CSE 569: Fundamentals of Statistical Learning and Pattern Recognition
Project 1 : Vatsal Gaurang Shah - 1229832502

Table of Contents

Introduction

The project involves the utilization of a modified subset of images derived from the MNIST dataset, which originally consists of 70,000 images of handwritten digits, divided into 60,000 training images and 10,000 testing images. The focus here is specifically on images of the digits "5" and "6," both of which have undergone slight modifications to align with the objectives of this project. The data is stored in ".mat" files.

Dataset Information:

	Digit 5	Digit 6
Training	5421	5918
Testing	892	958

For the classification task at hand, we maintain the assumption that the prior probabilities for both digits are equal, i.e., P(5) = P(6) = 0.5.

In the original .mat file, each image is stored as a 28x28 array. However, working in the 784-dimensional space, we encounter challenges when applying Bayesian decision theory, particularly for tasks like minimum error rate classification. To address this, we will employ Principal Component Analysis (PCA) as a dimensionality reduction technique before proceeding with the classification task.

Methodology

Task 1: Data Conditioning through Feature Normalization

Before starting, we have to perform data normalization to ensure consistent feature scaling on 784-dimensional vectorsin order to enhance the effectiveness of subsequent tasks.

To achieve this, the following steps must be executed:

Step 1: Compute Mean and Standard Deviation (STD)
Using all the training images, consider each image as a 784-dimensional vector (X) and calculate the mean (mi) and standard deviation (STD) (si) for each of the 784 features.

Step 2: Data Normalization
The computed mean (mi) and standard deviation (si) for each feature (xi) will be employed to normalize all data samples, including both training and testing data. For any given sample, each feature (xi) will be normalized using the following formula:
yi = (xi - mi) / si
This normalization process ensures that each feature is scaled appropriately, facilitating consistent and accurate analysis across all data samples.

Task 2: PCA using the training samples

In this task, we perform Principal Component Analysis on the training dataset.
We do this on our normalized training data to just work with smaller number of components/dimensions. This is done to ease our computation and allows us to not consider all 784 dimensions.
We explicitly code the steps for computing PCA and to do this, we first compute eigenvalues and covariance matrix.
We identify the principal components based on the sorting the eigenvalues in descending order.

Task 3: Dimension reduction using PCA

The primary objective is to explore 2-dimensional projections of the data samples on the first and second principal components, thereby creating new 2-dimensional representations of the samples. Subsequently, we engage in the visualization of these 2-dimensional representations, both for the training and testing datasets. The essence of this task is to create a visual representation of the data that has undergone dimensionality reduction. By projecting the data onto the first and second principal components, we aim to capture the most significant sources of variance in the dataset. Additionally, we scrutinize the visual representation to ascertain if each class exhibits characteristics are similar to a normal distribution.

Task 4: Density Estimation

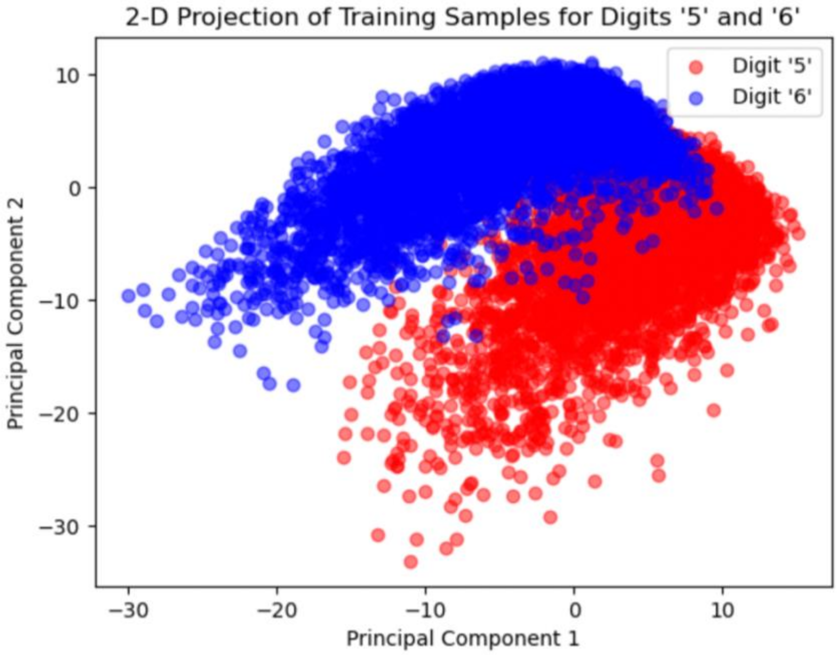
We estimate the mean and the covariance for both Digit 5 & Digit 6, as the mean & the covariance act as the parameters for the 2D Normal (Gaussian)distribution on training data.
We use the calculated mean & covariance for Task 5.

