University Of Azad Jammu And Kashmir

Muzaffarabad

Open Ended Report

|  |  |
| --- | --- |
|  | Muhammad Hussain |
| Roll No | 2022-SE-41 |
| Course Code | SE-3105 |
| Semester | 5th |
| Session | 2022-26 |
| Submitted To | Engr. Awais Rathor |

Open Ended Report of Mnist Dataset

# 1. Introduction

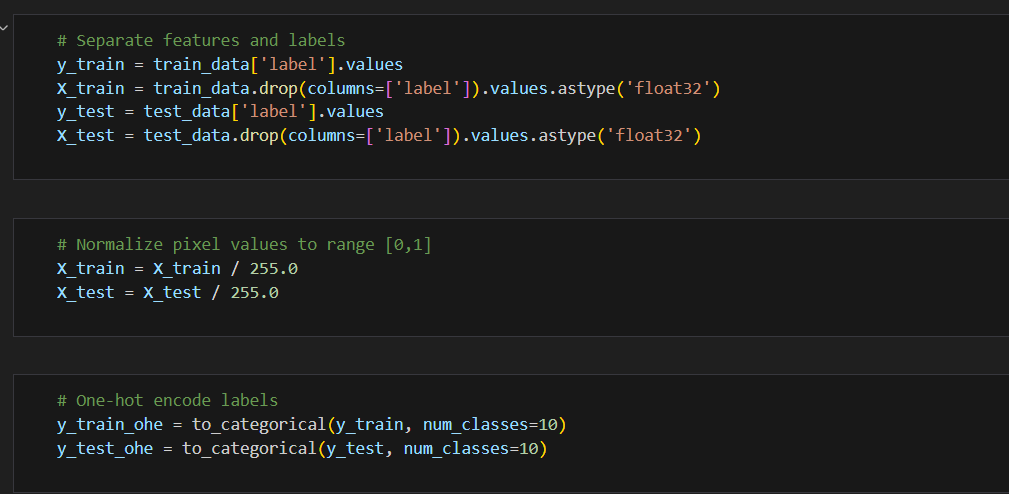
The dataset used for this classification task consists of pre-split training and test sets, commonly used in image classification problems. The training dataset comprises **60,000 rows and 785 columns**, while the test dataset contains **10,000 rows and 785 columns**. Each instance in the dataset represents a high-dimensional input with **784 features** (likely pixel values of a **28×28 grayscale image**) and one label column indicating the class. The primary objective of this study is to evaluate the performance of multiple classification models and determine the most effective one based on various performance metrics.

# 2. Methodology

## 2.1 Dataset Preparation

Since the dataset was already divided into training and test sets, no further splitting was required. However, preprocessing steps were applied to ensure optimal model performance:

## Preprocessing Steps:



* **Feature Scaling:**
  + Pixel values were **normalized to the range [0,1]** by dividing each value by **255**.
  + This ensures that all input features are within the same scale, preventing models (especially deep learning models) from being biased toward larger values.
* **Label Encoding:**
  + For models that require categorical outputs (such as ANN), the class labels were **one-hot encoded** to transform categorical labels into a format that the neural network can process.
  + Other models, such as KNN and Random Forest used integer-encoded labels.

## 2.2 Models Used & Hyperparameter Tuning

## Artificial Neural Network (ANN)

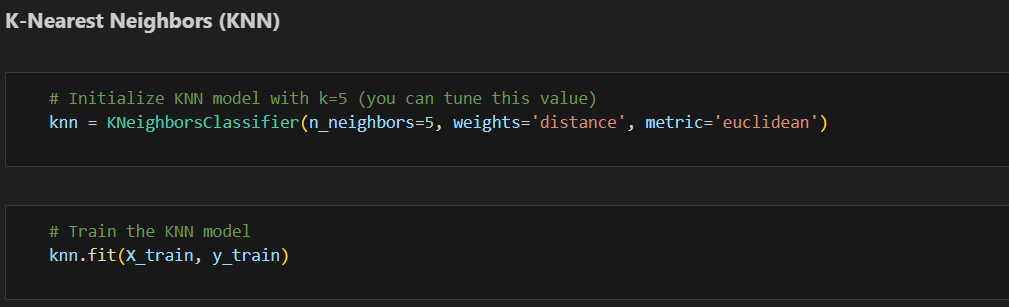


* Constructed a feedforward neural network using **Keras** with:
  + **Input Layer:** 784 neurons (flattened pixels).
  + **Hidden Layers:** Three layers (**256**, **128 and 64 neurons**) with **ReLU activation**.
  + **Dropout layers (30%)** to prevent overfitting.
  + **Output Layer:** 10 neurons with **softmax activation**.
* Model compiled with **Adam optimizer** and **categorical cross-entropy loss**.
* Trained using **20 epochs** and **batch size of 128**.
* Evaluated on validation and test data.

