MNIST Classification Report: k-Nearest Neighbors and Decision Tree

KORHAN ERDOĞDU 30838

https://colab.research.google.com/drive/1WZMD7GYE_43UDqoKFwJorP8-j8rg0b2d?usp=sharing

1. Overview

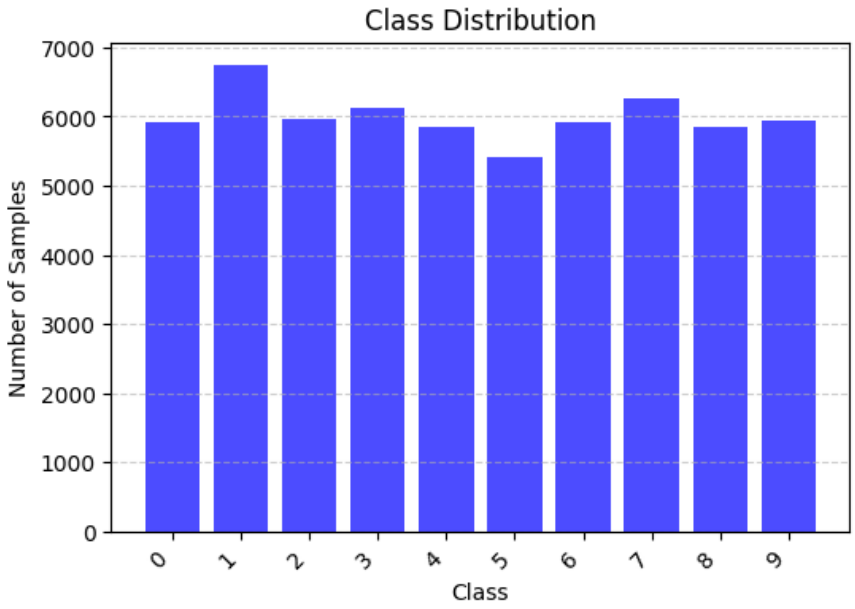
This report documents the implementation and evaluation of two classifiers—**k-Nearest Neighbors (k-NN)** and **Decision Tree**—on the MNIST dataset. The MNIST dataset consists of 28×28 grayscale images of handwritten digits (0–9). The aim was to preprocess the data, tune hyperparameters, and evaluate model performance using accuracy, precision, recall, F1-score, confusion matrices, and ROC curves.

2. Methodology

2.1 Data Analysis and Preprocessing

Data Loading and Initial Exploration

- **Dataset Source:** The MNIST dataset was loaded using the Keras API.
- **Data Split:**
  - **Training Set:** 80% of the original training data
  - **Validation Set:** 20% of the original training data
  - **Test Set:** The provided test set (unchanged)
- **Class Distribution:**
  We computed the frequency of each digit to ensure that the dataset is balanced.



- **Basic Statistics:**
  The minimum and maximum pixel values, mean, and standard deviation were calculated to understand the range and spread of the pixel intensities.
- **Visualization:**
  Sample images from each digit (0–9) were visualized to gain insight into the dataset.

