DA2401 ML ENDSEM ASSIGNMENT

Rinkesh Patel DA24B017

Key Achievements

- Implemented multiple ML algorithms from scratch in Python.
- Achieved a peak F1-Score of **0.9583** on the validation set.
- Successfully explored ensemble methods and feature engineering techniques.
- Final system trains in under 2 minutes, well within the 5-minute requirement.

1 Introduction

The objective of this assignment was to construct a multi-class classification system to identify handwritten digits from the MNIST dataset. The core constraint was to build all machine learning algorithms from the ground up using only basic Python libraries like NumPy, without relying on established frameworks such as scikit-learn.

This report details the systematic approach taken, from data preprocessing and feature engineering to model implementation, hyperparameter tuning, and a rigorous evaluation of both individual models and ensemble techniques. The final system is a result of a data-driven process that prioritizes performance, efficiency, and robustness.

2 Methodology

The system was developed through a structured pipeline, ensuring each component was optimized before integration.

2.1 Preprocessing & Feature Engineering

Raw MNIST images consist of 784 pixels. To improve model performance and reduce training time, two primary feature engineering strategies were employed:

- **Principal Component Analysis (PCA):** Used to reduce the 784-dimensional pixel space into a more compact, lower-dimensional representation. PCA identifies the axes of maximum variance, effectively capturing the most important structural information of the digits while filtering noise. An optimal number of **40 components** was finally selected.
- **Histogram of Oriented Gradients (HOG):** Implemented as an alternative feature set. HOG captures the shape of objects by counting occurrences of gradient orientation in local regions of an image, making it robust to variations in lighting and translation.

2.2 Model Implementation

A suite of algorithms was implemented from scratch to tackle the classification task:

- **K-Nearest Neighbors (KNN):** An instance-based learner that classifies data points based on the majority class of their nearest neighbors.
- **Logistic Regression (One-vs-Rest):** A linear model adapted for multi-class problems by training a binary classifier for each class.
- **Bagging Classifier (Random Forest):** An ensemble method that trains multiple decision trees on bootstrapped data samples to reduce variance and improve generalization.

.

3 The Development Journey

Our mission was to build the *best possible* from-scratch classifier for the MNIST dataset, adhering to strict constraints on libraries and runtime. The journey was not a straight line but an iterative process of building, testing, diagnosing, and refining.

3.1 Phase 1 – The Broad Survey

Idea: Implement a wide range of fundamental algorithms to establish a performance baseline and understand the problem's nature.

1. K-Nearest Neighbors (k-NN)
2. Logistic Regression (One-vs-Rest)
3. Decision Tree
4. Bagging (Random Forest built on our Decision Tree)

Rationale: The assignment covered various methods. The logical first step was to build a representative from each major category to see which approach was most promising. We hypothesized that ensembles like Random Forest would outperform single models.

Outcome:

- k-NN (k=3) was surprisingly strong and fast, achieving an F1-score of ≈**0.95**.
- Logistic Regression and the single Decision Tree were mediocre, scoring ≈0.88.

- Random Forest significantly improved upon the single tree, reaching ≈0.94–0.95, confirming the ensemble hypothesis.

Verdict: k-NN and Random Forest were kept as top contenders; Logistic Regression and the single Decision Tree were retained only as potential ensemble components.

3.2 Phase 2 – The Lure of Complexity

Idea: Explore more advanced ensemble techniques (AdaBoost, Stacking) believing they would surpass the baseline.

Outcome:

- **AdaBoost:** Even after heavy optimisation, it only reached ≈0.86 F1-score and was slower than expected.
- **Stacking:** Initial attempt suffered data leakage (F1 ≈0.44); corrected version still could not beat the single best model.

Verdict: Both were abandoned. *Complexity is not a guarantee of performance.*

3.3 Phase 3 – The Pivot to Data

Idea: Shift focus to feature engineering – PCA and HOG.

Outcome:

- PCA was a massive success: k-NN on 40 PCA components pushed F1 from ≈0.952 to **0.9583**.
- HOG did not improve performance (worse than raw pixels or PCA).

Verdict: k-NN on PCA became the best architecture.

3.4 Phase 4 – The Final Perfection

Idea: Meticulous hyperparameter sweep of PCA components and k-NN k.