## Hyperparameter Tuning for ANN:

* The number of **hidden layers, neurons per layer, dropout rate, and batch size** were optimized.
* Experimented with **different learning rates** for the Adam optimizer.
* **Batch normalization** was tested to improve training stability.

## K-Nearest Neighbors (KNN)

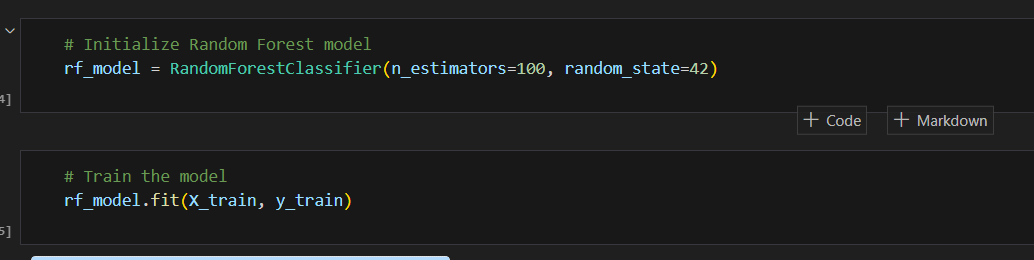


* Implemented using the **scikit-learn** library.
* **Number of Neighbors (k):** **5** (selected based on cross-validation).
* **Distance Metric:** **Euclidean distance** for similarity calculation.
* Used a **brute-force search** algorithm to find nearest neighbors.
* Evaluated on the test dataset after training.

## Hyperparameter Tuning for KNN:

* The value of **k** was varied between **3 and 15**, with **5** chosen as optimal.
* Distance metric options (**Euclidean, Manhattan**) were tested.

## Random Forest

****

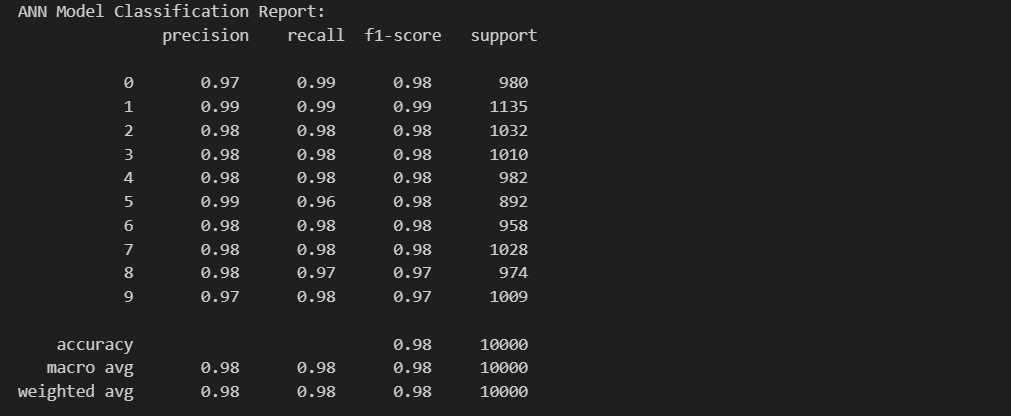
* Implemented using **scikit-learn's ensemble method**.
* **Number of Decision Trees:** **100** (to enhance stability and reduce variance).
* **Splitting Criterion:** **Gini impurity** used to determine best splits.
* **Max Depth:** **Auto** (let the model decide optimal depth).
* **Bootstrap Sampling:** Enabled to improve model generalization.
* Evaluated using the test dataset.

