EE6227 Assignment 1: Bayes Classifiers and LDA

Xueyi Ke
Matric G2303895E
xke001@e.ntu.edu.sg

Kezhi Mao
Assoc. Prof.
ekzmao@ntu.edu.sg

***Abstract*—In this coursework study, we will employ three classifiers, utilizing a provided labeled training dataset, to predict outcomes on a test dataset. By implementing classifiers based on Bayes Decision Rules, Naive Bayes, and Linear Discriminant Analysis(LDA) respectively, we will obtain and compare the results of the given test dataset among these three classifiers. Due to the absence of labels on the test set, we are unable to conduct further evaluation of these models. However, the primary aim of this study is to generate results and elucidate the theory behind these machine learning classifiers. We hope that this practice and study will not only demonstrate what we have learned in course EE6227 but also facilitate our research and studies in the field of machine learning. Code and results have been made available.**[1]

***Index Terms*—supervised classification, Bayes decision rules, naive Bayes, linear discriminate analysis**

DISCLAIMER

This paper incorporates GPT as a tool for enhancing its composition and visual presentations. It is emphasized that all core code was *independently* developed by the author.

I. INTRODUCTION

Machine learning, an integral part of artificial intelligence, leverages algorithms to interpret, predict, and classify data. Among these algorithms, classifiers play a pivotal role in pattern recognition and decision-making processes. This study focuses on three foundational classifiers:

The Bayes Decision Rule, rooted in Bayesian probability theory, offers a framework for decision-making under uncertainty. Proposed by Thomas Bayes in the 18th century, it has evolved into a fundamental principle in statistical inference and machine learning [1].

The Naive Bayes classifier, a simplification of Bayesian models, assumes independence among predictors. Despite its simplicity, Naive Bayes remains effective for numerous applications and was popularized in the 1950s for text classification and spam detection [2].

Linear Discriminant Analysis, introduced by R.A. Fisher in 1936, is another cornerstone of statistical classification [3]. By finding a linear combination of features that best separates two or more classes, LDA has been extensively used in the fields of biology, finance, and beyond for its simplicity and efficacy.

This study employs these three classifiers to predict outcomes on a test dataset using a labeled training dataset. While our evaluation is limited by the absence of labels in the test set, our primary aim is to elucidate the theoretical underpinnings of these classifiers. Through this exploration, we hope to enhance our understanding of machine learning concepts taught in course EE6227.

[1]https://github.com/Kexueyi/EE6227-ML-Assignment

II. METHODOLOGY

In this section, we will introduce the theory behind these three classifiers and explain how we implemented them from scratch **without** relying on any off-the-shelf composed packages.

Noted that we assume that the provided training data are *Gaussian distributed* in the implementation of the Bayes Decision Rule and Naive Bayes classifiers. Additionally, all code is written in *Python* to facilitate plotting and visualization.

A. Dataset and Preprocess

1) Dataset Information: The training dataset comprises 420 samples, each with 5 features, and divided into two classes, labeled as 1 and -1, respectively. Similarly, the test dataset contains 26 samples, also characterized by 5 features, prepared for binary classification. Figure 1 illustrates the equal distribution of the two classes, indicating no imbalance issues.

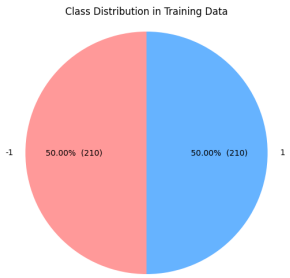


Fig. 1: Equal class distribution in the training set.

Given that all training data points exist in a 5-dimensional space, visualizing them poses a significant challenge. To achieve a better understanding of the data, we propose two approaches:

- Select the first three features and visualize them in a 3D scatter plot, which is the maximum dimensionality feasible for visualization (Figure 2).
- Employing the t-SNE technique [4] facilitated a preliminary visualization and succinct overview of the data, as illustrated in Figure 3. Notably, this approach has already enabled the preliminary delineation of class decision boundaries, given the clear separation observed among samples.