Task 5: Bayesian Decision Theory for optimal classification

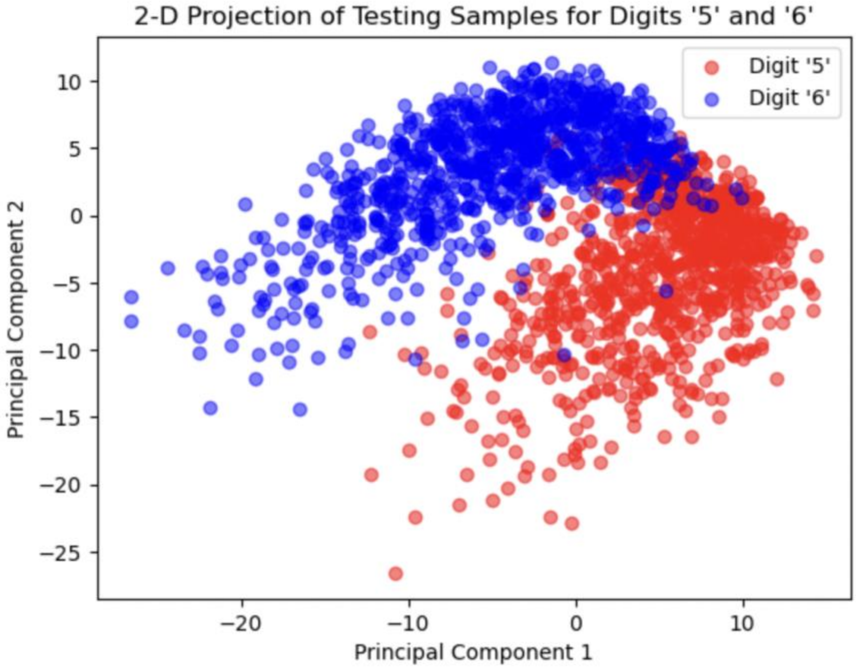
Utilize the estimated distributions to perform minimum-error-rate classification. This is done by calculated the Probability Density Function also known as PDF, and the classification is done on the basis of PDF. Whichever digit has a higher PDF, the image is classified under that digit. Eventually, we have to provide a report on the accuracy achieved for both the training and testing sets.

Result and observations.

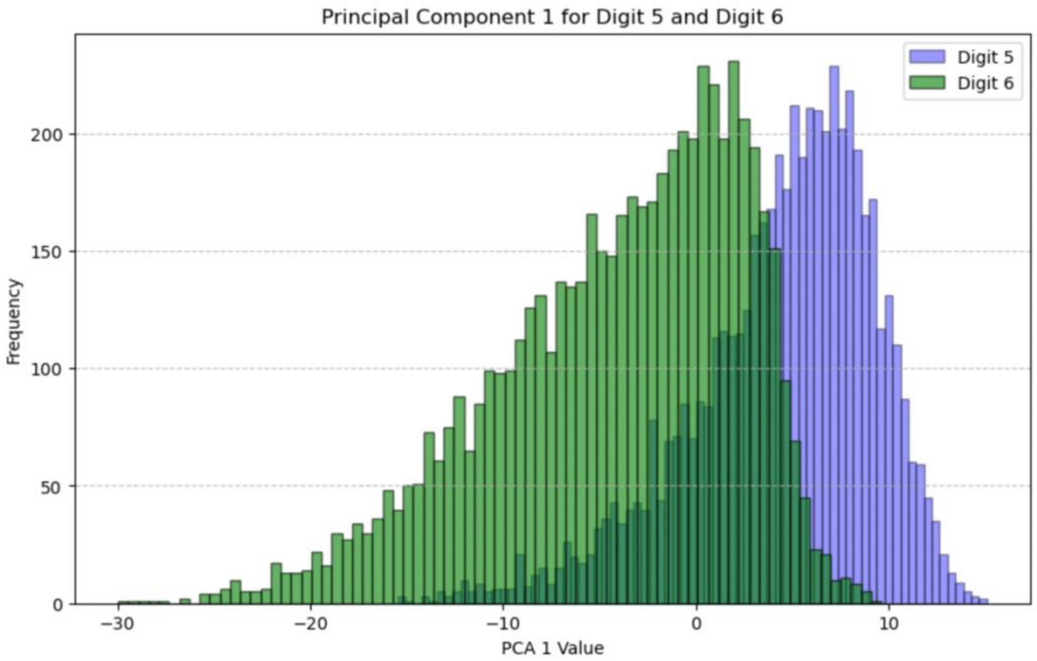
PCA Projection for Digit 5 & Digit 6 for Training Data