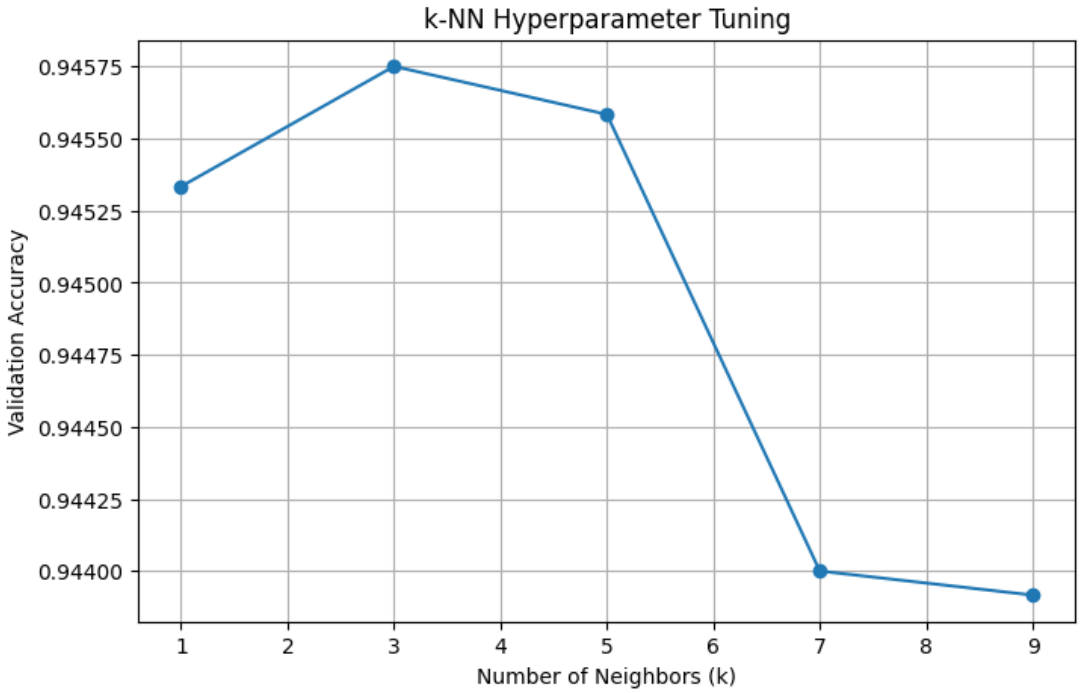
Preprocessing Steps

1. **Normalization:**
   Pixel values were scaled from [0, 255] to [0, 1] to standardize the input features.
2. **Flattening:**
   The normalized images were reshaped into one-dimensional arrays (size 784) since the classifiers expect 2D input (samples × features).
3. **Standardization:**
   A StandardScaler was applied to the flattened data. This step further standardized the features to have a mean of 0 and a standard deviation of 1, which can improve the performance of many machine learning algorithms.

2.2 Model Choices and Hyperparameter Tuning

k-Nearest Neighbors (k-NN)

- **Model Initialization:**
  The k-NN classifier was initialized with the Euclidean distance metric.
- **Hyperparameter Tuning:**
  The number of neighbors (k) was varied among the values [1, 3, 5, 7, 9]. Validation accuracy was measured for each value.
- **Justification:**
  The choice of k was based on the trade-off between bias and variance. A smaller k might be too sensitive to noise, while a larger k may oversmooth the decision boundary.
- **Final Training:**
  The best k was selected based on validation performance, and the classifier was retrained on the combined training and validation set before evaluation on the test set.