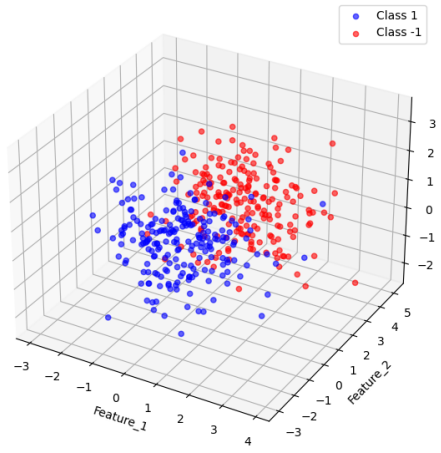


Fig. 2: Simplified 3D visualization of the training set, with axes 0, 1, 2 corresponding to features 1, 2, 3 respectively.

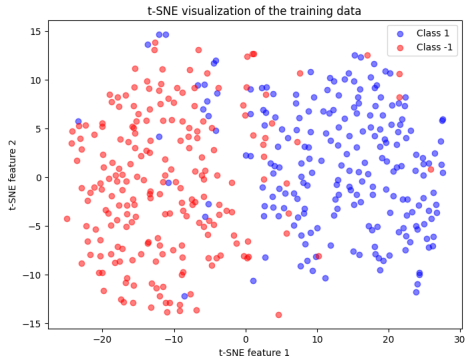


Fig. 3: t-SNE visualization of the training set.

2) Preprocessing: The datasets are provided in *.mat MATLAB* file format. Since the assignment didn't specify the use of a particular programming language, we have chosen *Python* for further analysis. So we need few preprocessing steps, transforming .mat file into dataframe in *numpy*.

```
# Load .mat data
data_train =
↪  loadmat('./data_train.mat')['data_train']
label_train =
↪  loadmat('./label_train.mat')['label_train']
data_test = loadmat('./data_test.mat')['data_test']

# Turn into dataframes
df_train = pd.DataFrame(data_train,
↪  columns=[f'Feature_{i+1}' for i in
↪  range(data_train.shape[1])])
df_label = pd.DataFrame(label_train,
↪  columns=['Label'])
df_test = pd.DataFrame(data_test,
↪  columns=[f'Feature_{i+1}' for i in
↪  range(data_test.shape[1])])
```

Then, we will apply these 3 different classifiers for supervised binary classification.

B. Bayes Decision Rule

The Bayes Decision Rule is pivotal in machine learning for classification tasks, guiding the selection of the most probable class given the observed features.

1) Formulation: Formally, the Bayes Decision Rule is articulated as:

$$\text{Choose } w_i \text{ if } P(w_i|x) > P(w_j|x) \text{ for all } j \neq i,$$

where $P(w_i|x)$ is the posterior probability of class w_i given features x, computed via Bayes' Theorem:

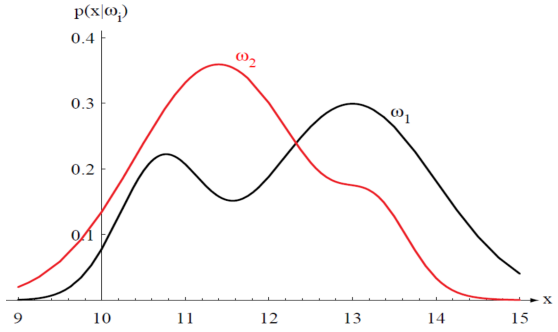
$$P(w_i|x) = \frac{P(x|w_i)P(w_i)}{P(x)}.$$

Here, $P(x|w_i)$ is the class conditional probability, $P(w_i)$ the prior probability of class w_i, and $P(x)$ the evidence or marginal likelihood of observing x.

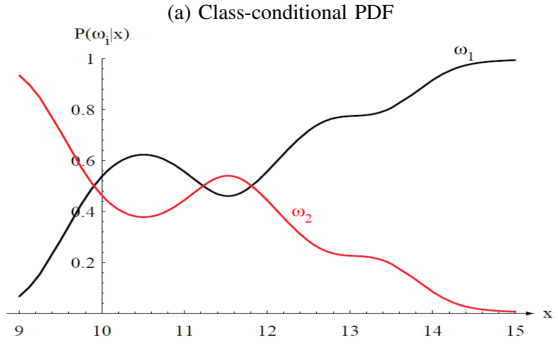
In the case of multivariate normal distribution, which in our dataset is 5 features as multi-variables, the class conditional probability $P(x|w_i)$ can be expressed using the multivariate normal density function:

$$P(x|w_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right),$$

where d is the dimensionality of the feature space, μ_i is the mean vector of class w_i, and Σ_i is the covariance matrix of class w_i.