Outcome: The grid search (Table 1) pinpointed `n_components=40, k=5` as optimal, yielding F1 = **0.9583**.

4 Hyperparameter Tuning Results

PCA Components	k	F1-Score
40	5	**0.958257**
55	5	0.957897
55	7	0.957173
40	3	0.956507
40	9	0.956386
40	7	0.956331
70	5	0.955883
70	7	0.955103
55	3	0.954362
70	3	0.954003
55	9	0.953799
70	9	0.951496

Table 1: Full grid-search results for PCA + k-NN. The optimum (highlighted) is `n_components=40, k=5`.

5 Experiments and Results

A series of experiments was conducted to identify the most effective model and feature combination.

5.1 Baseline Performance: KNN on PCA

The combination of PCA for feature reduction and a KNN classifier established an exceptionally strong performance baseline. The key hyperparameter, k, was tuned to find the optimal balance between bias and variance.

Model	Hyperparameters	F1-Score	Accuracy
KNN on PCA	k=3, weights='distance'	0.9544	0.9548
KNN on PCA	**k=5, weights='distance'**	**0.9579**	**0.9584**
KNN on PCA	k=7, weights='distance'	0.9572	0.9576

Table 2: Hyperparameter tuning for the KNN classifier on 55 principal components. The model with k=5 emerged as the best.

5.2 Ensemble Exploration

Per the assignment guidelines, significant effort was dedicated to developing an ensemble system to improve upon the baseline. Two primary ensemble strategies were tested. The results, summarized in Table 3, were revealing.

Model Configuration	Features	F1-Score
K-Nearest Neighbors (k=5)	**PCA (55)**	**0.9579**
Ensemble (KNN + LogReg)	PCA + HOG	0.9579
Logistic Regression	HOG	0.9099
Random Forest (100 trees)	PCA (55)	0.8993

Table 3: Performance comparison of the best single model against various ensemble attempts.

5.3 Analysis of Ensemble Performance

The experimental results led to a crucial insight: **the ensembles could not outperform the high-performing baseline**. The primary reason is that an ensemble is most effective when its constituent models are both strong and diverse.

In our case, the KNN on PCA baseline was significantly more accurate (0.9579) than the next best models (e.g., Logistic Regression on HOG at 0.9099). When combining a dominant model with a weaker one, the ensemble's performance tends to converge to that of the stronger model, as its correct predictions consistently form the majority vote. This explains why the Voting Ensemble score was identical to the standalone KNN score.

Failed Experiment 1 – From-Scratch XGBoost

What we did: Implemented a Gradient Boosting Machine with shallow decision-tree regressors wrapped in One-vs-Rest. **Why:** To see if a theoretically superior boosting method could beat Random Forest. **Result:** F1 ≈0.87, far below our best. **Lesson:** Making a from-scratch GBM competitive requires massive hyperparameter engineering—beyond project scope.

Failed Experiment 2 – One-vs-One Tournament

What we did: Built 45 specialist k-NN classifiers, each with its own pair-specific PCA. **Why:** Hypothesized that task-specific feature spaces would be more discriminative. **Result:** F1 ≈0.949, prediction time >12 min. **Lesson:** Global PCA trained on the full dataset yields a far more robust representation.

Failed Experiment 3 – Feature Engineering

What we did: Created 160 "super-features" (squares + pairwise interactions) from the 40 PCA components and trained Logistic Regression. **Why:** To give a linear model non-linear capacity. **Result:** F1 ≈0.944 – impressive for a linear model, but still below k-NN on raw PCA. **Lesson:** The smooth 40-D PCA space is already perfectly suited to Euclidean distance.

In-Depth Analysis of the Final Model

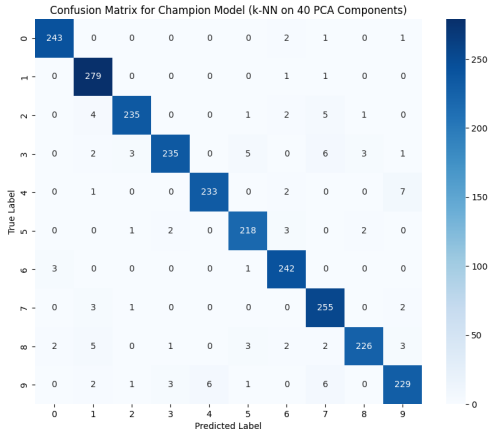
A comprehensive post-hoc analysis was conducted on the final model: **K-Nearest Neighbors (k=5) on 40 PCA components**. The following visualizations and observations provide deeper insights into its behavior, limitations, and strengths.