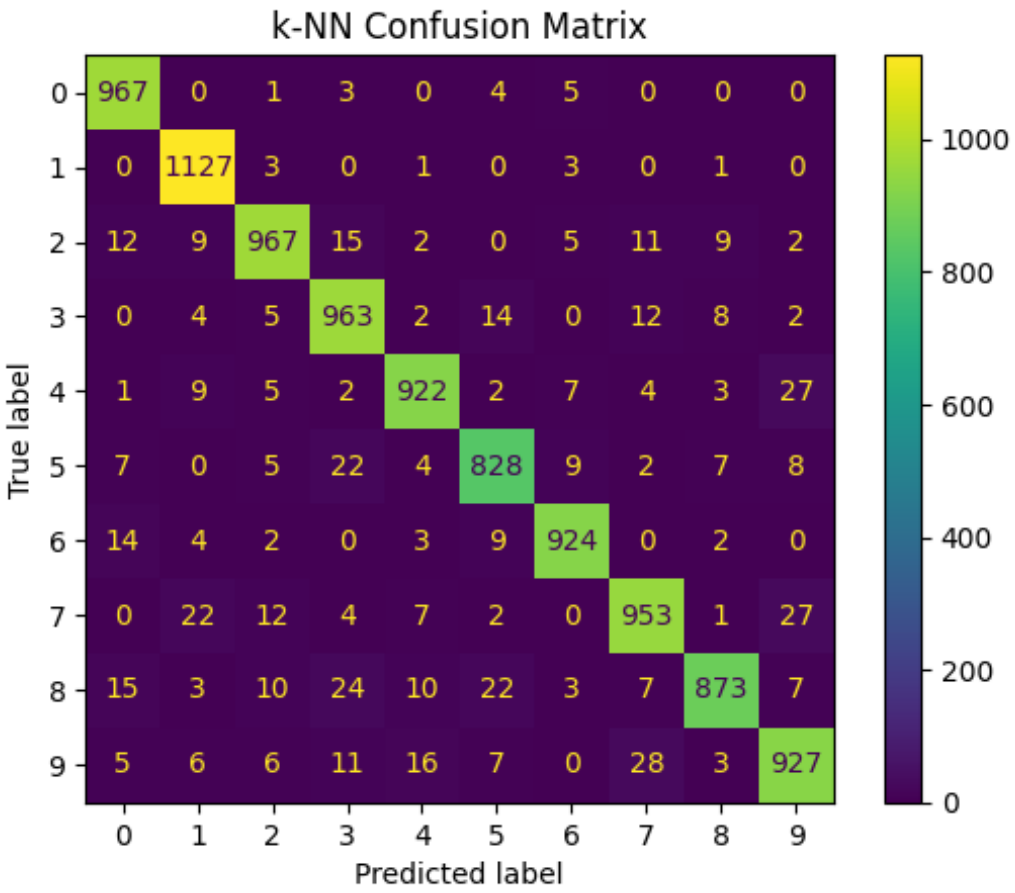
Decision Tree

- **Model Initialization:**
  A Decision Tree classifier was initialized with a fixed random seed for reproducibility.
- **Hyperparameter Tuning:**
  A grid search was performed over the parameters:
  - **max_depth:** [2, 5, 10]
  - **min_samples_split:** [2, 5]
- **Justification:**
  Tuning these parameters helped to control the complexity of the tree. A shallower tree (smaller max_depth) reduces overfitting, while a deeper tree might capture more intricate patterns at the risk of overfitting.
- **Final Training:**
  The best model parameters from grid search were used to train the final Decision Tree model on the full training set (training + validation).