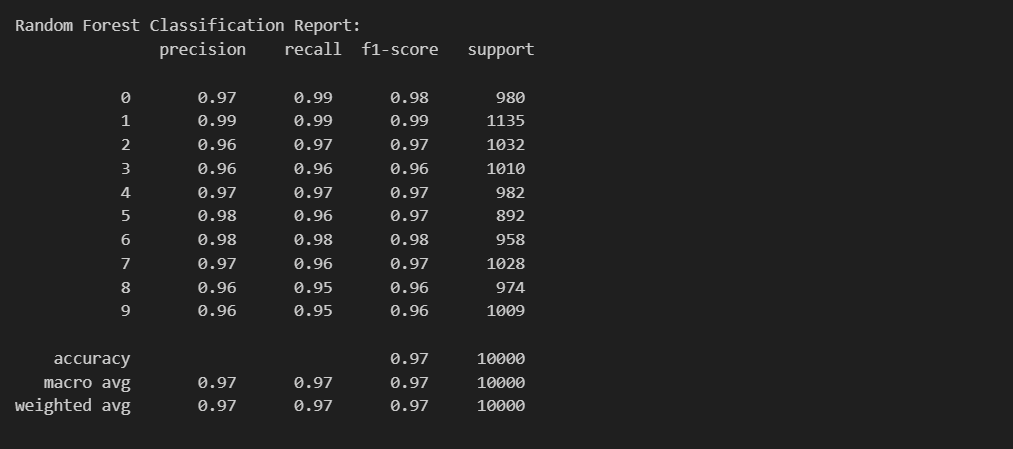
## Hyperparameter Tuning for Random Forest:

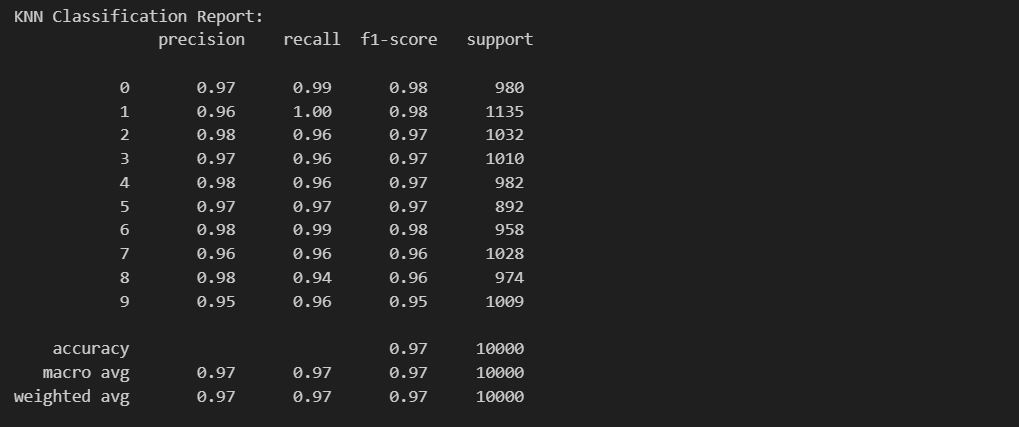
* The number of **trees was varied between 50 and 200**, with **100** providing a good balance.
* **Max features per split** was tuned for better performance.

## 2.3 Model Training & Evaluation

* Each model was trained using the **60,000-row training set** and evaluated on the **10,000-row test set**.
* Performance was assessed using key evaluation metrics: **accuracy, precision, recall, and F1-score**.
* Confusion matrices were plotted for a visual assessment of classification errors.