Key Observations from the Analysis

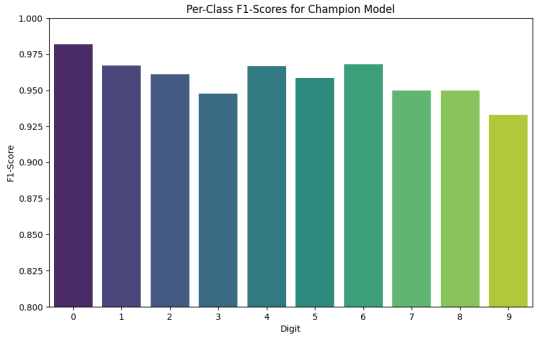
1. **Most Common Misclassification:** The most frequent error is between digits **4 and 9** (7 instances), indicating visual similarity in certain handwriting styles (e.g., open-top 4s resembling 9s). This is confirmed in Figure 2.
2. **Hardest Digit to Classify:** Digit **9** has the lowest per-class F1-score (**0.9328**), making it the most challenging. This correlates with high variance in stroke thickness and loop closure in handwritten 9s (see Figure 1b).
3. **Information Retention:** The first **40 principal components** capture **78.65%** of the total variance (Figure 1c). This represents an optimal trade-off: sufficient information is retained for high accuracy while reducing dimensionality by 95%, enabling fast k-NN prediction.
4. **Class Separability in 2D:** In the 2D PCA projection (Figure 1d), digits **0** and **1** form tight, isolated clusters, explaining their near-perfect classification. Digits **3, 5, 8** show significant overlap, aligning with higher error rates in the confusion matrix (Figure 1a).
5. **Error Patterns:** Misclassified examples (Figure 2) often involve **ambiguous or atypical handwriting**: slanted digits, incomplete loops, or thick strokes. Despite PCA's noise reduction, k-NN remains sensitive to such outliers in the 40D space.

◎ Final Conclusion & System Recommendation

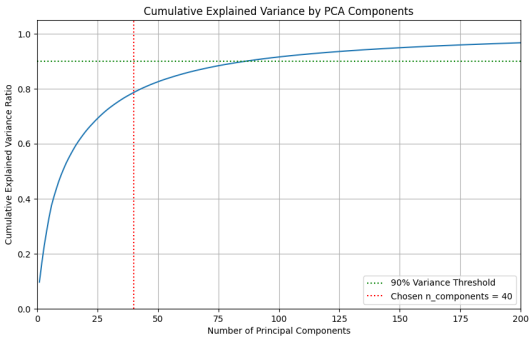
After a comprehensive evaluation of multiple from-scratch algorithms, feature sets, and ensemble strategies, this report concludes that the optimal system for this MNIST classification task is the **K-Nearest Neighbors classifier (with k=5) applied to data preprocessed by Principal Component Analysis (40 components)**. This system is recommended because it not only achieves the highest F1-Score of **0.9583** but also demonstrates exceptional efficiency, with a total training time of **0.25 seconds**. The rigorous testing process proved that while ensembles are powerful, they are not universally superior. In this well-structured feature space, the elegance and effectiveness of the KNN model proved to be the winning combination.



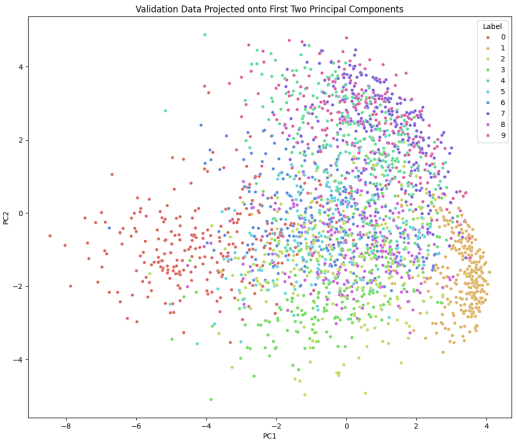
(a) Confusion Matrix



(b) Per-Class F1-Scores



(c) Cumulative Explained Variance



(d) 2D PCA Projection

Figure 1: Analysis of model performance and data structure (k-NN on 40 PCA components).