(a) Class-conditional PDF



(b) Posterior PDF

Fig. 4: An example PDF of 2 class, assuming prior probabilities are $2/3$ and $1/3$ respectively [5]

2) Implementation: The central concept of Bayes Decision Rule can be encapsulated in the following implementation:

```
def class_conditional_prob(x, mean, cov):
    d = len(mean)  # Dimension of the feature vector
    var = x - mean
    return (1 / np.sqrt((2 * np.pi)**d * np.linalg.det(cov))) * np.exp(-0.5 * var.T @ np.linalg.inv(cov) @ var)

def bayes_classifier(x, mean1, cov1, prior1, mean2, cov2, prior2):
    """A simple design for Bayes Decision Rule"""
    prob1 = class_conditional_prob(x, mean1, cov1) * prior1
    prob2 = class_conditional_prob(x, mean2, cov2) * prior2

    if prob1 > prob2:
        return 1 # Class 1
    elif prob1 < prob2:
        return -1 # Class 2
    else:
        return 0  # Right on the boundary
```

3) Visualization: Visualizing the decision boundary through scatter plots is a commonly employed technique for intuiting the separation between different classes. Unfortunately, in this study, our training set comprises five features, exceeding the three-dimensional limit that can be comfortably visualized by humans. As a result, direct visualization of the Probability Density Function (PDF) within the framework of Bayes' decision rules is not feasible. However, we will explore alternative visualization strategies in subsequent sections to gain insights into our model's performance.

C. Naive Bayes Classifier

The Naive Bayes classifier extends the Bayes Decision Rule to classification tasks with a simplified assumption: **feature independence** given the class label. This assumption distinguishes it from more general Bayesian decision-making models by significantly simplifying the computation of class conditional probabilities. However, this simplification comes with the drawback of being *Naive*, due to its strong assumption of feature independence.

1) Formulation: Unlike the general Bayes Decision Rule, which using covariance matrix Σ representing relationship within features, Naive Bayes assumes the features in x are conditionally independent given the class w:

$$P(x|w) = \prod_{k=1}^{n} P(x_k|w),$$

where $x = (x_1, x_2, ..., x_n)$ represents the feature vector, and $P(x_k|w)$ is the probability of feature x_k given class w.

Now the posterior probability $P(w|x)$ can now be computed more feasibly:

$$P(w|x) = \frac{\prod_{k=1}^{n} P(x_k|w)P(w)}{P(x)},$$

with $P(w)$ as the prior probability of class w, and $P(x)$ the evidence, typically calculated as a normalizing constant.

2) Implementation: The core idea behind Naive Bayes can be implemented as follows:

```
def gaussian_pdf(x, mean, std):
    return (1 / (np.sqrt(2 * np.pi) * std)) * np.exp(-((x - mean) ** 2 / (2 * std ** 2)))

def naive_bayes_classifier(stats, input_features):
    y_pred = []
    for i in range(input_features.shape[0]):
        row = input_features.iloc[i]
        probabilities = {}
        for label, params in stats.groupby(level=0):
            probabilities[label] = 1
            for feature in input_features.columns:
                feature_mean = params.loc[label, (feature, 'mean')]
                feature_std = params.loc[label, (feature, 'std')]
                # Maximum likelihood estimation
                probabilities[label] *= gaussian_pdf(row[feature], feature_mean, feature_std)
        y_pred.append(max(probabilities, key=probabilities.get))
    return y_pred
```

3) Visualization: In the context of Bayes Decision Rules, directly visualizing high-dimensional probability distributions is impractical. However, leveraging the Naive Bayes assumption that all features are independent, it becomes feasible to individually plot the Gaussian distribution of each set of five features, as illustrated in Figure 5. Furthermore, we can visualize the composite distribution used for classification, which is derived from the product of these individual distributions, as shown in Figure 6.

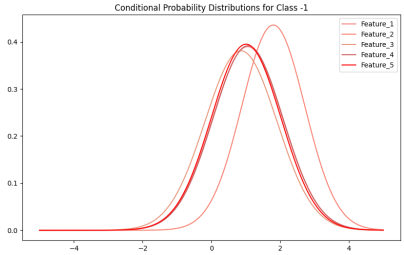
D. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a method used for dimensionality reduction and classification. In this study, we will particularly focus on Fisher Linear Discriminant (FLD). Simply say, it will trying to find a projection of samples on the hyperplane to be well separated, and with a discriminate function telling the sample belong to which class7.