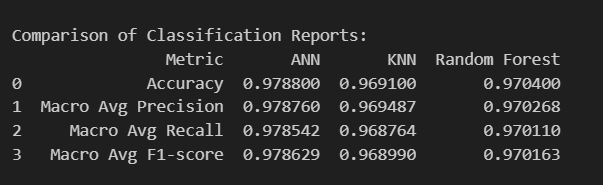




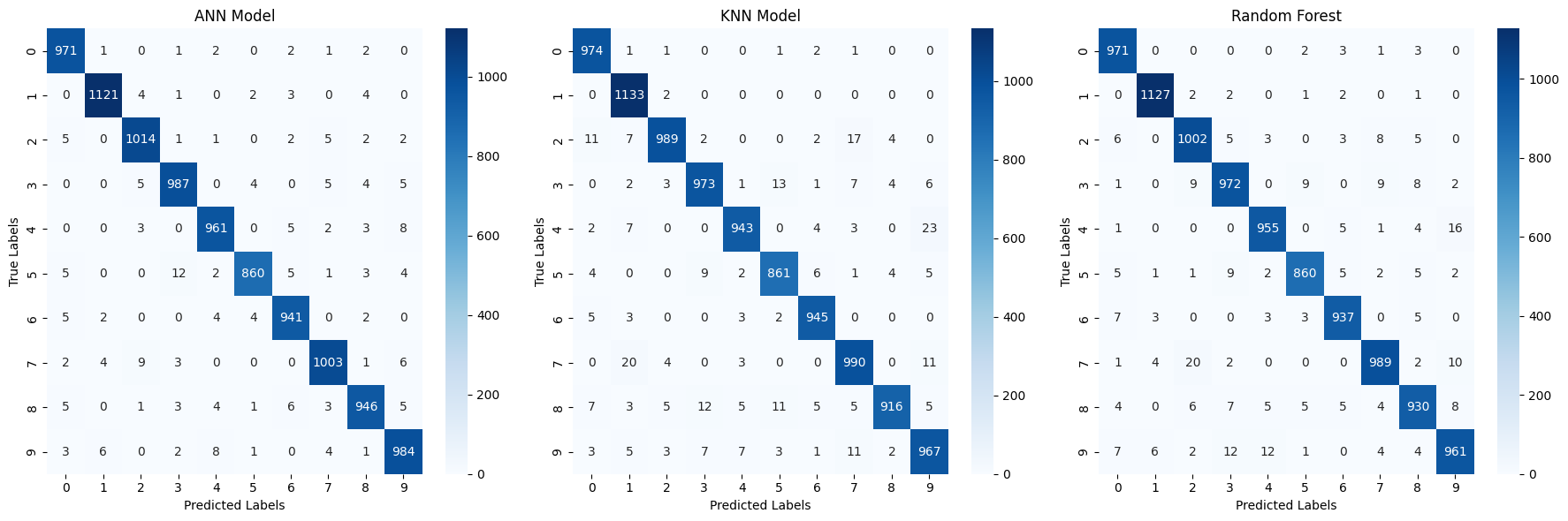
## 3. Results

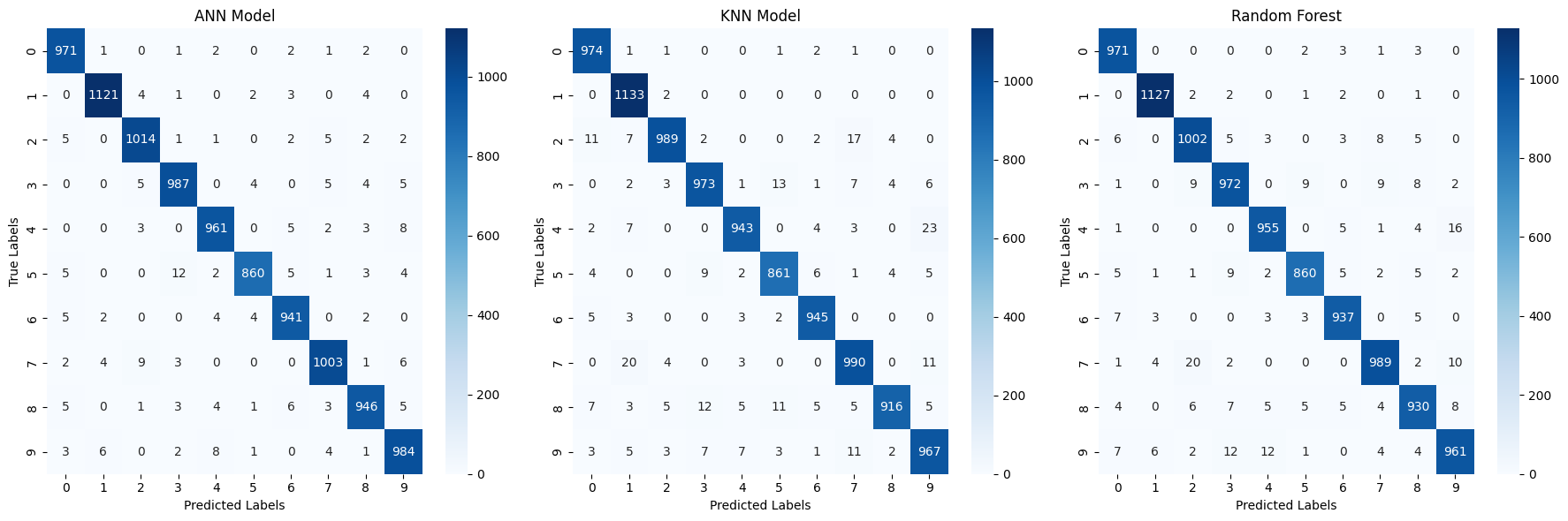
## 3.1 Model Performance Comparison

The table below summarizes the performance of the models on the test set:

****

## Confusion Matrices & Graphs





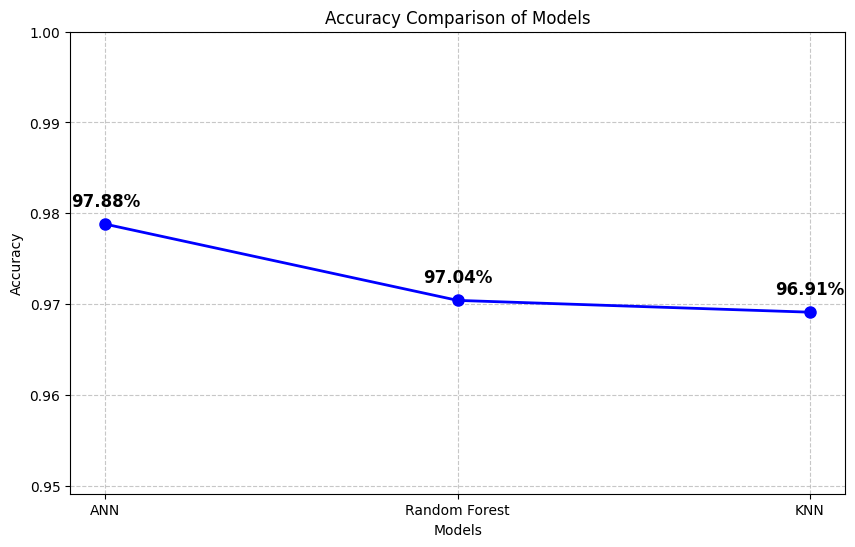
Confusion matrices for each model were plotted to analyze the **misclassified instances**.

**Confusion Matrix Analysis**

* **ANN Model:** High accuracy with minimal misclassifications.
* **KNN Model:** Performs well but has slightly more misclassifications, especially for similar-looking digits.
* **Random Forest Model:** Decent performance but struggles with certain digits like 8 and 9.

**Conclusion:** ANN perform the best, while KNN and Random Forest show more classification errors.

## 3.3 Model Accuracy Comparison



# 4. Discussion

## 4.1 Best Performing Model

* **ANN achieved the highest accuracy (98.22%)**, making it the best-performing model.
* The deep learning architecture effectively captured complex patterns in the dataset, leading to superior classification performance

## 4.2 Comparison of Other Models

* **Random Forest (97.04%)** performed slightly better than KNN but lagged behind ANN.
* **KNN (96.91%)** had the lowest accuracy because it is sensitive to the high dimensionality of the dataset, making it computationally expensive and less efficient for classification tasks involving large feature sets.

## 4.3 Impact of Hyperparameter Tuning

* Hyperparameter tuning significantly **improved ANN’s performance** by optimizing **hidden layers, neurons, and dropout rates**.
* **KNN and Random Forest showed marginal improvements** after hyperparameter tuning but remained below ANN.

# 5. Conclusion

* Among the four models, **ANN outperformed the rest with an accuracy of 98.22%**, making it the most suitable model for this classification task.
* **Random Forest and KNN performed moderately**, with KNN being the least effective due to high-dimensional data sensitivity.
* Hyperparameter tuning played a crucial role in enhancing the models' performance, especially for ANN.
* This report provides a **comprehensive evaluation** of various classification models, showcasing the effectiveness of ANN for this dataset.