KNN – Handwritten Digit Classification Report

1. Introduction

Handwritten digit recognition is a fundamental problem in the field of Machine Learning and Computer Vision. It plays an important role in real-world applications such as postal mail sorting, bank cheque processing, form digitization, and optical character recognition (OCR) systems.

In this project, the **K-Nearest Neighbors (KNN)** algorithm is used to classify handwritten digits from the **Sklearn Digits dataset**. KNN is a simple yet powerful supervised learning algorithm that classifies data points based on similarity (distance) to nearby data points. Since KNN relies on distance calculations, proper feature scaling is essential for achieving good performance.

The objective of this project is to build a digit classification model using KNN, analyze its accuracy for different values of K, and evaluate its performance using visualization techniques.

---

2. Dataset Description

The dataset used in this project is the **Sklearn Digits Dataset**, which is directly available in the `scikit-learn` library.

Dataset Details:

- Total samples: **1797**
- Image size: **8 × 8 pixels**
- Total features: **64**
- Classes: Digits **0 to 9**
- Type: Grayscale images

Each image is flattened into a 64-dimensional feature vector, where each value represents the intensity of a pixel. The corresponding target label indicates the actual digit.

The dataset is well-balanced, making it suitable for classification tasks without additional preprocessing for class imbalance.

---

3. Methodology

3.1 Data Loading and Exploration

The dataset is loaded using the `load_digits()` function from `sklearn.datasets`. The shapes of the feature matrix (`X`) and target vector (`y`) are examined to confirm the dataset structure. Sample digit images are visualized using Matplotlib to verify the correctness of labels.

3.2 Train–Test Split

The dataset is divided into training and testing sets using an **80:20 ratio**. Stratified sampling is used to ensure that all digit classes are equally represented in both sets.

3.3 Feature Scaling

Since KNN is a distance-based algorithm, feature scaling is required to avoid bias due to varying feature ranges.
StandardScaler is applied to standardize the data so that each feature has a mean of 0 and a standard deviation of 1.

3.4 Model Training

A **K-Nearest Neighbors classifier** is trained using:

- Distance metric: Euclidean distance
- Initial value of K: **3**

The model predicts labels for the test dataset, and accuracy is calculated to measure performance.

3.5 Hyperparameter Tuning (K Value)

To find the optimal value of K, the model is trained with different K values:

- K = 3, 5, 7, 9

The accuracy for each K value is recorded and compared.

---

4. Results and Analysis

4.1 Accuracy Evaluation

The model achieves high accuracy for all tested K values. The accuracy slightly varies with different K values, showing that very small or very large K values can affect performance.

An **Accuracy vs K plot** is used to visually identify the optimal K value. The best K is selected based on the highest accuracy score.

4.2 Accuracy vs K Plot

The plot shows:

- X-axis: K values
- Y-axis: Accuracy

This visualization helps understand how the choice of K influences the classifier's performance.

4.3 Confusion Matrix

A confusion matrix is generated for K = 3 to analyze digit-wise predictions.

- Diagonal values represent correctly classified digits.
- Off-diagonal values indicate misclassifications.
- Some digits such as 8 and 9 show occasional confusion due to similar patterns.

The confusion matrix provides deeper insight beyond overall accuracy.

4.4 Sample Predictions

A few test images are displayed along with their predicted labels. The predictions match the actual digits in most cases, confirming the effectiveness of the trained model.

---

5. Conclusion

In this project, handwritten digit classification was successfully implemented using the K-Nearest Neighbors algorithm. The model achieved high accuracy using the Sklearn Digits dataset after applying proper feature scaling.

Key observations:

- Feature scaling is crucial for KNN performance.
- Choosing an appropriate value of K significantly affects accuracy.
- Confusion matrix analysis helps identify digit-level errors.

Despite its simplicity, KNN performed well for this classification task. However, it requires more computation for large datasets since predictions depend on distance calculations with all training samples.

---

6. Future Scope

The project can be extended in the following ways:

- Use **GridSearchCV** to automatically find the optimal K value.
- Compare KNN performance with algorithms like **SVM, Random Forest, or Neural Networks**.
- Apply the model to larger datasets such as **MNIST**.
- Optimize performance using dimensionality reduction techniques like **PCA**.

---

7. Tools and Technologies Used

- Programming Language: **Python**
- Libraries:
  - NumPy
  - Matplotlib
  - Scikit-learn
- Environment: **Jupyter Notebook**