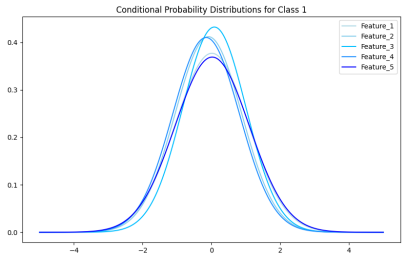
1) Formulation: Given a dataset with N samples and D features, let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ be the feature matrix where each $\mathbf{x}_i$ represents a sample, and $y = \{y_1, y_2, ..., y_N\}$ be the corresponding class labels.

The objective is to maximize the Fisher criterion:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$



(a) Gaussian Distribution for Class -1



(b) Gaussian Distribution for Class 1

Fig. 5: Visualization of the Gaussian distribution for all five features within the training set. Note that although these features are presented on the same axis for simplicity, they originate from differing dimensions.

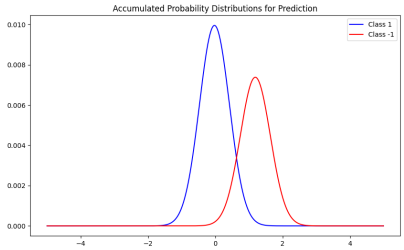


Fig. 6: The Gaussian distribution following the accumulation of multiplicative operations, with the vertical axis logarithmically scaled for easier visualization.

where $\mathbf{S}_B$ is the between-class scatter matrix and $\mathbf{S}_W$ is the within-class scatter matrix.

The optimal projection vector $\mathbf{w}$ is obtained by solving:

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}$$

Once obtained, samples are projected onto this vector for classification.

2) Implementation: The core concept of Linear Discriminant Analysis (LDA) for binary classification is succinctly captured in the *pseudo code* below, which outlines the key functions involved in the process. In our specific study, the target dimensionality (*n_components*) is set to one, implying that the feature space is reduced to a single dimension. This reduction is particularly suited for binary classification tasks. For a comprehensive view of the implementation, interested readers are directed to the repository link provided in the abstract.

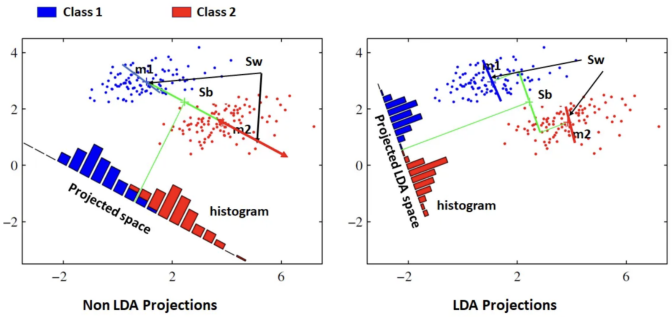


Fig. 7: An example explanation of LDA [6]

```
# train LDA model
def fit(self, X, y):
    class_means, overall_mean =
    ↪  self._compute_means(X, y)
    # calculate within-class scatter matrix and
    ↪  between-class scatter matrix
    S_W = self._compute_within_class_scatter_matri
    ↪  x(X, y,
    ↪  class_means)
    S_B = self._compute_between_class_scatter_matr
    ↪  ix(X, y, class_means,
    ↪  overall_mean)

    eig_vals, eig_vecs =
    ↪  self._solve_eigen_problem(S_W, S_B)

    # choose first n_components eigenvectors
    self.eig_vectors = eig_vecs[:,
    ↪  :self.n_components]
    self._compute_class_stats_lda(X, y)

# solve the eigenvalue problem by computing
↪  S_W^(-1)S_B
def _solve_eigen_problem(self, S_W, S_B):
    eig_vals, eig_vecs =
    ↪  np.linalg.eig(np.linalg.inv(S_W).dot(S_B))
    # sort eigenvalues and eigenvectors in
    ↪  descending order
    sorted_indices = np.argsort(-eig_vals)
    sorted_eig_vals = eig_vals[sorted_indices]
    sorted_eig_vecs = eig_vecs[:, sorted_indices]
    return sorted_eig_vals, sorted_eig_vecs

# transform samples into LDA space
def transform(self, X):
    return X.dot(self.eig_vectors)

def fit_transform(self, X, y):
    self.fit(X, y)
    return self.transform(X)

# Caculate discriminant function
def discriminant_function(self, X):
    X_lda = self.transform(X)
    discriminant_values = {}
    for cls, stats in self.class_stats_lda.items():
```

```
        diff = X_lda - stats['mean']
        discriminant_values[cls] = -0.5 * np.sum(d
        ↪  iff.dot(np.linalg.inv(stats['cov'])) *
        ↪  diff, axis=1) + np.log(stats['prior'])
    return discriminant_values

# predict test label based on discriminant function
def predict(self, X):
    discriminant_values =
    ↪  self.discriminant_function(X)
    classes = list(discriminant_values.keys())
    predicted_labels = [classes[i] for i in
    ↪  np.argmax(np.vstack(list(discriminant_valu
    ↪  es.values())),
    ↪  axis=0)]
    return predicted_labels
```

