Name:-V.VedaVachan Reddy
Roll Number:- DA24B029

DA2401 ENDSEM REPORT

(A)

Summary of Models Used and System Architecture

This project implements four machine learning models from scratch for MNIST digit classification: Random Forest (bagging), a XGBoost-like gradient boosting model, Softmax Regression, and K-Nearest Neighbours (KNN). The Random Forest uses multiple decision stumps trained on bootstrap samples, while the boosting model sequentially fits regression stumps on residuals. Softmax Regression provides a linear baseline using gradient descent, and KNN performs distance based classification. The system includes separate modules for data loading, metric computation, model training, and result comparison.

(B)

Hyper Parameter Tuning Summary

- **Random Forest**
  - Number of trees: 60
  - Max per tree: 200
  - Split Criterion: Classification stump
- **XGBoost-like Model**
  - Boosting Rounds: 50
  - Learning Rate: 0.15
  - Max Random : 200
  - Threshold : 8
- **Softmax Regression**
  - Learning Rate: 0.1
  - Number of Epochs: 120
  - Optimization: Gradient Descent
- **KNN Classifier**
  - Number of Neighbors (k): 7
  - Metric: Euclidean
  - Training Method: Lazy learning (no training)

Comparison of Model Performance Across Accuracy, Precision, Recall, F1-Score, and Training Time

Random Forest Results

- **Accuracy:** 0.7431
- **Precision:** 0.1503
- **Recall:** 0.1483
- **F1 Score:** 0.1493

XGBoost-Like Results

- **Accuracy:** 0.8099
- **Precision:** 0.1626
- **Recall:** 0.1618
- **F1 Score:** 0.1622

Softmax Regression Results

- **Accuracy:** 0.8583
- **Precision:** 0.1717
- **Recall:** 0.1716
- **F1 Score:** 0.1717

KNN Results

- **Accuracy:** 0.9720
- **Precision:** 0.1945
- **Recall:** 0.1943
- **F1 Score:** 0.1944

(C)

System Optimization and Evaluation Summary

To keep the total run-time under 5 minutes, several optimization steps were applied:
(1) all models were implemented using NumPy without external ML libraries,
(2) only a limited random subset of features was used for Random Forest and XGBoost-like models to reduce computation,
(3) Decision Stumps were chosen as base learners to keep tree training extremely fast,

(4) KNN was optimized by vectorized distance computation,
(5) learning rates, number of boosting rounds, and number of trees were carefully tuned to balance accuracy and speed.

Overall, the system achieved good accuracy while maintaining a low execution time.

(D)

Observations and Learnings

This exercise provided how different classical ML models behave on a large dataset like MNIST.
I observed that simple models (KNN, Softmax Regression) are easy to implement but struggle to reach very high accuracy compared to ensemble methods.
Random Forest and XGBoost-like models showed noticeable performance gains, proving that ensemble learning reduces variance and improves stability.
I also realized that computational cost grows quickly with model complexity, so tuning parameters while respecting time limits is crucial.
Overall, this assignment understand that bias–variance trade-offs, optimization, and the importance of combining models for better performance.

The KNN classifier achieved the highest overall performance with an accuracy of 0.9720 and the best macro–F1 score. Its effectiveness shows that a simple distance-based model can perform extremely well on MNIST while remaining computationally efficient. Therefore, KNN is clearly the best-performing classifier among all models evaluated in this system.