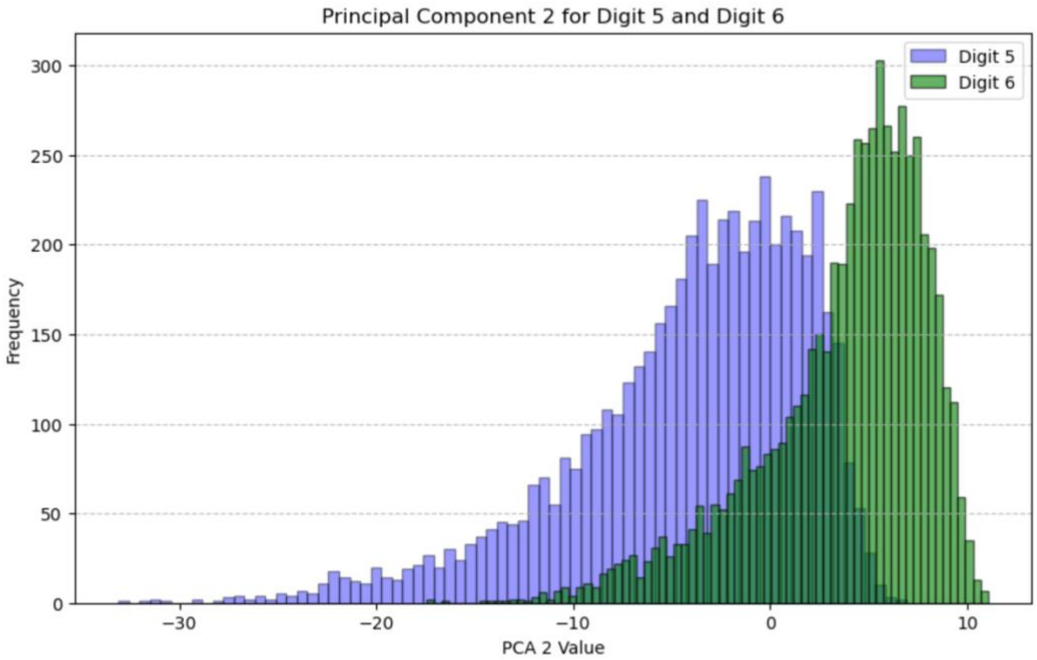
PCA Projection for Digit 5 & Digit 6 for Testing Data



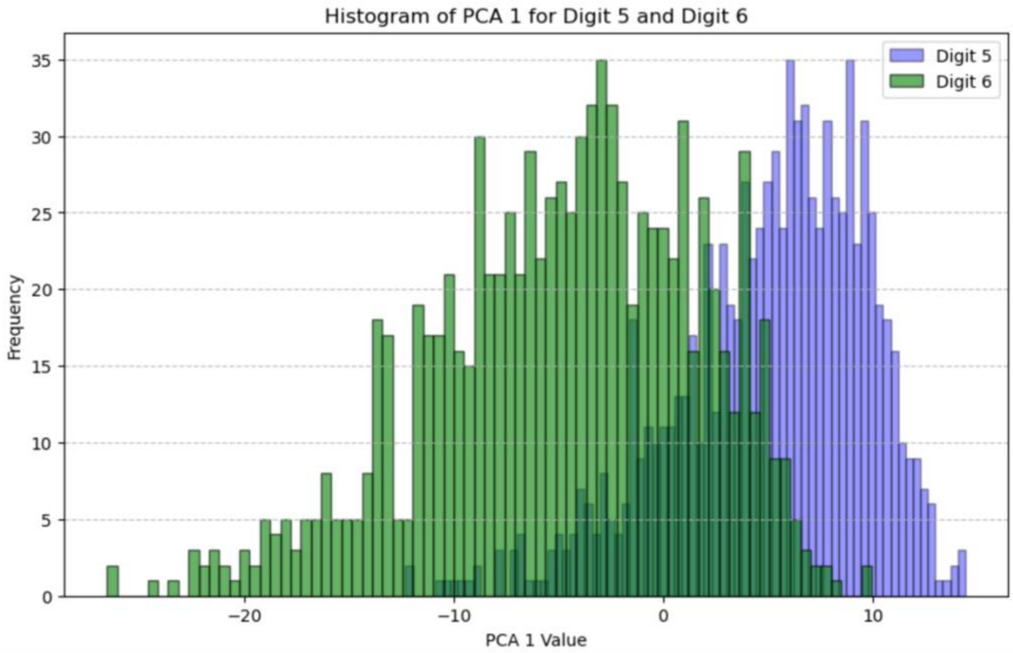
Histogram for Training Data for Digit 5 & 6 using PC-1 data



