Notebook Link: <https://colab.research.google.com/drive/1rFPa8QgWvX14T9wMAs9KTQ1evAB0Qe7P?usp=sharing>

**Mete Kerem Berk**

**CS412 Machine Learning Homework 1**

**February 22, 2025**

1. **Overview**

In this assignment, k-Nearest Neighbors (k-NN) and Decision Tree classifiers are implemented on the MNIST dataset. The MNIST dataset contains 28×28 grayscale images of handwritten digits (0-9), where each pixel value ranges from 0 to 255.

1. **Dataset and Processing**
   1. **Data Loading**

First, the MNIST dataset is loaded using the Keras API. The dataset is loaded as training and test sets. The initial training set is split between training and validation set 80/20 respectively and the test set is left as is.



Figure 1 - Data Loading Code

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 2 - Data Loading Output

The initial training set is split between the training set and the validation set to help in hyperparameter training and prevent overfitting. The validation set acts as a checkpoint to detect problems before testing.

* 1. **Data Analysis**

After loading the dataset, data analysis is done to identify dataset characteristics and necessary preprocessing steps.

First, class distribution is controlled. The number of samples per digit is computed and displayed to check for imbalances.



Figure 3 - Class Distribution Code

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4 - Class Distribution

From Figure 4, it is seen that the samples are balanced in the dataset. This shows that preprocessing techniques like resampling or class weighting are not necessary.

Then, the mean and the standard deviation of the pixel values are computed. The mean helps understand the general brightness level of images and the standard deviation shows whether the images have high contrast or are mostly uniform in brightness.



Figure 5 - Mean & Standard Deviation Code



Figure 6 - Mean & Standard Deviation

The mean and standard deviation values are moderate on a 0-255 scale. These values will need to be normalized (0-1 scale) in the preprocessing steps.

Finally, random samples from each digit are displayed to confirm data quality and structure. It also helps detect noise, mislabeling and unexpected artifacts.



Figure 7 - Visualization Code

metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 8 - Random Samples

* 1. **Data Preprocessing**

In the data preprocessing step, the images are normalized so the pixel values are scaled to the range [0,1]. To do this, Z-score normalization is used. This process standardizes the dataset so that the pixel values have a mean of 0 and a standard deviation of 1.



Figure 9 - Standardization Code

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 10 - Standardization Results

Then, the dataset needs to be reshaped because k-NN classifiers works with 2D inputted data. Currently, the shape of the dataset is **(number of samples, height, width)** but this shape does not work for k-NN as it is 3D. The shape will be changed to **(number of samples, height \* width)** which is 2D.



Figure 11 - Reshaping Code



Figure 12 - Reshaping Results

1. **k-NN Classifier**
   1. **Model Initialization and Hyperparameter Tuning**

The k-NN classifier model is initialized and is experimented for the k values of [1, 3, 5, 7, 9]. During the experimentation, the validation set is used to determine the accuracy of the classifier.



Figure 13 - k-NN Initializer and k-Value Tester

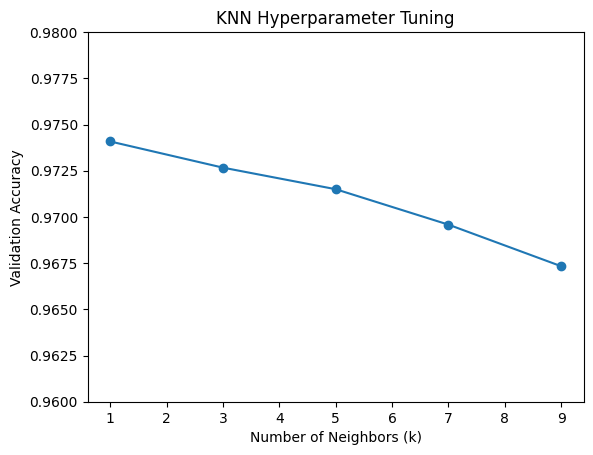


Figure 14 - Accuracy of Different k Values

The best value for k is found to be 1 with a validation accuracy of 97.41%. This can be due to the dataset being well separated or dataset not being noisy. However, a low k value will result in a higher sensitivity to noise and overfitting.

* 1. **Final Model Training and Evaluation**

For the final model, the training and validation sets are combined to retrain the k-NN classifier. Then, the final model is evaluated on the test set by reporting accuracy, precision, recall and F1-score.



Figure 15 - Final k-NN and Evaluation Code

metin, ekran görüntüsü, menü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 16 - Classification Report

The accuracy score from the test set is 97% which matches the validation accuracy of 97.41%. Each of the digits have remarkably high precision and recall values which means fewer false positives and false negatives, respectively.

Then, the confusion matrix is created to determine which digits are being confused with each other.



Figure 17 - Confusion Matrix Code

ekran görüntüsü, metin, kare, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 18 - Confusion Matrix

From the confusion matrix, the misclassifications can be determined.

* 8 is misclassified as 3 and 5. This can be because of similar round shapes.
* 2 and 7 are confused with each other. This is likely because of their similar shapes.
* 3 and 5 are confused with each other. This can be because of similar round shapes.
* 4 and 9 are confused with each other. This is likely because of their similar shapes.

metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 19 - Some Misclassifications

1. **Decision Tree Classifier**
   1. **Model Training and Hyperparameter Tuning**

Next, a Decision Tree classifier is trained on the MNIST dataset. The hyperparameters are tuned in the following range:

* Max. depth: [2, 5, 10]
* Min. samples split: [2, 5]



Figure 20 - Decision Tree Code

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 21 - Decision Tree Hyperparameter Results

The results show that with a ‘max\_depth’ of 10 and ‘min\_samples\_split’ of 2 results with the highest accuracy with 84.75%. The accuracy is proportional with the maximum depth of the tree. This makes sense as a deeper tree allows for a more detailed capture of the data. The minimum samples split is inversely proportional with accuracy. This is because a higher number of results in less overfitting.

* 1. **Evaluation**

The Decision Tree classifier is evaluated in the same way as the k-NN classifier. A classification report is created to determine accuracy, precision, recall and F1-score.



Figure 22 - Evaluation Code

metin, ekran görüntüsü, menü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 23 - Classification Report

The accuracy score from the test set is 87%. It performs best with digits 1 (precision: 0.95, recall: 0.96) and 0 (precision: 0.91, recall: 0.94). However, the model struggles with digits 5 (precision: 0.84, recall: 0.80) and 8 (precision: 0.80, recall: 0.81).

ekran görüntüsü, kare, renklilik, kalıp, desen, düzen içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 24 - Decision Tree Confusion Matrix

metin, çizgi, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 25 - Decision Tree ROC Curve

The AUC values range from 93% to 99% which show a remarkably high accuracy from the decision tree. The most confused digit is 8 with an AUC of 93% and the least confused digit is 1 with an AUC of 99%. With the confusion matrix it can be seen that 8 is mostly confused as 2, 3 and 9. The most false positives happen with digit 4 being confused as 9.

1. **Conclusion**

In summary, both the k Nearest Neighbors (k-NN) and Decision Tree models did well on the MNIST dataset with k-NN having an accuracy rate of 97% compared to the 87% of the Decision Tree. The k-NN classifier had high accuracy but can face overfitting issues with lower k values while the Decision Tree model encountered challenges with specific digits, especially those that had resemblances in their shapes.