PROJECT REPORT

Rathod Mithilesh da24b048

November 2025

1 Introduction

In this project , we will build a multi class classification system to predict the class label (0 to 9). Complete the assignment using plain Python with sim- ple packages like NumPy and SciPy, without using any built in ML packages like sklearn or xgb. Share Python file (.py format), that can be executed and tested.

my report:

I made this project useing the bellow models and the main model is KNN, the performance of several Models include Softmax Regression, Logistic Regression (OvR), Random Forest, KNN, XGBoost-style boosting, and multiple stacked ensembles. . first normalizes all images and then applies PCA to reduce the feature size from 784 to 60. After this, different models are trained, including Softmax Regression, Logistic Regression (OvR), KNN, Random Forest, and a NumPy-based XGBoost model. Stacking ensembles were also created by combining the predictions of multiple base models.

All results are listed in table format for clarity.

2 Individual Model Results

2.1 KNN (k = 5)

Metric	Train	Validation
Accuracy	0.9624	0.9504
Macro Precision	0.1967	0.1957
Macro Recall	0.1967	0.1956
Macro F1	0.1967	0.1957
Micro Precision	0.9834	0.9784
Micro Recall	0.9834	0.9784
Micro F1	0.9834	0.9784

Table 1: KNN (k=5) Performance

2.2 Softmax Regression

Metric	Value
Validation Accuracy	0.8219
Macro Precision	0.1650
Macro Recall	0.1642
Macro F1	0.1642
Micro Precision	0.8219
Micro Recall	0.8219
Micro F1	0.8219

Table 2: Softmax Regression Performance

2.3 Logistic Regression (OvR)

Metric	Value
Validation Accuracy	0.8583
Macro Precision	0.1717
Macro Recall	0.1716
Macro F1	0.1716
Micro Precision	0.8583
Micro Recall	0.8583
Micro F1	0.8583

Table 3: Logistic Regression (OvR) Performance

2.4 Random Forest

Metric	Value
Validation Accuracy	0.9352
Macro Precision	0.1871
Macro Recall	0.1870
Macro F1	0.1870
Micro Precision	0.9352
Micro Recall	0.9352
Micro F1	0.9352

Table 4: Random Forest Performance

2.5 XGBoost-style Boosted Trees

Metric	Value
Validation Accuracy	0.8535
Macro Precision	0.1707
Macro Recall	0.1707
Macro F1	0.1707
Micro Precision	0.8535
Micro Recall	0.8535
Micro F1	0.8535

Table 5: XGBoost Multi-class Performance

3 Hyperparameters of All Models and Ensembles

Each model was tuned by adjusting learning rates, epochs, number of trees, and boosting rounds. KNN performed best with $k = 5$, giving a validation accuracy of 97.84%. Softmax and Logistic Regression performed moderately well, while Random Forest and XGBoost improved accuracy but took more time. Stacking models such as *KNN + Softmax → Logistic* also reached 97.84% accuracy and gave more stable predictions

Table 6: Hyperparameters of All Models and Ensembles

Model	**Hyperparameters**
PCA (Dimensionality Reduction)	Components = 60; Max samples for SVD = 20,000; Mean-centering applied
Softmax Regression	Learning rate = 0.1; Epochs = 25; Classes = 10; Optimization = Gradient Descent
Logistic Regression (OvR)	Learning rate = 0.05; Epochs = 8; Batch size = 128; Classes = 10; One-vs-Rest training
KNN Classifier	k = 5; Distance = Euclidean; Brute-force search; Predicts using majority voting
Random Forest (Main)	Trees = 20; Subsample = 0.7; Feature subsample = 0.3; Max depth = 8; Min samples split = 10; Thresholds per feature = 10; Classes = 10
Random Forest (Stacks)	Trees = 15; Subsample = 0.7; Feature subsample = 0.5; Max depth = 7; Min samples split = 10; Classes = 10
XGBoost Binary (per class)	Estimators = 120; Learning rate = 0.15; Regularization $\lambda = 1.0$; Base learner = decision stump; Gradient = $g = p - y$; Hessian = $h = p(1-p)$
XGBoost Multi-Class	10 binary classifiers (one per class); Estimators = 120 each; LR = 0.15; Variant used in Stack-2: Estimators = 90
Stack 1: KNN + Softmax → Logistic	*Base models:* KNN(k=5), Softmax(lr=0.1, epochs=25). *Meta model:* Logistic Regression (lr=0.05, epochs=8, bs=128).
Stack 2: KNN + Logistic → Logistic	*Base models:* KNN(k=5), Logistic(lr=0.05, epochs=8). *Meta model:* Logistic(lr=0.05, epochs=8).
Stack 3: Logistic + XGB + RF → KNN	*Base models:* Logistic(lr=0.05, epochs=8), XGBoost-Multi(estimators=120, lr=0.15), RandomForest(trees=15, depth=7). *Meta model:* KNN(k=5).
Stack 4: KNN + RF + Logistic → Softmax	*Base models:* KNN(k=5), RandomForest(trees=15, depth=7), OvR Logistic Regression. *Meta model:* Softmax Regression(lr=0.1, epochs=25).

4 Final Comparison Table

Table 7: Final Comparison of All Models on MNIST Validation Set

Model	**Accuracy**	**Macro F1**
KNN (k = 5)	0.9504	0.1957
Softmax Regression	0.8219	0.1642
Logistic Regression (OvR)	0.8583	0.1716
Random Forest	0.9352	0.1870
XGBoost Multi-Class	0.8535	0.1707

5 Optimization Steps and Evaluation Results

To reduce training time, PCA was used to shrink the feature size. Mini-batches were used for gradient-based models, and Random Forest used subsampling. XGBoost used simple one-level stumps to keep boosting fast.and Other models had lower accuracy but much faster inference.

Stacking improved stability but increased prediction time.

6 Conclusion

Among all models, KNN (k=5) produced the strongest single-model performance with 95.04% validation accuracy. Ensemble stacking (KNN + Softmax $\rightarrow$ Logistic) matched this accuracy while improving stability. Traditional models such as Softmax, Logistic Regression, and XGBoost showed moderate performance, while Random Forest provided strong nonlinear learning but required significantly higher training time. This exercise showed that simple models can perform very well when combined with good preprocessing. KNN gave the highest accuracy after PCA. Linear models like Softmax and Logistic Regression were fast but less accurate. Random Forest and XGBoost improved performance but required more computation. Stacking models helped improve stability and combined the strengths of multiple algorithms.

Overall, this project helped in understanding how these models work internally and how processing steps