

**The University of Azad Jammu and Kashmir, Muzaffarabad** Department of Software Engineering

**Machine Learning (HU-3105)**

**Lab Instructor: Engr. Muhammad Awais**

**Submitted by: Muhammad Haider Ali Raja**

**Submitted To: Engr. Muhammad Awais**

**Roll No: 2022-SE-32**

**Semester: 5th**

**Assigned On: March 03,2025**

**Submission Date: March 18,2025**

**~Open Ended lab~**

Contents

[**Introduction**: 3](#_Toc193176283)

[**Methodology** 3](#_Toc193176284)

[**Dataset Preparation:** 3](#_Toc193176285)

[**Models Used:** 3](#_Toc193176286)

[**Why These Models for MNIST?** 3](#_Toc193176287)

[1. **Logistic Regression** 3](#_Toc193176288)

[2. **K-Nearest Neighbors (KNN)** 3](#_Toc193176289)

[3. **Naïve Bayes** 3](#_Toc193176290)

[4. **Multi-Layer Perceptron (MLP - Neural Network)** 3](#_Toc193176291)

[**Model Evaluation Metrics** 4](#_Toc193176292)

[**Results** 4](#_Toc193176293)

[1. **Logistic Regression** 4](#_Toc193176294)

[2. **K-Nearest Neighbors (KNN)** 4](#_Toc193176295)

[3. **Naïve Bayes** 4](#_Toc193176296)

[4. **Multi-Layer Perceptron (MLP - Neural Network)** 4](#_Toc193176297)

[**Hyperparameter Tuning** 4](#_Toc193176298)

[**Conclusion** 4](#_Toc193176299)

**Introduction**: The MNIST dataset consists of handwritten digits (0-9) represented as 28x28 grayscale images, flattened into 784-dimensional vectors with pixel intensities ranging from 0 to 255. The dataset is pre-split into training (mnist\_train.csv, 60,000 samples) and testing (mnist\_test.csv, 10,000 samples) sets. This lab leverages these preprocessed CSV files to train and evaluate classification models, exploring their performance on digit recognition in an open-ended framework.

# **Methodology**

## **Dataset Preparation:**

* **Loading:** Loaded mnist\_train.csv and mnist\_test.csv into pandas DataFrames.
* **Feature and Label Separation:**
  + **Training:** X\_train (60,000 × 784 features), y\_train (labels).
  + **Testing:** X\_test (10,000 × 784 features), y\_test (labels).
* **Preprocessing:** Applied feature scaling using StandardScaler and performed feature selection using SelectKBest.

# **Models Used:**

* **Logistic Regression**
* **K-Nearest Neighbors (KNN)**
* **Naïve Bayes**
* **Multi-Layer Perceptron (MLP - Neural Network)**

# **Why These Models for MNIST?**

## **Logistic Regression**

* + Serves as a simple baseline model.
  + Works well on linearly separable data.
  + Provides a benchmark for comparison with non-linear models.
  + Implemented with max\_iter=1000 to ensure convergence.

## **K-Nearest Neighbors (KNN)**

* + Instance-based learning method that does not assume data distribution.
  + Handles non-linear decision boundaries better than Logistic Regression.
  + Effective in distinguishing visually similar digits.
  + Implemented with n\_neighbors=5 for optimal performance.

## **Naïve Bayes**

* + Probabilistic model based on Bayes’ theorem.
  + Works well with high-dimensional data.
  + Assumes feature independence, making it computationally efficient.
  + Implemented using GaussianNB().

## **Multi-Layer Perceptron (MLP - Neural Network)**

* + Captures complex pixel relationships.
  + Uses activation functions for non-linearity, improving classification.
  + Learns feature representations automatically.
  + Implemented with max\_iter=1000 to ensure sufficient training.
  + Default hidden layers were used, with one hidden layer unless specified otherwise.

# **Model Evaluation Metrics**

* **Accuracy:** Measures the percentage of correctly classified instances.
* **Precision:** The proportion of true positive predictions among all predicted positives.
* **Recall:** The proportion of true positive predictions among actual positives.
* **F1-Score:** The harmonic mean of precision and recall.
* **Confusion Matrix:** Identifies misclassification patterns.

# **Results**

## **Logistic Regression**

* + **Accuracy:** 91.2%
  + **Precision:** Ranges from 85% to 96%
  + **Recall:** Ranges from 84% to 97%
  + **F1-score:** Averages around 90%
  + **Confusion Matrix:** Shows minor misclassification errors, particularly between similar digits.

## **K-Nearest Neighbors (KNN)**

* + **Accuracy**: 96.9%
  + **Precision:** Ranges from 95% to 99%
  + **Recall:** Ranges from 94% to 100%
  + **F1-score:** Averages around 97%
  + **Confusion Matrix:** Shows improved classification with fewer misclassifications compared to Logistic Regression.
  + The most commonly misclassified digits were **3 and 8**, and **4 and 9**.

## **Naïve Bayes**

* + **Accuracy:** 85.7%
  + **Precision:** Varies significantly per class due to independence assumption.
  + **Recall:** Performs well for some digits but struggles with overlapping patterns.
  + **F1-score:** Averages around 86%
  + **Confusion Matrix**: Shows higher misclassification rates than other models.
  + The highest confusion occurred between **3 and 8**.

## **Multi-Layer Perceptron (MLP - Neural Network)**

* + **Accuracy:** 95.3%
  + **Precision:** Ranges from 92% to 98%
  + **Recall:** Ranges from 91% to 97%
  + **F1-score:** Averages around 95%
  + **Confusion Matrix:** Shows strong classification performance but slightly lower than KNN.
  + Default hidden layers were used, typically one layer unless modified.

# **Hyperparameter Tuning**

* No explicit hyperparameter tuning was performed in the experiment.
* If tuning were applied:
  + KNN could have been tested with different K values (e.g., n\_neighbors=3, 7, 9).
  + MLP could have been optimized with more hidden layers or different activation functions.
  + Logistic Regression could have experimented with different solvers (lbfgs, saga).

# **Conclusion**

* KNN achieved the highest accuracy (96.9%), followed by MLP (95.3%), Logistic Regression (91.2%), and Naïve Bayes (85.7%).
* The confusion matrices suggest that certain digits, such as **3 and 8 or 4 and 9**, were more prone to misclassification.
* The results indicate that KNN performed best for this dataset, making it a preferable choice for digit classification under the given conditions.
* Naïve Bayes performed the worst, likely due to its strong independence assumption, which does not hold well for MNIST data.
* Future improvements could include hyperparameter tuning, additional preprocessing steps like normalization, and testing more advanced deep learning models such as Convolutional Neural Networks (CNNs) to enhance classification performance further.