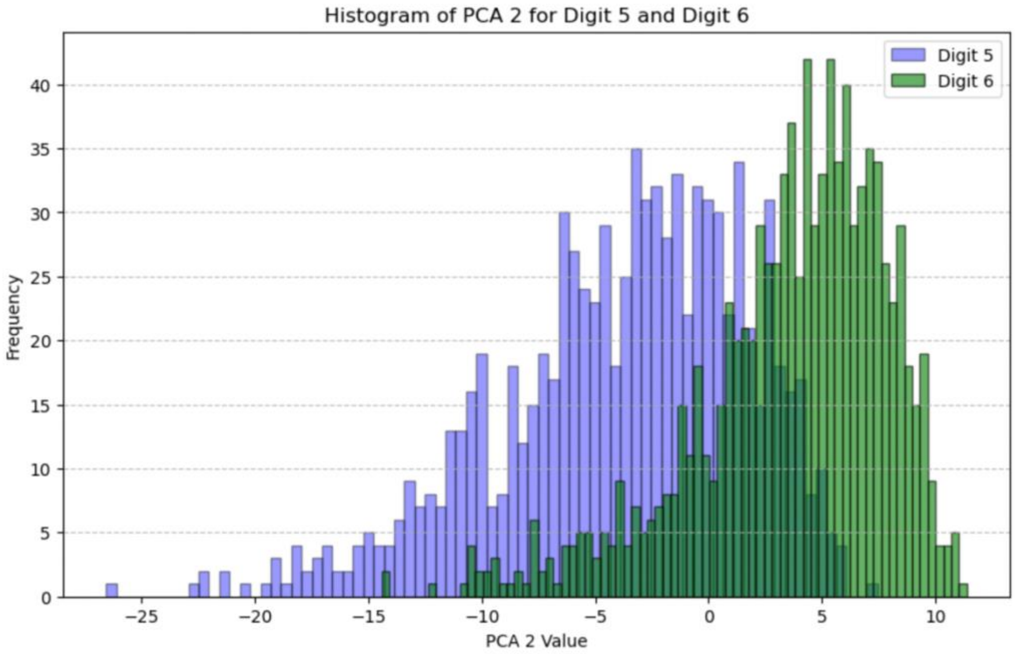
Histogram for Training Data for Digit 5 & 6 using PC-2 data



Histogram for Testing Data for Digit 5 & 6 using PC-1 data

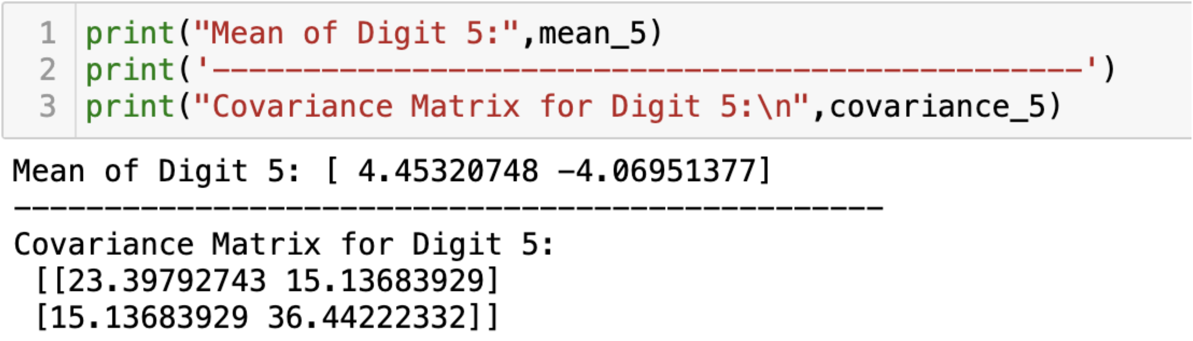


Histogram for Testing Data for Digit 5 & 6 using PC-2 data



Each class (Digit 5 & Digit 6) resemble a Normal distribution.

