CS412 HW1 Barış Pome Report

Barış Pome - 31311

February 2025

Jupyter Notebook Link: Click here to access the notebook

1 Introduction

The primary objective of this project is to classify handwritten digits from the MNIST dataset using machine learning techniques. The MNIST dataset consists of grayscale images of handwritten digits (0-9), each of size 28×28 pixels, where each pixel value ranges from 0 to 255.

The methodology involves several critical stages, beginning with data analysis and preprocessing to prepare the dataset for model training. Following this, two classification models are applied: the k-Nearest Neighbors (k-NN) classifier and the Decision Tree classifier. Each model undergoes hyperparameter tuning to achieve the best possible performance.

The project evaluates the models based on several metrics, including accuracy, precision, recall, and F1-score. Furthermore, confusion matrices and Receiver Operating Characteristic (ROC) curves are generated to assess the classification performance more comprehensively. An analysis of misclassifications is conducted to highlight frequent errors and provide potential explanations.

The overall goal of this report is to document the methodology, present the results, and compare the performances of different classifiers. All experimental results are supported by tables, plots, and visualizations generated during the evaluation process.

2 Dataset and Preprocessing

2.1 Data Loading

The MNIST dataset, consisting of 28×28 grayscale images of handwritten digits ranging from 0 to 9, was loaded using the Keras API. Each pixel value in the dataset ranges from 0 to 255.

The dataset was split into three subsets as follows:

- **Training Set**: 80% of the original training data.

- **Validation Set**: 20% of the original training data, used for hyperparameter tuning.
- **Test Set**: The provided test set, used for evaluating final model performance.

The shapes of the training, validation, and test sets were printed to verify the splits:

- Training Set: 48,000 images
- Validation Set: 12,000 images
- Test Set: 10,000 images

2.2 Data Analysis

Before preprocessing, exploratory data analysis was performed to better understand the dataset.

2.2.1 Class Distribution

The class distribution was analyzed to ensure the dataset was balanced. The distribution of digits across the dataset was visualized using a bar plot (Figure 1). The dataset was found to be evenly distributed across all classes.

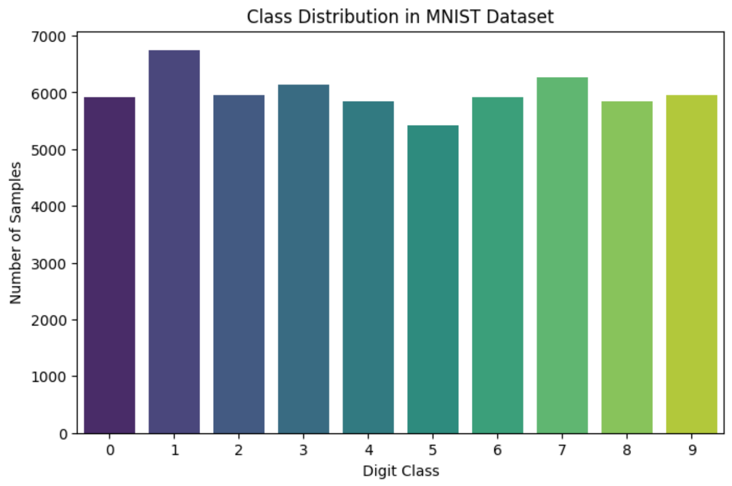


Figure 1: Class distribution of the MNIST dataset.

2.2.2 Basic Statistics

The mean and standard deviation of pixel values were calculated across the dataset:

- **Mean Pixel Value**: 33.32
- **Standard Deviation**: 78.57

Additionally, class-wise statistics were computed to analyze variations across different digits. These results confirmed that all classes were similarly distributed in terms of pixel intensity values.

2.2.3 Visualization of Sample Images

One sample image from each digit class was visualized to verify the dataset's integrity and confirm that all digits were represented accurately (Figure 2).

