# Handwritten Digit Recognition using SVM and k-NN

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

# Step 1: Load the Dataset

digits = datasets.load\_digits() # Built-in dataset (8x8 images, 1797 samples)

X = digits.data # Feature matrix (each image as flattened 8x8=64 vector)

y = digits.target # Labels (0-9)

print("Dataset shape:", X.shape)

print("Unique labels:", np.unique(y))

# Visualize some samples

plt.figure(figsize=(6, 4))

for i in range(8):

plt.subplot(2, 4, i+1)

plt.imshow(digits.images[i], cmap='gray')

plt.title(f'Label: {digits.target[i]}')

plt.axis('off')

plt.show()

# Step 2: Split the Dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Step 3: SVM Model

svm\_model = SVC(kernel='rbf', gamma=0.001, C=10)

svm\_model.fit(X\_train, y\_train)

y\_pred\_svm = svm\_model.predict(X\_test)

# Evaluate SVM

print("\n🔹 SVM Model Results 🔹")

print("Accuracy:", accuracy\_score(y\_test, y\_pred\_svm))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred\_svm))

# Confusion Matrix for SVM

plt.figure(figsize=(6,4))

sns.heatmap(confusion\_matrix(y\_test, y\_pred\_svm), annot=True, fmt='d', cmap='Blues')

plt.title("SVM Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

# Step 4: k-NN Model

knn\_model = KNeighborsClassifier(n\_neighbors=5)

knn\_model.fit(X\_train, y\_train)

y\_pred\_knn = knn\_model.predict(X\_test)

# Evaluate k-NN

print("\n🔹 k-NN Model Results 🔹")

print("Accuracy:", accuracy\_score(y\_test, y\_pred\_knn))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred\_knn))

# Confusion Matrix for k-NN

plt.figure(figsize=(6,4))

sns.heatmap(confusion\_matrix(y\_test, y\_pred\_knn), annot=True, fmt='d', cmap='Greens')

plt.title("k-NN Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

# Step 5: Compare Performance

svm\_acc = accuracy\_score(y\_test, y\_pred\_svm)

knn\_acc = accuracy\_score(y\_test, y\_pred\_knn)

print("\n✅ Model Comparison")

print(f"SVM Accuracy : {svm\_acc\*100:.2f}%")

print(f"k-NN Accuracy: {knn\_acc\*100:.2f}%")

# Visualize comparison

plt.bar(['SVM', 'k-NN'], [svm\_acc, knn\_acc], color=['skyblue', 'lightgreen'])

plt.title("Model Accuracy Comparison")

plt.ylabel("Accuracy")

plt.show()











