CST 3170 MACHINE LEARNING COURSEWORK REPORT

Digit Recognition System

Author: Dumitru Nirca

Date: December 2025

Course: CST 3170 Machine Learning

SECTION 1. EXECUTIVE SUMMARY

Digit recognition over the UCI 8×8 grayscale dataset is evaluated with two-fold cross-validation using the provided dataSet1.csv / dataSet2.csv split. Features are z-score normalized, optionally augmented with spatial row/column means, 512 Random Fourier Features (RBF approximation), and degree-2 polynomial interactions (capped to ~800 terms). This expands the linear SVM feature space from 64 raw pixels to well over 1.3k dimensions, while keeping the code inside a single Java file as required.

Latest two-fold results (January 2026 run):

> - Nearest Neighbor (k = 1): **98.26 %**
> - Best k-NN per fold: **98.27 %**
> - Weighted k-NN: **98.24 %**
> - Linear SVM (Pegasos): **97.30 %**

Runtime profile: full evaluation ~7 hours, driven mainly by the linear SVM stage (45 one-vs-one models × 6 validation repeats per setting × 5-model ensemble, up to 240 epochs each).

Takeaway: exact Euclidean k-NN saturates above 98 %, while the Pegasos linear SVM demonstrates the advanced algorithm requirement with aggressive feature engineering and hyperparameter exploration.

SECTION 2. INTRODUCTION

Problem Statement

Build a machine learning pipeline that recognizes digits (0-9) from 8×8 pixel grids. The solution must achieve strong accuracy and compare multiple algorithms.

Dataset

- **Source:** UCI Machine Learning Repository
- **Format:** CSV files, 64 numeric features + 1 label
- **Classes:** Ten digits (0 through 9)
- **Features:** 8×8 grayscale pixel intensities

System Architecture

- **Models:** data structures for samples, predictions and results
- **Algorithms:** k-NN variants and linear SVM implementation
- **Utilities:** distance computations, evaluation metrics, loaders

SECTION 3. ALGORITHMS IMPLEMENTED

Nearest Neighbor Baseline

- Simple Euclidean lookup implemented via KNearestNeighbors with k = 1.

k-Nearest Neighbors

- Majority vote across k in {1, 3, 5, 7, 9, 11}.
- Each fold evaluates every k and reports the best performer.

Weighted k-Nearest Neighbors

- Same neighbor pool as standard k-NN.
- Votes weighted by 1 / (distance + epsilon) to dampen noisy points.

Linear Support Vector Machine

- One-vs-one linear SVM (45 binary classifiers) trained with Pegasos-style SGD.
- Automatic sweep over regularization (lambda), epoch count, learning rate floor, and shuffle strategy.
- Feature engineering: spatial augmentation (row/column averages), 512 Random Fourier Features (RBF approximation), and polynomial features (degree 2) for interaction capture.
- Ensemble of 5 models with voting for final predictions.
- Demonstrates feature scaling, regularization, and convergence monitoring.

SECTION 4. DISTANCE METRICS

- Euclidean distance is the required baseline and the metric used for submitted scores.

- Manhattan and Minkowski distances (order p) are available inside DistanceCalculator to support additional experiments without refactoring the pipeline.

SECTION 5. EVALUATION METHODOLOGY

1. **Two-fold cross-validation:** Fold 1 trains on dataSet1.csv and tests on dataSet2.csv; Fold 2 swaps them.
2. **Metrics:** overall accuracy, 10×10 confusion matrices, per-class precision/recall/F1, macro averages.
3. **Runtime hyperparameter selection:**
    - k-NN evaluates every candidate value each fold.
    - Linear SVM shuffles, reserves an 85 % / 15 % validation split, and averages six repeats per hyperparameter set.

SECTION 6. RESULTS AND ANALYSIS

Two-Fold Summary (Latest Run)

Algorithm	Fold 1 Accuracy	Fold 2 Accuracy	Average
Nearest Neighbor (k = 1)	98.04 %	98.47 %	**98.26 %**
Best k-NN per fold	98.08 % (k = 3)	98.47 % (k = 1)	**98.27 %**
Weighted k-NN	98.01 %	98.47 %	**98.24 %**
Linear SVM (Pegasos)	97.30 %	97.30 %	**97.30 %**

Technical observations

- **k tuning matters:** Fold 1 prefers k = 3 for noise smoothing, while Fold 2 falls back to k = 1; re-tuning per fold avoids locking into one radius.
- **Distance weighting is slightly worse than unweighted k-NN on this split,** suggesting the closest mislabeled/ambiguous pixels can dominate when heavily weighted.
- **Linear SVM stack:** per-feature z-score normalization, spatial means, 512 RFF dimensions (RBF approximation), and up to ~800 polynomial interaction terms expand the linear decision surface.

- **Best SVM configurations** cluster around lambda in [3e-4, 1.2e-3], 100-240 epochs, and minimum learning rates 5e-8 to 1.5e-7; early stopping kicks in when few hinge-loss updates remain.
- **Ensemble voting** across 5 independently trained SVMs reduces variance from random Fourier sampling and stochastic shuffling.
- **Runtime is dominated by the SVM stage:** 45 one-vs-one classifiers × 6 validation repeats × 5-model ensemble × up to 240 epochs.

Error Analysis

- Most confusions occur between visually similar digits (1 vs 8, 3 vs 5, 4 vs 9) where stroke thickness and loops overlap.
- Class weighting inside each one-vs-one SVM pair counteracts local class imbalance, improving margins on thinner-stroke digits; k-NN still leads on curved digits where local shape cues dominate.

SECTION 7. CODE QUALITY AND IMPLEMENTATION

- All logic is contained in DigitRecognitionApp.java, partitioned into models, algorithms and utilities with clear comments.
- The Classifier interface enforces consistent APIs; evaluation helpers centralize reporting.
- Constants hold dataset paths, k grids and hyperparameter ranges, eliminating "magic numbers".
- Defensive programming includes null checks, normalization guards and informative logging.
- Style follows JavaDoc conventions and remains readable despite the single-file constraint.

SECTION 8. CONCLUSIONS

Key Achievements

1. Baseline requirement exceeded with 98.27 % average accuracy.
2. Advanced algorithm implemented: Pegasos linear SVM achieving 97.30 % accuracy through comprehensive hyperparameter search and ensemble training (~7 hour computation).

3. Automated two-fold evaluation with confusion matrices and per-class summaries.
4. High code quality maintained under single-file limitation.
5. Documentation captures methodology and future work.

Future Enhancements

- Explore feature engineering (PCA, intensity statistics) before training the SVM.
- Experiment with kernelized SVMs or shallow neural networks for higher "quality of algorithm" credit.
- Persist evaluation summaries to CSV/text automatically to make reporting repeatable.