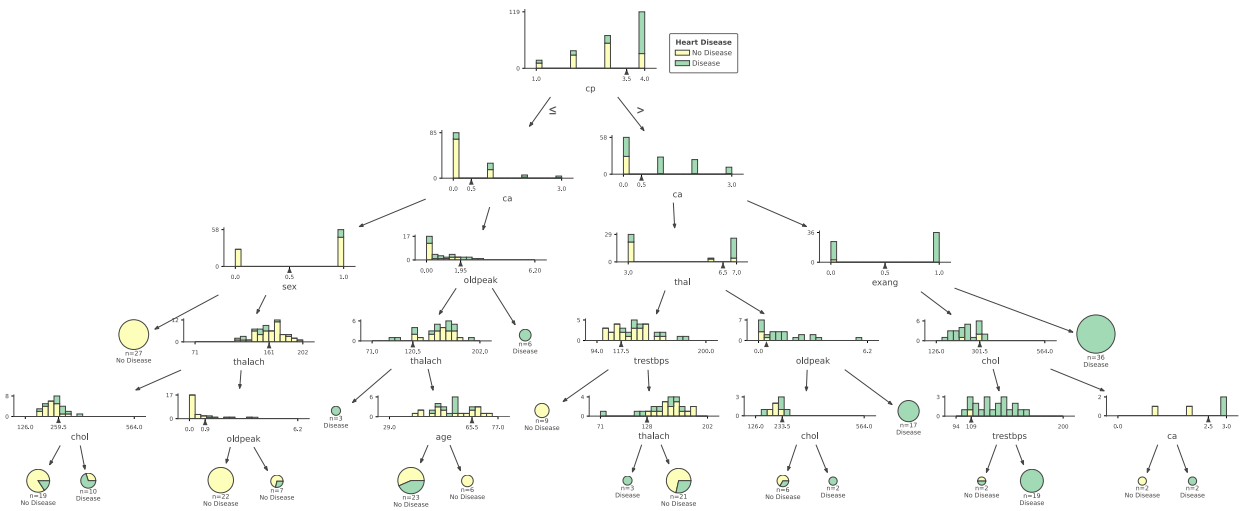

Figure 13: decision tree

The decision tree classifier has the following metrics to test it's efficacy:

$$\text{Accuracy} = \mathbf{0.9344}$$
$$\text{Precision} = \mathbf{0.9000}$$
$$\text{Recall} = \mathbf{0.9000}$$
$$\text{F1 Score} = \mathbf{0.9000}$$

Based on the decision tree, we can conclude **'chest pain type'** abbreviated as **'cp'** in the columns is the most important feature for predicting heart disease.

References:

- Pattern Classification, Duda Hart
- Belhumeur, Peter Hespanha, Joao Kriegman, David. (1997). Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Trans. Pattern Anal. Mach. Intell.. 19. 711-720. 10.1109/34.598228.