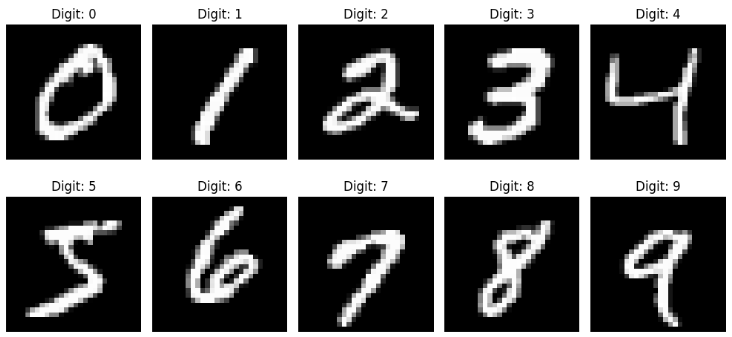


Figure 2: Sample images from the MNIST dataset representing each digit class.

2.3 Data Preprocessing

Prior to model training, the following preprocessing steps were applied:

- **Reshaping**: Each image was reshaped from a 2D array of shape $(28, 28)$ to a 1D array of 784 pixels for compatibility with machine learning algorithms.
- **Normalization**: Pixel values were scaled to the range [0, 1] using `StandardScaler` from Scikit-Learn, which transforms the data to have a mean of 0 and a standard deviation of 1.

The dataset was split into training, validation, and test sets after normalization to ensure consistent preprocessing across all subsets.

3 k-NN Classifier

3.1 Model Initialization and Hyperparameter Tuning

The k-Nearest Neighbors (k-NN) algorithm was applied to classify handwritten digits from the MNIST dataset. The main hyperparameter of the k-NN algorithm is the number of neighbors (k). The following values were tested:

$$k = 1, 3, 5, 7, 9$$

Each model was trained using the training set and evaluated on the validation set. Table 1 presents the validation accuracies for each k value.

Number of Neighbors (k)	**Validation Accuracy**
1	0.9704
3	0.9703
5	0.9675
7	0.9663
9	0.9654

Table 1: Validation accuracy of k-NN classifier for different values of k.

The best validation accuracy of 0.9704 was achieved with $k = 1$. Therefore, $k = 1$ was selected for the final model. Figure 3 shows how the validation accuracy varied across different values of k.

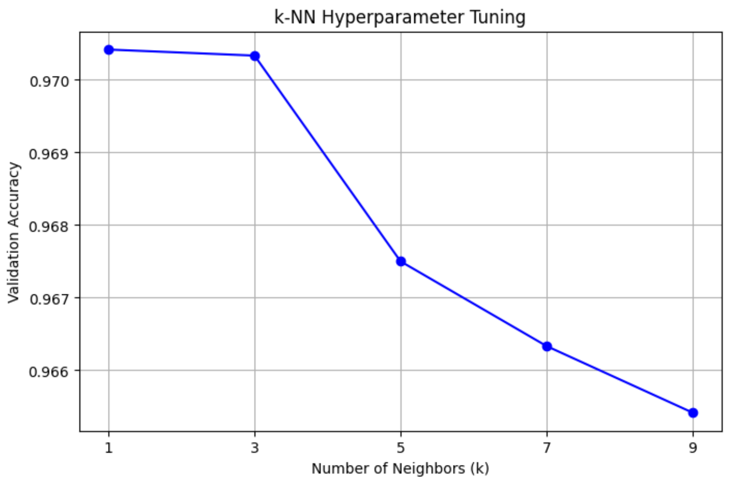


Figure 3: Validation accuracy of the k-NN classifier for different values of k.

3.2 Final Model Training and Evaluation

Using the optimal hyperparameter ($k = 1$), the final k-NN classifier was retrained on the combined training and validation sets and then evaluated on the test set.

3.2.1 Performance Metrics

The model's performance on the test set was evaluated using the following metrics:

- **Accuracy**: The overall proportion of correctly classified images.
- **Precision**: The proportion of positive identifications that were correct.
- **Recall**: The proportion of actual positives that were correctly identified.
- **F1-Score**: The harmonic mean of precision and recall.

The overall test accuracy of the k-NN classifier was **0.9691**. The detailed classification report is presented in Table 2.

Digit	Precision	Recall	F1-Score	Support
0	0.98	0.99	0.99	980
1	0.97	0.99	0.98	1135
2	0.98	0.96	0.97	1032
3	0.96	0.96	0.96	1010
4	0.97	0.96	0.97	982
5	0.95	0.96	0.96	892
6	0.98	0.99	0.98	958
7	0.96	0.96	0.96	1028
8	0.98	0.94	0.96	974
9	0.96	0.96	0.96	1009

Table 2: Classification report for the k-NN classifier on the test set.

