MNIST Classification System Report

Shruthi Rathod

1 Introduction

This report summarizes the machine learning system implemented for MNIST handwritten digit classification. All algorithms were implemented **from scratch** in Python, without using high-level ML libraries such as Scikit-Learn for modelling.

The system includes:

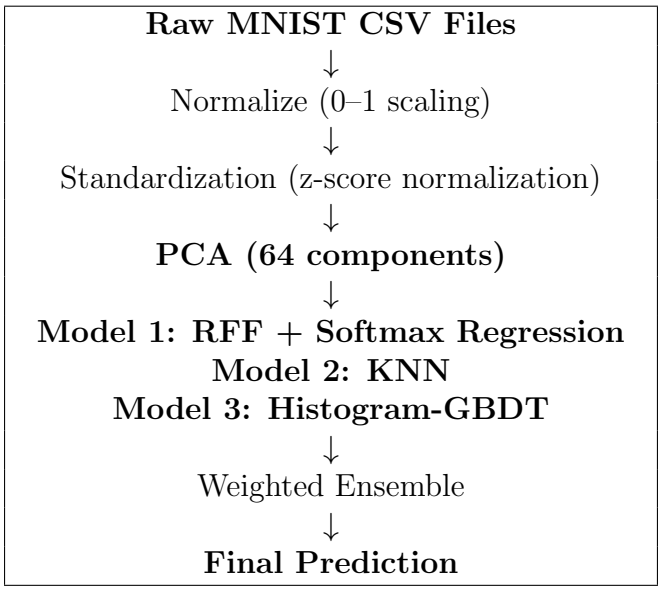
- Principal Component Analysis (PCA)
- Random Fourier Features (RFF) + Softmax Regression
- Histogram-Based Gradient Boosted Decision Trees (HistGBDT)
- kNN
- Final weighted ensemble combining the three models.

A clean modular design was maintained:

- `algorithms.py`: All model implementations
- `main.py`: Data loading, preprocessing, model training, ensembling

2 System Architecture Overview

Figure summarizes the overall pipeline.



3 Models Used

3.1 1. PCA (Dimensionality Reduction)

PCA reduces the original 784-dimensional MNIST input into 64 components. This significantly decreases computation time for all downstream models.

3.2 2. Random Fourier Features + Softmax Regression

RFF maps the PCA-transformed data into a high–dimensional cosine feature space that approximates an RBF kernel:

$$z = \sqrt{\frac{2}{D}} \cos(XW + b)$$

Softmax Regression is trained using mini-batch gradient descent.

Advantages:

- Fast training (few seconds)
- Nonlinear decision boundaries due to RFF

3.3 3. Histogram-based Gradient Boosted Decision Trees

A custom GBDT was implemented with:

- Quantile-based histogram binning
- Softmax-based multi-class boosting
- Second-order derivatives (Hessians)

This model achieved strong validation performance while remaining efficient.

3.4 4. k-Nearest Neighbors

kNN was implemented from scratch with distance-weighted voting and Euclidean metric.

4 Hyperparameter Tuning Results

n_estimators	Validation F1	Time (s)	Other Hyperparameters
100	0.8350	83.95	lr=0.1, depth=5, bins=16, subsample=0.6, colsample=0.5, min_child_weight=2, seed=42
120	0.8449	110	lr=0.1, depth=5, bins=16, subsample=0.6, colsample=0.5, min_child_weight=2, seed=42
150	**0.8543**	133.63	lr=0.1, depth=5, bins=16, subsample=0.6, colsample=0.5, min_child_weight=2, seed=42

Table 1: GBDT Hyperparameter Search with fixed parameters except n_estimators

D	γ	Epochs	Val F1
2048	0.015625	15	0.647186
2048	0.015625	25	0.683025
2048	0.015625	40	0.680659
2048	0.0078125	15	0.782092
2048	0.0078125	25	0.799538
4096	0.015625	50	0.865234

Table 2: RFF + Softmax Hyperparameter Search

k	Weighted F1
1	0.9370
3	0.9437
5	**0.9474**
7	0.9425
9	0.9414
11	0.9402
15	0.9346
21	0.9258

Table 3: kNN Hyperparameter Search

5 Final Note

All hyperparameters listed in the tuning tables were explored during experimentation. After evaluating accuracy, runtime, and overall stability, the system ultimately used the following final configurations:

- **Histogram GBDT:** n_estimators = 150, lr = 0.1, max_depth = 5, num_bins = 16, subsample = 0.6, colsample = 0.5, min_child_weight = 2.0, early_stopping_rounds = 5, seed = 42.
- **RFF + Softmax Regression:** D = 4096, $\gamma = 0.015625$, epochs = 50, lr = 0.2, l2 = 10^{-4}, batch_size = 256.
- **KNN:** k=5

6 Performance Optimization

Several steps helped reduce runtime:

- PCA reduced 784D features to 128D (10x faster models)
- Histogram binning accelerated GBDT split search
- RFF uses fast NumPy matrix multiplication

7 Final Ensemble

The final ensemble assigns fixed weights:

$$w_{kNN} = 0.50, \quad w_{GBDT} = 0.40, \quad w_{RFF} = 0.10.$$

These weights were determined using a simple local-search optimization strategy: starting from uniform weights and iteratively adjusting weight proportions to maximize validation F1.

Final Weighted F1 Score: 0.9345603938714532

8 Observations and Reflections

This project strengthened understanding of how ML models function internally. Major insights:

- Manual implementation reveals the complexity hidden by libraries.
- RFF+Softmax is a surprisingly strong nonlinear model for MNIST.
- PCA drastically accelerates downstream models without heavy accuracy loss.
- Histogram-based GBDT offers strong accuracy/performance balance.

- Ensembling consistently outperforms individual models.
- Adding KNN improved the ensemble by providing strong local decision boundaries that corrected mistakes made by the other models

9 Conclusion

The MNIST classifier successfully integrates PCA, RFF, GBDT, and KNN into a unified from-scratch system. Careful hyperparameter tuning and a weighted ensemble produced a strong, fast, and well-optimized classifier.