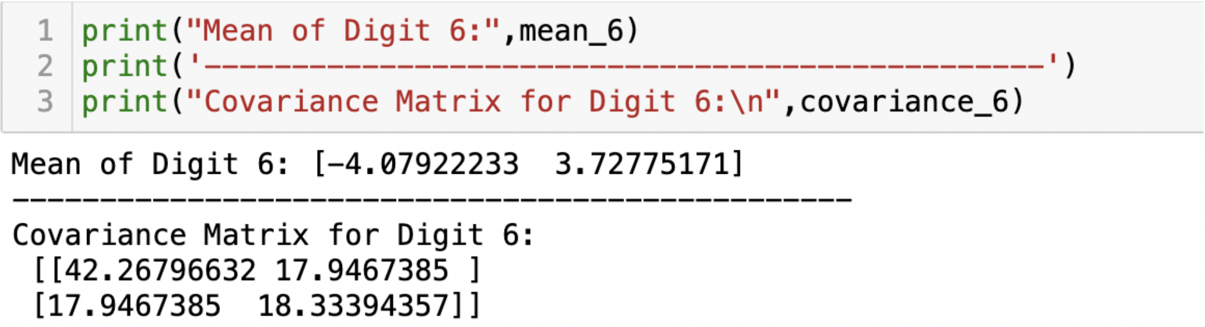
Estimated Mean & Covariance for Digit 5



```python
print("Mean of Digit 5:",mean_5)
print('-------------------------------------------------')
print("Covariance Matrix for Digit 5:\n",covariance_5)
```

```
Mean of Digit 5: [ 4.45320748 -4.06951377]
------------------------------------------------
Covariance Matrix for Digit 5:
 [[23.39792743 15.13683929]
 [15.13683929 36.44222332]]
```

Estimated Mean & Covariance for Digit 6

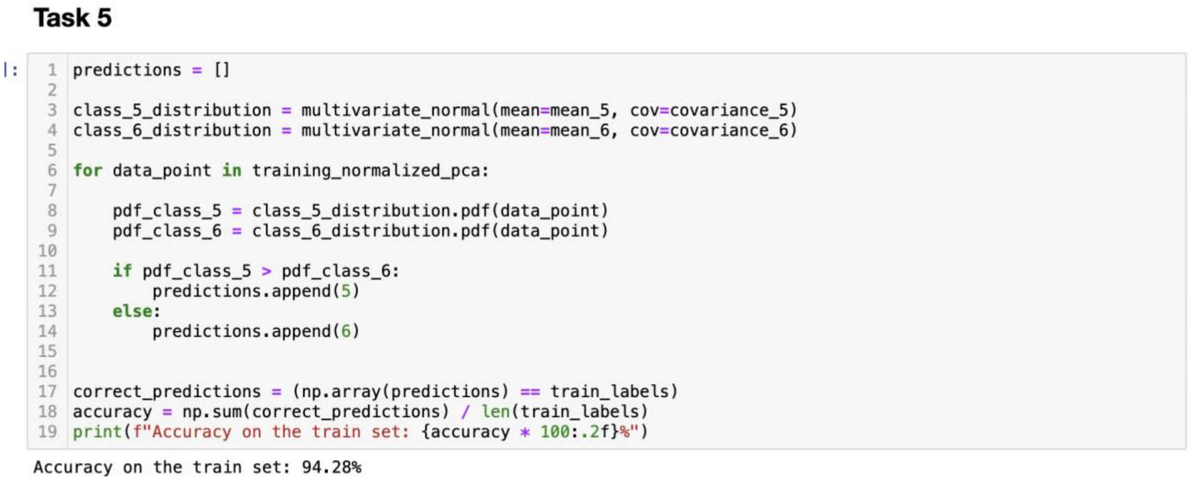


```python
print("Mean of Digit 6:",mean_6)
print('-------------------------------------------------')
print("Covariance Matrix for Digit 6:\n",covariance_6)
```

```
Mean of Digit 6: [-4.07922233  3.72775171]
------------------------------------------------
Covariance Matrix for Digit 6:
 [[42.26796632 17.9467385 ]
 [17.9467385  18.33394357]]
```

