Hand Written Digit Prediction

Objective

The objective of the code is to perform handwritten digit prediction using the MNIST dataset. It trains a Support Vector Machine (SVM) classifier on the training data and evaluates its performance on the test data.

Data Source

The data source used in the code is the MNIST dataset, which is a popular dataset of handwritten digits.

Import Library

import numpy as np

import matplotlib.pyplot as plt

Import Data

from sklearn import datasets

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

from sklearn.svm import SVC

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

Describe Data

print("MNIST Handwritten Digits Dataset:")

print("Number of samples:", len(digits.data))

print("Image data shape:", digits.data.shape)

print("Target labels:", np.unique(digits.target))

Data Visualization

fig, axes = plt.subplots(2, 5, figsize=(10, 4))

for i, ax in enumerate(axes.ravel()):

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

ax.set\_title(f"Label: {digits.target[i]}")

ax.axis('off')

plt.tight\_layout()

plt.show()

Data Preprocessing

scaler = StandardScaler()

X = scaler.fit\_transform(X)

# Split the dataset into training and testing sets

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

# Create an instance of SVM classifier

svm\_classifier = SVC(kernel='linear', C=1.0)

# Train the classifier on the training data

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

# Make predictions on the test data

y\_pred = svm\_classifier.predict(X\_test)

Define Target Variable (y) and Feature Variables (X)

# D # Define Target Variable (y) and Feature Variables (X)

X = digits.data

y = digits.target

# Data Preprocessing: Standardization

scaler = StandardScaler()

X = scaler.fit\_transform(X)

# Split the dataset into training and testing sets

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

# Create an instance of SVM classifier

svm\_classifier = SVC(kernel='linear', C=1.0)

# Train the classifier on the training data

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

# Make predictions on the test data

y\_pred = svm\_classifier.predict(X\_test)

# Calculate the accuracy of the classifier

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

# Plot a confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.imshow(conf\_matrix, cmap=plt.cm.Blues, interpolation='nearest')

plt.colorbar()

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Confusion Matrix')

plt.show()

Train Test Split

# Train Test Split

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

Modeling

# Create an instance of SVM classifier

svm\_classifier = SVC(kernel='linear', C=1.0)

# Train the classifier on the training data

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

# Make predictions on the test data

y\_pred = svm\_classifier.predict(X\_test)

Model Evaluation

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

print(conf\_matrix)

PREDICTION

# Train the classifier on the training data

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

# Make predictions on the test data

y\_pred = svm\_classifier.predict(X\_test)

EXPLANATION

Import Libraries: The code begins by importing the required Python libraries, including NumPy, Matplotlib, scikit-learn datasets, model\_selection, svm, metrics, and preprocessing.

Load the MNIST Dataset: The code loads the MNIST dataset using datasets.load\_digits() from scikit-learn. The dataset contains images of handwritten digits (0 to 9) and their corresponding labels.

Data Description and Visualization: The code prints some basic information about the dataset, such as the number of samples, image data shape, and unique target labels (digits). It also visualizes some sample images from the dataset using Matplotlib.

Define Target Variable (y) and Feature Variables (X): The code separates the target variable y (labels) and the feature variables X (pixel values of images) from the loaded dataset.

Data Preprocessing: The code applies data preprocessing by standardizing the feature variables X using StandardScaler from scikit-learn. Standardization scales the features to have a mean of 0 and a standard deviation of 1, which can improve the performance of some machine learning algorithms.

Train Test Split: The code splits the dataset into training and testing sets using the train\_test\_split() function from scikit-learn. It assigns 80% of the data to the training set (X\_train and y\_train) and 20% to the test set (X\_test and y\_test).

Modeling: The code creates an instance of the SVM classifier with a linear kernel and regularization parameter C set to 1.0. It then trains the classifier on the training data (X\_train and y\_train) using the fit() method.

Model Evaluation: The code makes predictions on the test data using the trained SVM classifier (svm\_classifier.predict(X\_test)) and calculates the accuracy of the classifier on the test data using accuracy\_score() from scikit-learn. It also calculates the confusion matrix using confusion\_matrix() from scikit-learn and prints it.

Plotting the Confusion Matrix: The code uses Matplotlib to plot the confusion matrix, visualizing how well the SVM classifier performs for each class (digit).

Overall, this code demonstrates the process of loading a dataset, preprocessing the data, splitting it into training and testing sets, training an SVM classifier on the training data, and evaluating its performance on the test data using accuracy and a confusion matrix. The SVM classifier is used for handwritten digit recognition, and the code helps assess how well the classifier can identify different digits.