AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
International Credit Hours Programs (ICHEP)
Mechatronics and Automation

Machine Vision (CSE480)

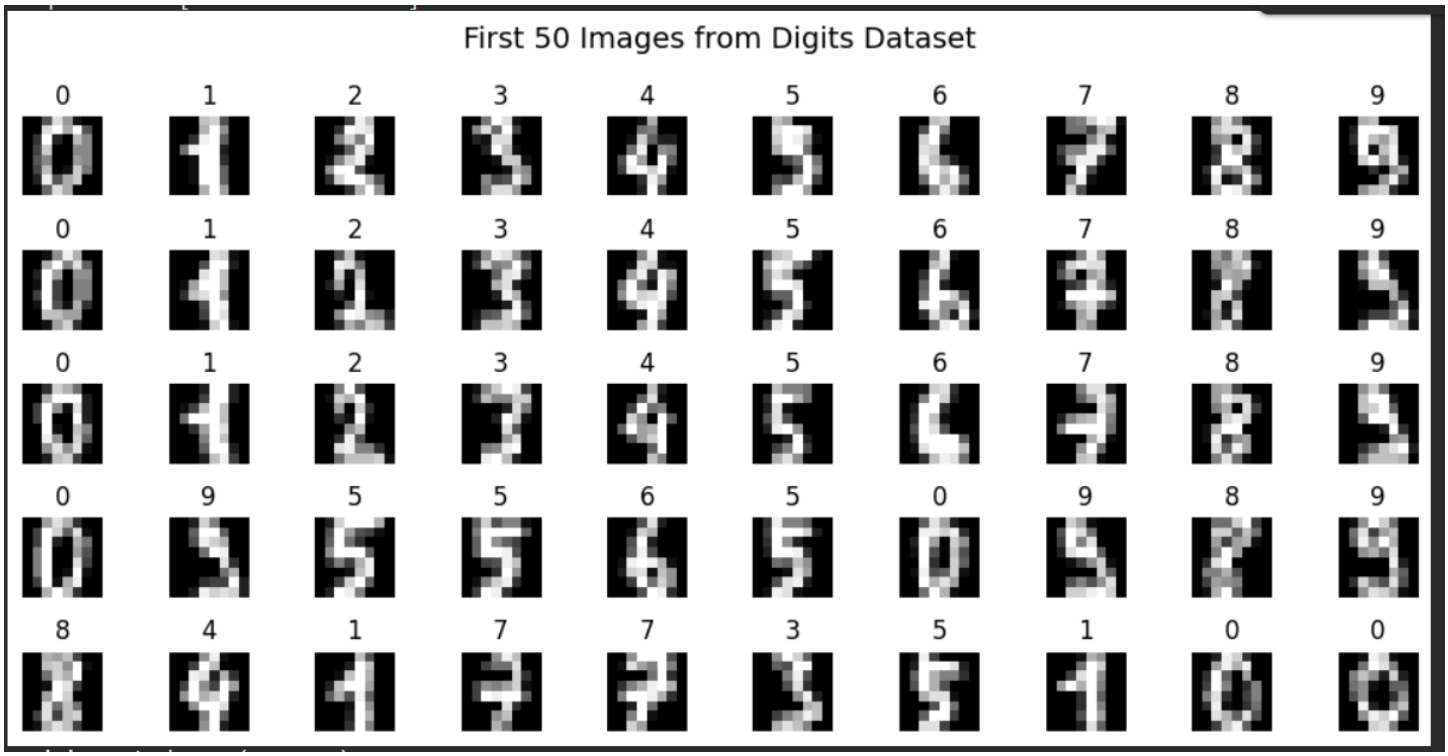
Lab 6 Report

Name	ID	Section
Mahmoud Elsayd Abdelqader Labib Eldwakhly	21P0017	1

Submitted to Dr Hossam Hassan & Eng. Dina Zakaria

Fall 2025

Output :



Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	43
1	0.93	1.00	0.96	37
2	0.93	0.97	0.95	38
3	0.98	0.93	0.96	46
4	0.98	0.98	0.98	55
5	0.98	0.97	0.97	59
6	0.96	1.00	0.98	45
7	1.00	0.98	0.99	41
8	0.97	0.95	0.96	38
9	0.93	0.90	0.91	48
accuracy			0.97	450
macro avg	0.97	0.97	0.97	450
weighted avg	0.97	0.97	0.97	450


```
Data shape: (1797, 64)
Images shape: (1797, 8, 8)
Target shape: (1797,)
Unique labels: [0 1 2 3 4 5 6 7 8 9]
```

```
Training set shape: (1347, 64)
Test set shape: (450, 64)
```

Introduction

In this task, we worked with the **Digits dataset** provided by Scikit-learn. The dataset contains small grayscale images of handwritten numbers from **0 to 9**. Each image has a size of **8 × 8 pixels**. The main goal was to understand the dataset, visualize the images, and build a simple machine learning model to recognize digits automatically.

Dataset Exploration

First, the dataset was loaded and explored. We found that:

- Each image is converted into **64 numerical features**
- Each image has a correct label representing the digit it shows
  We displayed the **first 50 images** to visually confirm that the data is correct and understandable. This step helped us trust the dataset before training the model.

Data Preparation

The data was split into:

- **75% training data**
- **25% testing data**

We then applied **standardization** to the data. This step is important because the K-Nearest Neighbors (KNN) algorithm depends on distances, and scaling ensures that all features contribute fairly.

Model Training

We used the **K-Nearest Neighbors (KNN)** algorithm with **k = 3**. The model learns by comparing a new digit with the closest digits it has seen before and choosing the most common class among them.

Code :

```
# ================================
# Digits Dataset By Mahmoud Elsayd - 21P0017
# ================================

import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# ------------------------------
# 1. Load and Explore the Dataset
# ------------------------------
digits = load_digits()

print("Dataset keys:", digits.keys())
print("Data shape:", digits.data.shape)
print("Images shape:", digits.images.shape)
print("Target shape:", digits.target.shape)
print("Unique labels:", np.unique(digits.target))

# ------------------------------
# 2. Show the First 50 Images
# ------------------------------
fig, axes = plt.subplots(5, 10, figsize=(10, 5))

for i, ax in enumerate(axes.flat):
    ax.imshow(digits.images[i], cmap='gray')
    ax.set_title(digits.target[i])
    ax.axis('off')

plt.suptitle("First 50 Images from Digits Dataset", fontsize=14)
plt.tight_layout()
plt.show()

# ------------------------------------------
# 3. Train-Test Split (test size = 0.25)
# ------------------------------------------
X = digits.data
y = digits.target

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)

print("Training set shape:", X_train.shape)
print("Test set shape:", X_test.shape)

# ------------------------------
# 4. Standardization
# ------------------------------
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ------------------------------
# 5. Train KNN (k = 3)
# ------------------------------
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)

# ------------------------------
# 6. Predictions & Accuracy
```

```python
# ------------------------------
y_pred = knn.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)

print(f"\nTest Accuracy (k=3): {accuracy:.4f}")

# ------------------------------
# 7. Confusion Matrix
# ------------------------------
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
plt.imshow(cm, cmap='Blues')
plt.colorbar()
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")

plt.xticks(range(10))
plt.yticks(range(10))

for i in range(10):
    for j in range(10):
        plt.text(j, i, cm[i, j],
                 ha="center", va="center",
                 color="white" if cm[i, j] > cm.max() / 2 else "black")

plt.tight_layout()
plt.show()

# ------------------------------
# 8. Classification Report
# ------------------------------
print("Classification Report:\n")
print(classification_report(y_test, y_pred))
```