4. Handwritten Digit Recognition

Description: Classify digits (0–9) from images of handwritten digits using ML.

Steps: -

•Load the digits dataset from sklearn.datasets or use MNIST.

•Normalize image pixel values.

•Train a classifier (like RandomForest or SVM).

•Test with a separate test set.

•Display predictions with actual digit images using matplotlib.

Project Code

#handwritten_digit_recognition/main.py

```python
#handwritten_digit_recognition/main.py
from src.load_data import load_and_preprocess_data
from src.train_model import train_classifier
from src.evaluate_model import evaluate_classifier
from src.visualize_results import visualize_predictions

def main():
    # 1. Load and preprocess data
    X_train, X_test, y_train, y_test, images_test = load_and_preprocess_data()

    # 2. Train model
    model = train_classifier(X_train, y_train)

    # 3. Evaluate model
    accuracy, predictions = evaluate_classifier(model, X_test, y_test)
    print(f"Model Accuracy: {accuracy * 100:.2f}%")

    # 4. Visualize predictions
    visualize_predictions(images_test, predictions, y_test)

if __name__ == "__main__":
    main()
```

#handwritten_digit_recognition/src/visualize_results.py

```python
#handwritten_digit_recognition/src/visualize_results.py
import matplotlib.pyplot as plt

def visualize_predictions(images, predictions, actual_labels):
    fig, axes = plt.subplots(2, 5, figsize=(10, 5))
    for i, ax in enumerate(axes.flatten()):
        ax.imshow(images[i], cmap=plt.cm.gray_r)
        ax.set_title(f"Pred: {predictions[i]}\nActual: {actual_labels[i]}")
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```

#handwritten_digit_recognition/src/train_model.py

```python
#handwritten_digit_recognition/src/train_model.py
from sklearn.ensemble import RandomForestClassifier

def train_classifier(X_train, y_train):
    clf = RandomForestClassifier(n_estimators=100, random_state=42)
    clf.fit(X_train, y_train)
    return clf
```

#handwritten_digit_recognition/src/load_data.py

```python
#handwritten_digit_recognition/src/load_data.py
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

def load_and_preprocess_data():
    # Load digits dataset
    digits = load_digits()
    X_images = digits.images   # Shape: (n_samples, 8, 8)
    y = digits.target

    # Normalize pixel values (0–16 -> 0–1)
    X_images = X_images / 16.0

    # Flatten images for classifier
    n_samples = len(X_images)
    X_flat = X_images.reshape((n_samples, -1))

    # Train-test split (keep images for visualization)
    X_train, X_test, y_train, y_test, images_train, images_test = train_test_split(
        X_flat, y, X_images, test_size=0.2, random_state=42
    )

    return X_train, X_test, y_train, y_test, images_test
```

#handwritten_digit_recognition/src/evaluate_model.py

```python
#handwritten_digit_recognition/src/evaluate_model.py
from sklearn.metrics import accuracy_score

def evaluate_classifier(model, X_test, y_test):
    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    return accuracy, predictions
```

Project Output

```
PS P:\CODE-XI\AI-Py-Learn\internship-spark-iit\tasks for ai\handwritten_digit_recognition> python -u "p:\CODE-XI\AI-Py-Learn\internship-spark-iit\tasks for ai\handwritten_digit_recognition\main.py"
Model Accuracy: 97.22%
```