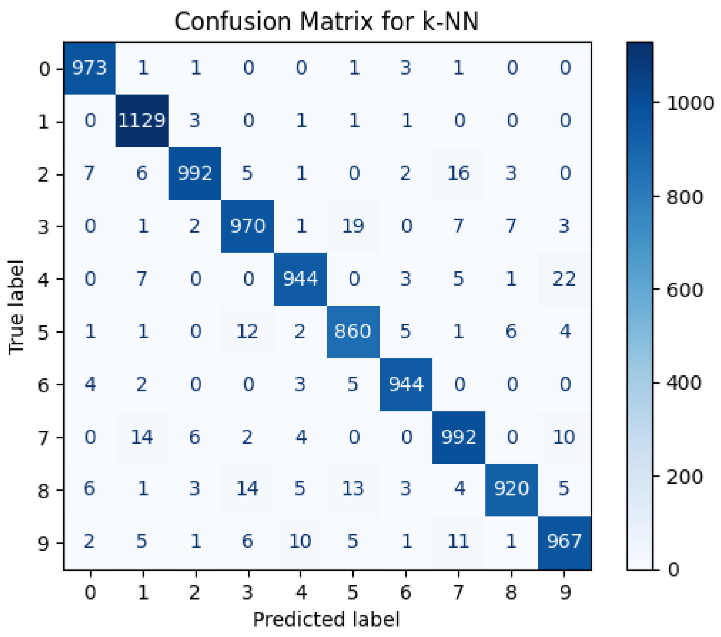
3.2.2 Confusion Matrix

The confusion matrix for the final k-NN classifier is shown in Figure 4. It highlights the model's performance across all classes.

3.2.3 Analysis of Misclassifications

The confusion matrix revealed several notable misclassifications. The most frequently confused digits share visual similarities, leading to classification errors:

- **Digit 4 misclassified as 9** (22 instances): A poorly written "4" may resemble a "9", especially when the top loop of "9" is not clearly formed.



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	973	1	1	0	0	1	3	1	0	0
1	0	1129	3	0	1	1	1	0	0	0
2	7	6	992	5	1	0	2	16	3	0
3	0	1	2	970	1	19	0	7	7	3
4	0	7	0	0	944	0	3	5	1	22
5	1	1	0	12	2	860	5	1	6	4
6	4	2	0	0	3	5	944	0	0	0
7	0	14	6	2	4	0	0	992	0	10
8	6	1	3	14	5	13	3	4	920	5
9	2	5	1	6	10	5	1	11	1	967

Figure 4: Confusion matrix for the k-NN classifier on the test set.

- **Digit 2 misclassified as 7** (16 instances): Similar visual features, such as the upper curve of ”2” resembling the crossbar of a ”7”, contribute to this misclassification.
- **Digit 3 misclassified as 5** (19 instances): The similarity arises from the rounded upper strokes and curves in handwritten digits.
- **Digit 8 misclassified as 3** (14 instances): The confusion is primarily due to the rounded structure of both digits, especially when the loops of the ”8” are not distinctly separated.
- **Digit 7 misclassified as 1** (14 instances): A handwritten ”7” without a crossbar can easily be mistaken for a ”1”.

Additionally, an analysis of the confusion matrix revealed the most frequently misclassified digits across the dataset:

- **Digit 3**
- **Digit 9**
- **Digit 8**

Visual examples of these misclassifications are presented in Figures 5 to 9. These images highlight the specific challenges faced by the classifier in distinguishing visually similar digits.

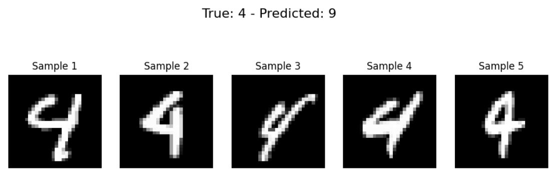


Figure 5: Examples of digit 4 misclassified as 9.

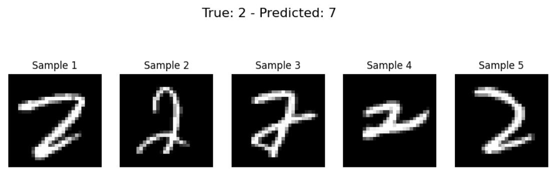


Figure 6: Examples of digit 2 misclassified as 7.