3. Results

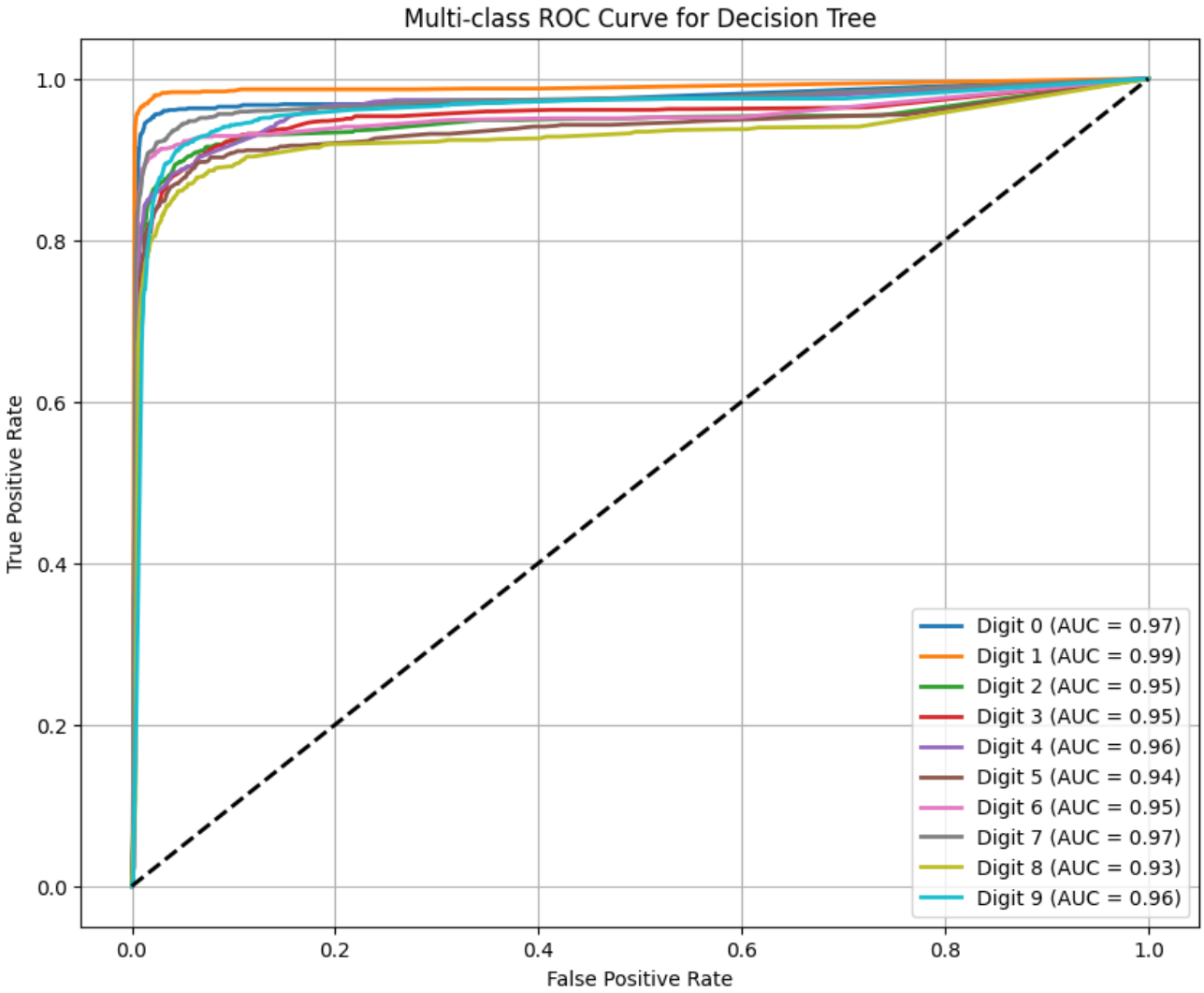
3.1 k-NN Classifier Results

- **Validation Accuracy Plot:**
  A plot was generated to visualize validation accuracy vs. different k values.
- **Test Set Performance:**
  The final model achieved high accuracy, with detailed metrics provided by the classification report:
  - **Accuracy, Precision, Recall, F1-score:** Computed on the test set.
- **Confusion Matrix:**
  The confusion matrix highlighted which digits were most frequently misclassified.
- **Misclassified Examples:**
  A subplot displaying 5 random misclassified examples was generated to visually inspect errors.



