**Phase-2**

**Student Name:** Hari Prasath.R

**Register Number:** 421323205017

**Institution:** Krishnasamy College of Engineering and Technology Cuddalore

**Department:** B.Tech IT

**Date of Submission:** 10-05-2025

**Github Repository Link: https://github.com/hariprasath0017/project..git**

### **1. Problem Statement**

 Recognizing image preprocessing steps like normalization and reshaping are critical.

 Understanding the importance of model accuracy, especially in real-world applications like postal automation or banking

 The input is a 2D image (28x28 pixels).

 The output is one of 10 discrete classes (digits 0–9)

 **Foundational AI Task:** Digit recognition is a classic computer vision task that serves as an entry point to image classification and deep learning.

 **Real-world Applications:**

**Postal systems:** Automated reading of ZIP codes on mail.

**Banking:** Reading handwritten check amounts.

**Forms and Surveys:** Digit recognition in handwritten forms (e.g., census data).

**2. Project Objectives**

 **Develop a Machine Learning Model** capable of classifying handwritten digits (0–9) with high accuracy.

 **Preprocess and Augment Image Data** to improve model robustness against variations in handwriting styles.

 **Evaluate Multiple Algorithms**, including traditional ML (e.g., SVM, KNN) and deep learning (e.g., CNNs), to find the best-performing approach

 **Accuracy Goal:** Achieve at least **98% classification accuracy** on the test set.

 **Real-world Readiness:** Create a model robust enough for use in real-world tasks like postal sorting or digitizing handwritten forms.

 **Efficiency:** Ensure fast inference times suitable for deployment in real-time applications.

 **Scalability:** Design the model in a modular way so it can be extended to other handwriting datasets (e.g., letters or special symbols Handling **noisy or poorly written digits** using preprocessing.

 Exploring **real-time application potential**, leading to interest in lighter or more efficient models.

 Considering **user interaction**, like digit recognition from touchscreen input, not just static images.

### 

### **3. Flowchart of the Project Workflow**

### 

### **4. Data Description**

* **Source/Origin:** Collected and curated by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges.
* **Image Data** (Unstructured in raw form, but can be structured as pixel arrays) **Grayscale** images of handwritten digits (0–9)

 **Training Set:** 60,000 images

 **Test Set:** 10,000 images*.*

* The dataset is fixed and does not change over time. Commonly used as a **benchmark** for evaluating image classification models in machine learning.
* **Target Variable Name:** label or digit. This is the correct digit corresponding to the image and is used as the ground truth in supervised learning models.

### **5. Data Preprocessing**

* *This dataset has* ***no missing values****. If there were any, we’d impute using mean (for numerical data).*
* *Duplicates are rare in MNIST/digits, but checking is a good habit. Here we drop them if found.*
* *Pixel values are bounded [0–16], so extreme outliers are unexpected. We clip as a safety check.*
* *All pixel values are float/int — this dataset is already clean in terms of data types.*
* *Most scikit-learn models accept integer class labels, but one-hot is required for neural networks (like in TensorFlow/Keras).*
* *Standardizing pixel values improves performance of models like SVM, k-NN, or neural networks. For deep learning, normalizing to [0,1] is also common.*

### 

### **6. Exploratory Data Analysis (EDA)**

* *Univariate Analysis:*
  + *All digits (0–9) are fairly equally represented, so the dataset is* ***balanced*** *— no need for resampling.*
  + *Most pixel values are near* ***0*** *(background), with fewer high-intensity values (strokes).*
  + *High variance in pixel values — especially center pixels, likely to capture key parts of digits.*
* *Bivariate/Multivariate Analysis:*
  + *Correlation is higher between adjacent pixels (consistent with strokes across pixels).*
  + *Some pixel combinations can help visually separate certain digits, showing potential model separability.*
  + *Different digits activate different pixels — this helps models differentiate between them.*
* *Insights Summary:*
  + *Most pixels are near 0 (background), but center pixels are more informative.*
  + *Neighboring pixels are correlated, useful for spatial pattern recognition.*

### **7. Feature Engineering**

* *This adds structural information beyond raw pixels and can help distinguish similar digits (e.g., "1" vs "7", "0" vs "6").*
* *Captures localized intensity information — helpful since certain digits have strokes concentrated in certain areas (e.g., "7" top-heavy, "2" bottom curve).*
* *Reduces sensitivity to slight variations in brightness and can help models generalize.*
* *Removes noise and redundant information, improves training speed, and may enhance model generalization by focusing on main variance directions.*
* *Helps visualize clustering structure and decide which new features may improve separation.*

### **8. Model Building**

### **Model 1: K-Nearest Neighbors (KNN)**

* + Simple, non-parametric, works well with image datasets.
  + Uses pixel similarity (or engineered feature similarity), which is intuitive for image classification.
  + Good baseline model for handwritten digits (e.g., KNN does surprisingly well on MNIST).

### **Model 2: Random Forest Classifier**

* + Ensemble model that can capture complex, non-linear patterns.
  + Robust to overfitting compared to single Decision Trees.
  + Can handle high-dimensional data like pixel values and engineered features.
  + Feature importance scores can help in feature selection/understanding.

 **Split ratio**: Typically 80% train, 20% test.

 **Stratification**: Important to maintain balanced class distribution in train/test sets.

### **Evaluation Metrics** (for Classification):

* **Accuracy**: Overall correctness.
* **Precision**: Correctness among positive predictions (per class).
* **Recall**: Ability to find all positive cases (per class).
* **F1-Score**: Balance between precision and recall.

### **9. Visualization of Results & Model Insights**

* *Counts of true vs. predicted labels.* *Identifies which digits are confused with each other (e.g., "3" misclassified as "5").*
* *Measures model’s ability to separate classes; ideal curves hug the top-left corner.*
* *Helps interpret which image areas or features matter most*
* *"Random Forest slightly outperforms KNN, achieving ~98.4% accuracy compared to 97.2% for KNN."*

### **10. Tools and Technologies Used***.*

* *Chosen for its powerful data science ecosystem and wide support in AI/ML applications*
* *Cloud-based, free access to GPUs, and easy collaboration.* *Ideal for interactive development, visualization, and experimentation.*
* **Matplotlib & Seaborn** (Primary visualization tools used)
* For confusion matrices, ROC curves, feature importance, and model comparisons
* **TensorFlow / PyTorch** → For deep learning models (e.g., CNNs on handwritten digits)
* **OpenCV** → For image preprocessing and feature engineering
* **Streamlit / Gradio** → To build interactive apps for model demos.

### 

### **11. Team Members and Contributions**

* **HARIPRASATH R:** Leads the project, manages the timeline, and communicates with stakeholders.
* **ABDUR RAHΜΑΝ Α:** Analyzes data, builds models, and finds insights to support the project goals.
* **MOHAMED RIYAZ A:** Develops the software or application and ensures it runs smoothly.
* **VIGNESH M:** Collects, cleans, and organizes data for use by the team.