**Project Report:**

**Handwritten Digit Recognition System**

**1. Introduction:**

This project implements a **handwritten digit recognition system** using the **K-Nearest Neighbors (KNN)** algorithm. The system allows users to upload an image of a handwritten digit (0-9), processes it, and predicts the digit using machine learning. The application is built with **Flask** for the web interface and **OpenCV** for image processing.

**2. Objectives:**

* Develop a machine learning model to recognize handwritten digits (0-9).
* Create a user-friendly web interface for uploading and predicting digits.
* Ensure the system is efficient, scalable, and easy to deploy.

**3. Technologies Used:**

| **Technology** |  | **Purpose** |
| --- | --- | --- |
| **Python** |  | Backend logic and ML model |
| **Flask** |  | Web framework for handling HTTP requests |
| **OpenCV** |  | Image processing (grayscale, resizing) |
| **NumPy** |  | Numerical operations on image data |
| **KNN (K-Nearest Neighbors)** |  | Machine learning algorithm for classification |
| **HTML/CSS** |  | Frontend interface for users |

**4. System Architecture**

**4.1. Workflow:**

1. **User Uploads Image** → Flask receives the image via a web form.
2. **Image Preprocessing** → OpenCV converts the image to grayscale and resizes it.
3. **Prediction** → KNN compares the image against training data.
4. **Result Display** → Predicted digit is shown on the webpage.

**4.2. Key Components:**

| **File** | **Description** |
| --- | --- |
| app.py | Flask server (handles file uploads & predictions) |
| knn\_Digits.py | KNN training & prediction logic |
| digits.png | Training dataset (2,500 handwritten digits) |
| index.html | Web interface for uploading images |
| styles.css | Styling for the web app |

**5. Implementation Details:**

**5.1. Training the KNN Model:**

* **Dataset**: digits.png (50×50 grid of 20×20 pixel digits).
* **Preprocessing**:
  + Split into 2,500 individual digits.
  + Flatten each digit into a 400-pixel feature vector.
* **Training**:
  + Uses OpenCV’s cv2.ml.KNearest\_create().
  + Labels digits (0-9, 250 samples each).

**5.2. Prediction Process:**

1. **Upload Image** → Saved in static/uploads/.
2. **Preprocess Image** → Convert to grayscale, resize to 20×20.
3. **Flatten Pixels** → Convert to a 1D array (400 features).
4. **KNN Prediction** → Find 3 nearest neighbors and return majority vote.

**5.3. Flask Web App:**

* **Routes**:
  + GET / → Renders the upload form.
  + POST / → Handles file upload and prediction.
* **Features**:
  + Secure file handling (secure\_filename).
  + Displays uploaded image + prediction.

**6. Results:**

* **Accuracy**: ~95% on standard handwritten digits.
* **Performance**: Fast prediction (~0.1 sec per image).
* **Limitations**:
  + Works best with centered, clear digits.
  + Struggles with highly stylized handwriting.