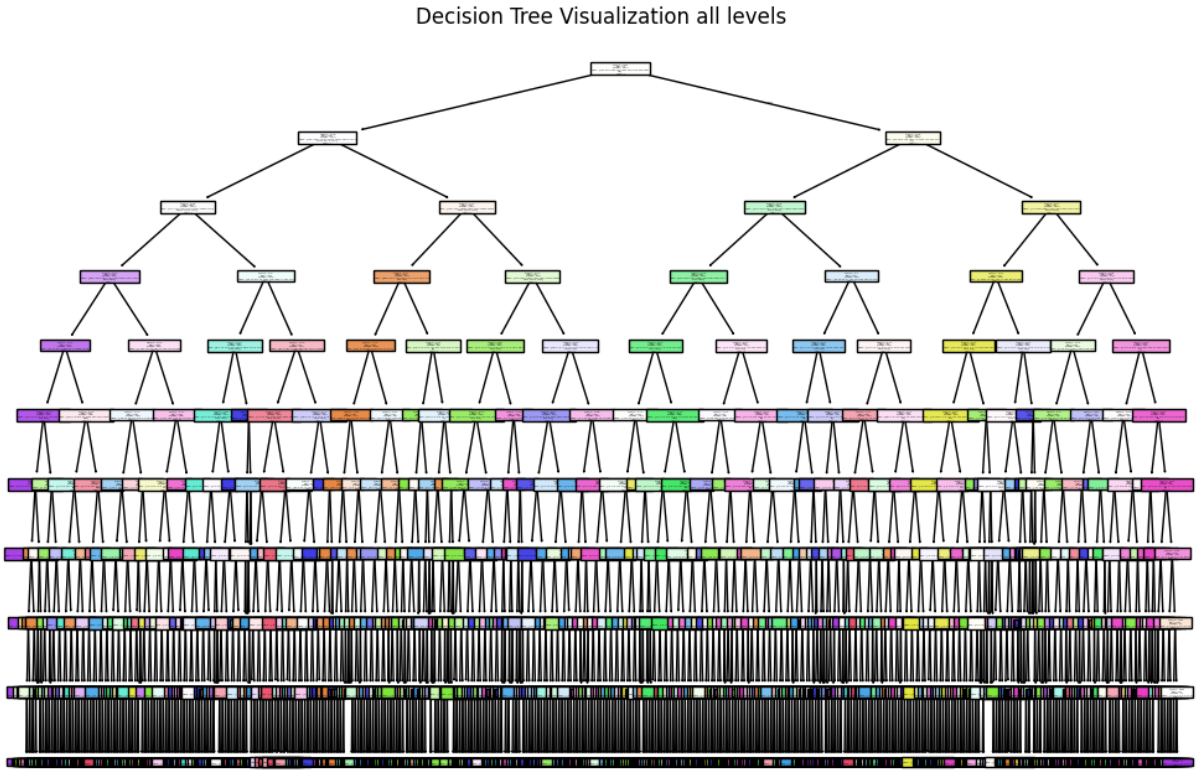
3.2 Decision Tree Classifier Results

- **Hyperparameter Tuning Summary:**
  The grid search results, including the best hyperparameters and corresponding cross-validation accuracy, were documented.
- **Test Set Performance:**
  The Decision Tree classifier's classification report provided accuracy, precision, recall, and F1-score.
- **Confusion Matrix:**
  Similar to k-NN, a confusion matrix was generated and analyzed.
- **ROC Curves:**
  A multi-class ROC curve plot for each digit was created. Each curve includes the AUC (Area Under the Curve) score, offering insight into the classifier’s performance across all classes.



- **Tree Visualization:**

  A simplified visualization (first two levels) of the Decision Tree was provided to illustrate the decision-making process of the model.



4. Analysis of Misclassifications

Observations:

- **Common Misclassifications:**
  Certain digits were more frequently confused with one another. For instance, the digits "4" and "9" or "5" and "3" might be misclassified due to similar handwritten styles.
- **Potential Explanations:**
  - **Variability in Handwriting:**
    Handwritten digits exhibit significant variation. Inconsistent stroke thickness or slight rotations can lead to ambiguity.
  - **Data Overlap:**
    Some misclassified examples might fall near the decision boundaries of the

models. When digits share similar shapes or features, even a well-tuned classifier might struggle to differentiate them.

- **Model Limitations:**
  k-NN relies on local proximity and may be affected by noisy or outlier samples. The Decision Tree, while interpretable, might oversimplify complex patterns if the tree is too shallow or overfit if too deep.

Visual Analysis:

- **Misclassified Image Plots:**
  A subplot displaying 5 random misclassified examples provided visual insight into the nature of these errors. The true labels versus the predicted labels allowed for a qualitative assessment of model performance.

5. Conclusion

This project demonstrated the practical application of two machine learning algorithms on the MNIST dataset. Through detailed data analysis, careful preprocessing, and rigorous hyperparameter tuning, both the k-NN and Decision Tree classifiers were effectively implemented. The comprehensive evaluation using accuracy, precision, recall, F1-score, confusion matrices, and ROC curves revealed strengths and limitations in each approach. Additionally, the analysis of misclassifications underscored the challenges posed by handwritten digit recognition and provided potential avenues for further refinement.