3) Visualization: LDA serves as an effective technique for projecting data into a lower-dimensional space, thereby enhancing the interpretability of transformed data samples. The decision boundary, orthogonal to the line connecting the means of the two classes, becomes evident in this reduced dimensionality. The one-dimensional plot depicted in Figure 8 seeks to illustrate the decision boundaries clearly.

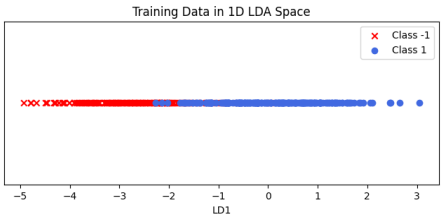


Fig. 8: Training samples scatter plot after LDA transformation

III. Results and Discussion

A. Results Analysis

Surprisingly, all three classifiers yielded identical classification results on our test dataset. The class distribution is illustrated in Figure 9. Even more astonishing is the uniformity across all classifiers regarding the classification of each test sample, as depicted in Figure 10. Specifically, for all three classifiers, samples from the 1st to the 15th were classified into class -1, and samples from the 16th to the 26th were classified as class 1. All results is included in repository from abstract link, under the folder ”Assignment-1-classifiers/results”

B. Discussion

This section presents discussion and also visualizations derived from the results of our test set across three classifiers. However, visualizing the Bayes decision rule classifier is difficult due to the dataset’s high dimensionality, as this method does not offer dimensionality reduction.

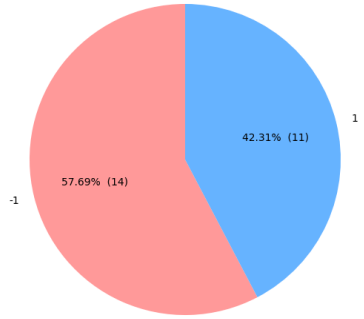


Fig. 9: Class distribution on test dataset

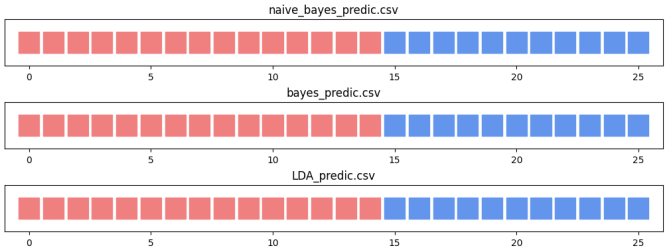


Fig. 10: Per-sample classification outcomes(indices start at 0)

1) Bayes Classifiers: For the Naive Bayes classifier, given its assumption that all features are uncorrelated, we employ a visualization approach similar to what was applied to the training set.

We illustrate the Gaussian distribution of each set of five features, as demonstrated in Figure 11. Additionally, we depict the composite distribution used for classification, derived from the product of these individual distributions, as shown in Figure 12.

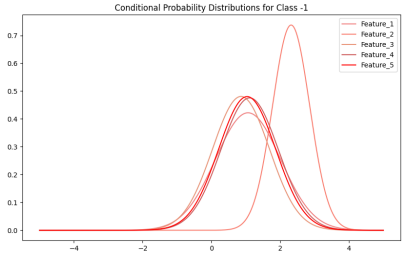
It is noteworthy that between the test and training sets, despite the mean of the final distribution remaining essentially similar after cumulative multiplication, there are distinct differences in the magnitude of the central peak: for Class -1, it is significantly smaller, whereas for Class 1, it becomes larger.

2) Linear Discriminate Analysis: The one-dimensional projection of the test set data in the LDA space, as shown in Figure 13, clearly demonstrates that all data points are distinctly separated, with no overlapping of boundaries. This clear separation facilitates the identification of the decision boundary, indicating that our trained LDA model performs effectively on the test dataset.