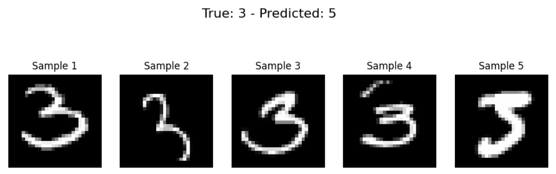


Figure 7: Examples of digit 3 misclassified as 5.

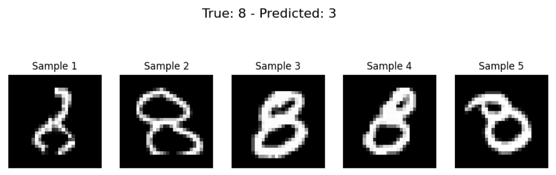


Figure 8: Examples of digit 8 misclassified as 3.

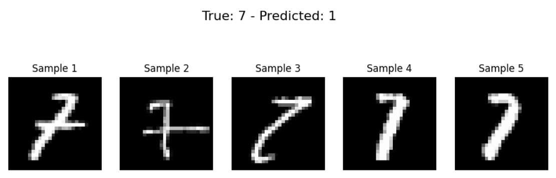


Figure 9: Examples of digit 7 misclassified as 1.

4 Decision Tree Classifier

4.1 Model Training and Hyperparameter Tuning

A Decision Tree classifier was applied to the MNIST dataset to classify handwritten digits. Two hyperparameters were tuned using grid search and 3-fold cross-validation:

- **max_depth**: Controls the maximum depth of the tree. Tested values: {2, 5, 10}.
- **min_samples_split**: Minimum number of samples required to split an internal node. Tested values: {2, 5}.

The best hyperparameters were found to be:

- `max_depth` = 10
- `min_samples_split` = 5

The best cross-validation accuracy achieved during the grid search was **0.8445**. Table 3 summarizes the validation accuracies for all tested hyperparameter combinations.

max_depth	min_samples_split	Validation Accuracy
10	5	0.8445
10	2	0.8443
5	2	0.6803
5	5	0.6803
2	2	0.3365
2	5	0.3365

Table 3: Validation accuracy for different hyperparameter combinations of the Decision Tree classifier.

4.2 Evaluation

Using the best hyperparameters, the final Decision Tree classifier was trained on the combined training and validation sets and evaluated on the test set.

4.2.1 Performance Metrics

The Decision Tree classifier achieved a test accuracy of **0.8608**. The classifier was evaluated using standard performance metrics such as accuracy, precision, recall, and F1-score. Table 4 presents the detailed results.

Digit	Precision	Recall	F1-Score	Support
0	0.93	0.93	0.93	980
1	0.95	0.94	0.94	1135
2	0.85	0.83	0.84	1032
3	0.83	0.82	0.82	1010
4	0.87	0.84	0.86	982
5	0.83	0.81	0.82	892
6	0.88	0.88	0.88	958
7	0.90	0.87	0.89	1028
8	0.79	0.78	0.79	974
9	0.78	0.88	0.83	1009

Table 4: Classification report for the Decision Tree classifier on the test set.

4.2.2 Confusion Matrix

The confusion matrix for the Decision Tree classifier is presented in Figure 10. This matrix highlights how well the model performed in distinguishing between the different digits.

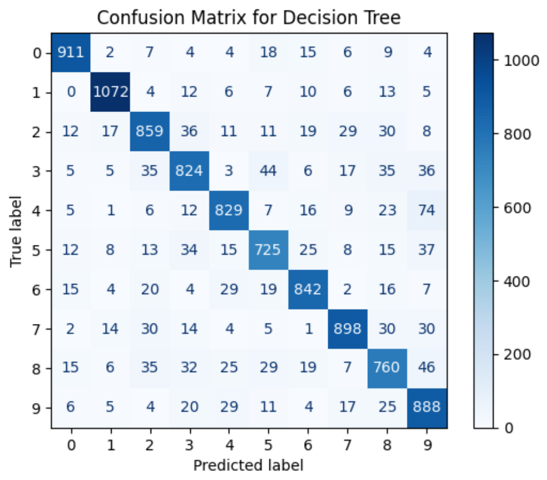


Figure 10: Confusion matrix for the Decision Tree classifier on the test set.