Accuracy for Training Data

Training Accuracy: 94.28%



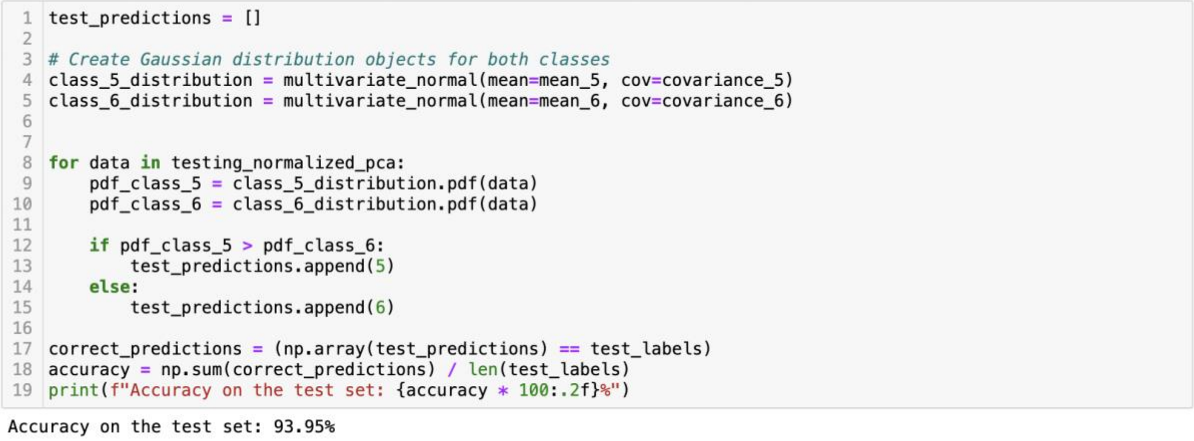
Task 5

```python
predictions = []

class_5_distribution = multivariate_normal(mean=mean_5, cov=covariance_5)
class_6_distribution = multivariate_normal(mean=mean_6, cov=covariance_6)

for data_point in training_normalized_pca:

    pdf_class_5 = class_5_distribution.pdf(data_point)
    pdf_class_6 = class_6_distribution.pdf(data_point)

    if pdf_class_5 > pdf_class_6:
        predictions.append(5)
    else:
        predictions.append(6)


correct_predictions = (np.array(predictions) == train_labels)
accuracy = np.sum(correct_predictions) / len(train_labels)
print(f"Accuracy on the train set: {accuracy * 100:.2f}%")
```

```
Accuracy on the train set: 94.28%
```

Accuracy for Testing Data

Testing Accuracy: 93.95%



```python
test_predictions = []

# Create Gaussian distribution objects for both classes
class_5_distribution = multivariate_normal(mean=mean_5, cov=covariance_5)
class_6_distribution = multivariate_normal(mean=mean_6, cov=covariance_6)


for data in testing_normalized_pca:
    pdf_class_5 = class_5_distribution.pdf(data)
    pdf_class_6 = class_6_distribution.pdf(data)

    if pdf_class_5 > pdf_class_6:
        test_predictions.append(5)
    else:
        test_predictions.append(6)

correct_predictions = (np.array(test_predictions) == test_labels)
accuracy = np.sum(correct_predictions) / len(test_labels)
print(f"Accuracy on the test set: {accuracy * 100:.2f}%")
```

```
Accuracy on the test set: 93.95%
```

Conclusion:

In this project, we performed a comprehensive image classification using a subset of the MNIST dataset, specifically focusing on handwritten digits "5" and "6." We began by adapting and normalizing the data to prepare it for analysis & then performing Principal Component Analysis to transform the original high-dimensional data into a more manageable form.

We then applied Bayesian Decision Theory for optimal classification. Leveraging the estimated distributions, we executed minimum-error-rate classification on both the training and testing sets. This classification approach, founded on the assumption of equal prior probabilities for both digits, allowed us to evaluate and report the accuracy of our classification model.