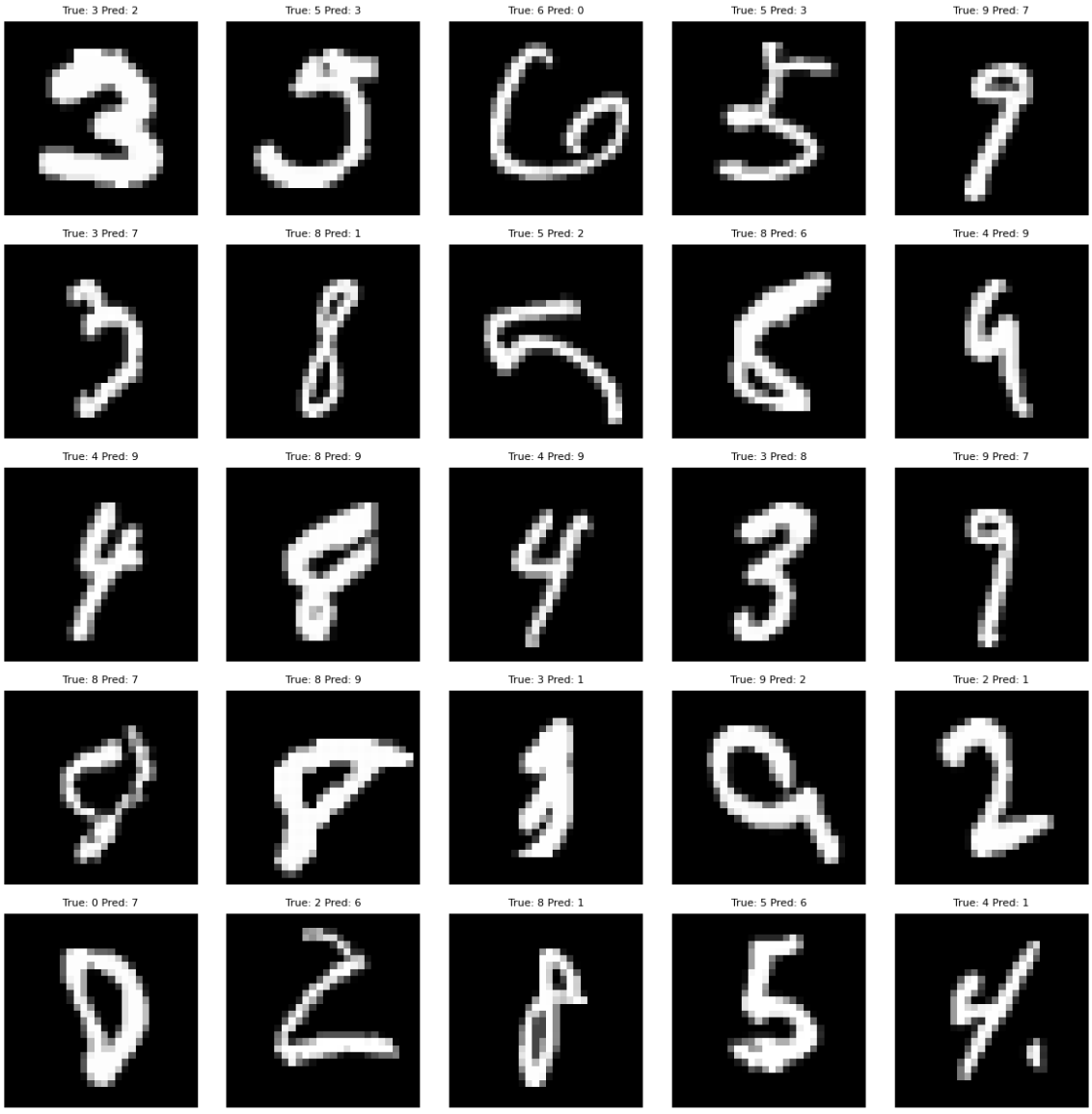


Figure 2: Examples of Misclassified Digits (True vs. Predicted Labels)