4.2.3 ROC Curve and AUC Scores

The Receiver Operating Characteristic (ROC) curve for each digit class was generated to evaluate the classifier's performance further. Figure 11 shows the ROC curves for all digit classes along with their AUC scores.

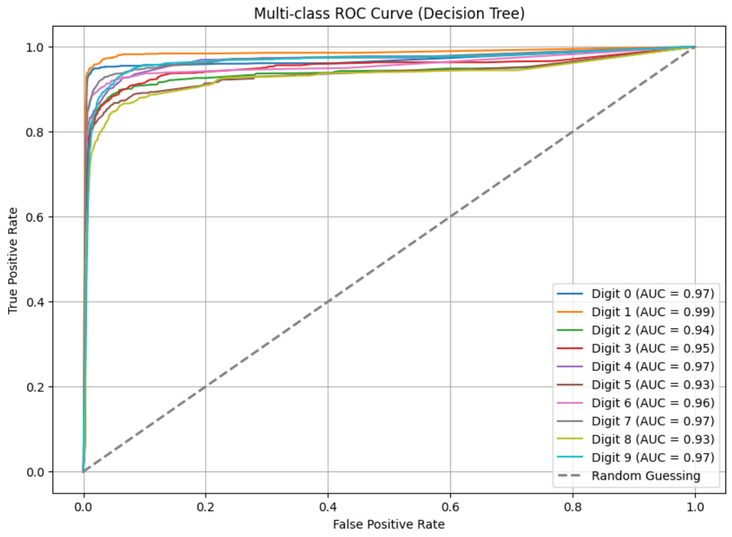


Figure 11: ROC curves for the Decision Tree classifier, including AUC scores for each digit.

4.2.4 Analysis of Misclassifications

The confusion matrix revealed common misclassifications that often occurred between visually similar digits:

- **Digit 2**: Frequently confused with 3 and 8, likely due to similarities in curved and straight-line features.
- **Digit 3**: Often misclassified as 5 or 9, potentially due to similar rounded stroke patterns.
- **Digit 8**: Commonly confused with 2 and 9 because of similarities in loop structures and round shapes.

An analysis of the confusion matrix also highlighted that the most frequently misclassified digits were:

- **Digit 2**
- **Digit 3**
- **Digit 8**

5 Conclusion

This project aimed to classify handwritten digits from the MNIST dataset using two machine learning algorithms: the k-Nearest Neighbors (k-NN) classifier and the Decision Tree classifier. Both models were thoroughly evaluated and compared in terms of accuracy, precision, recall, F1-score, and confusion matrices.

The k-NN classifier demonstrated superior performance compared to the Decision Tree classifier. After hyperparameter tuning, the best-performing k-NN model was achieved with $k = 1$, resulting in a high test accuracy of **96.91%**. The confusion matrix analysis revealed that most misclassifications occurred between digits with similar shapes, such as:

- Digit 4 misclassified as 9 (22 instances)
- Digit 2 misclassified as 7 (16 instances)
- Digit 3 misclassified as 5 (19 instances)
- Digit 8 misclassified as 3 (14 instances)
- Digit 7 misclassified as 1 (14 instances)

Additionally, the most frequently misclassified digits in the k-NN model were 3, 8, and 9, suggesting that digits with similar structural patterns posed challenges for the classifier.

In contrast, the Decision Tree classifier achieved a lower test accuracy of **86.08%**. The hyperparameter tuning process revealed that a maximum depth of 10 and a minimum sample split of 5 provided the best performance during cross-validation. The confusion matrix analysis indicated that the most frequent misclassifications involved digits:

- Digit 2: Frequently confused with 3 and 8
- Digit 3: Often misclassified as 5 or 9
- Digit 8: Commonly confused with 2 and 9

The ROC curve analysis further highlighted the performance gap between the two classifiers, with the k-NN classifier consistently outperforming the Decision Tree across all digit classes.

In conclusion, the k-NN classifier outperformed the Decision Tree classifier in classifying handwritten digits from the MNIST dataset. The simplicity and effectiveness of the k-NN algorithm, especially with a small value of k, allowed it to better capture the underlying patterns of the dataset. However, the Decision Tree classifier, despite being less accurate, offered valuable insights into feature importance and decision boundaries.