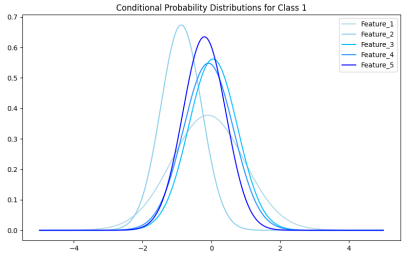
IV. Conclusion

This study applied three classifiers—Bayes Decision Rules, Naive Bayes, and Linear Discriminant Analysis (LDA)—to a test dataset. Remarkably, all classifiers yielded identical results, with a unanimous classification of samples into classes -1 and 1, as shown in Figures 9 and 10.

This unexpected uniformity across different classifiers suggests a potential inherent characteristic of the dataset or the robustness of the classifiers. However, the absence of test set labels limited further evaluation of the models’ accuracy.



(a) Gaussian Distribution for Class -1



(b) Gaussian Distribution for Class 1

Fig. 11: Visualization of the Gaussian distribution for all five features within the test set. Note that although these features are presented on the same axis for simplicity, they originate from differing dimensions.

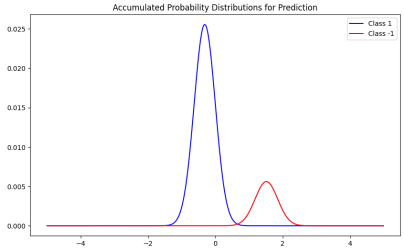


Fig. 12: The Gaussian distribution after cumulative multiplication on test set, with the vertical axis logarithmically scaled for simpler visualization.

In essence, this investigation not only meets the educational goals of course EE6227 but also paves the way for our future research in machine learning.

References

[1] Thomas Bayes and Richard Price. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions*, 53:370–418, 1763.

[2] M. E. Maron. Automatic indexing: An experimental inquiry. *Journal of the ACM*, 8(3):404–417, 1961.

[3] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[4] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[5] Mao Kezhi. The teaching slids from course ee6227. 2024.

[6] Christopher M Bishop. Pattern recognition and machine learning. *Springer google schola*, 2:5–43, 2006.

Appendix

Presented below are several questions and findings identified during this study.

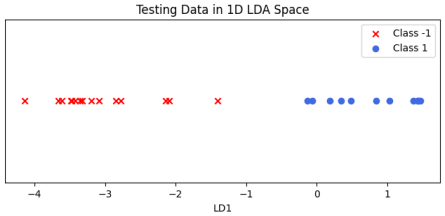


Fig. 13: Scatter plot of test samples after LDA transformation

A. *Is the Training Sample Gaussian Distributed?*

Bayes-based classifiers necessitate a Gaussian distribution. Given the impracticality of validating this across our dataset's five features, we focus on the first two features. A scatter plot with Gaussian distribution contours (Figure 14) illustrates their distribution, supporting our assumption.

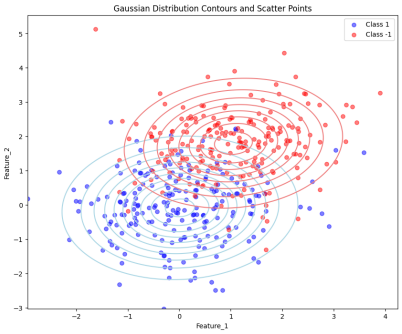
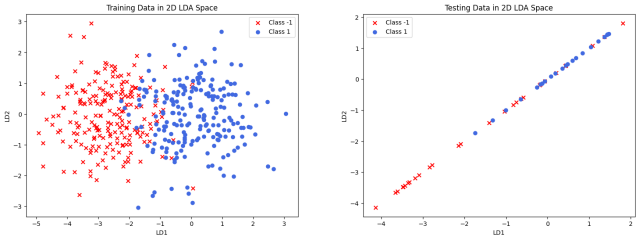


Fig. 14: Visualization of Gaussian Distribution using the first two features of the training set.

B. *Why is the Target Dimension in LDA Reduced to the Number of Classes Minus One?*

LDA inherently reduces dimensions to the number of classes minus one. Figures 15a and 15b show that representing data in a 2D space results in alignment along a straight line, indicating redundant dimensions and underscoring information redundancy.



(a) Training samples (b) Testing samples

Fig. 15: Visualization of 2D LDA space, demonstrating dimensionality redundancy.