Title of the project :- Hand Written Digit Prediction -

Classification Analysis

Objective:- The digits dataset consists of 8x8 pixel images of digits. The images attribute of the dataset stores 8x8 arrays of grayscale values for each image. We will use these arrays to visualise the first 4 images. The target attribute of the dataset stores the digit each image represents.

Data Source:

TensorFlow/Keras: In Python, you can directly access the MNIST dataset using the TensorFlow/Keras library with the functiontf.keras.datasets.mnist.load_data().

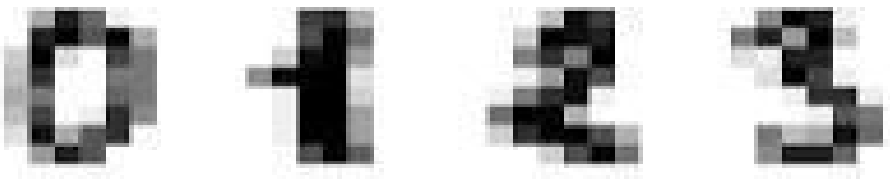
Import Library

```
[1] import pandas as pd

[2] Import numpy as np

[3] import matplotlib.pyplot as plt
```

Import Data

```
[4] from sklearn.datasets import load_digits

[5] df =load_digits()

[6] _, axes = plt.subplots(nrows=1, ncols=4, figsize=(10,3))

   for ax, image, label in zip (axes, df.images, df.target):

    ax.set_axis_off()

    ax.imshow(image, cmap-plt.cm.gray_r, interpolation="nearest")

    ax.set_title("Training: %i" % label)
```

Training 0: Training:1 Training 2: Training 3:



Describe Data

One of the most widely used datasets for this task is the MNIST dataset, which I'll use as an example to describe the typical characteristics of data for handwritten digit prediction:

Image Format: Each image in the dataset is represented as a 28x28-pixel grayscale image. Grayscale means that each pixel has a single intensity value representing the darkness or lightness of that pixel, ranging from 0 (black) to 255 (white).

Number of Samples: The MNIST dataset contains a total of 70,000 samples. This is typically divided into a training set and a test set. The training set consists of 60,000 samples, while the test set contains 10,000 samples.

Labeling: Each image is associated with a label, indicating the actual digit it represents. The labels are integers ranging from 0 to 9. For instance, the label "5" corresponds to an image of a handwritten digit "5."

Variability: The dataset captures a wide range of variability in how people write the digits. Different individuals have different handwriting styles, which makes the dataset diverse and challenging.

Data Balance: The dataset is balanced, meaning that it contains roughly the same number of samples for each digit. This ensures that the model doesn't become biased towards predicting certain digits more accurately than others.

Preprocessing: In some cases, the dataset may undergo preprocessing steps like normalization, which scales the pixel values to a range between 0 and 1. This makes it easier for machine learning algorithms to converge during training.

The goal of the handwritten digit prediction task is to train a machine learning model on the training set to learn patterns and features in the images associated with each digit. The trained model is then evaluated on the test set to assess its performance and accuracy in correctly predicting the digits in unseen handwritten images.

Data Visualistion

Sample Images Visualization: Display a grid of sample images from the dataset to get an idea of what the handwritten digits look like. This helps you understand the dataset and the variability in the handwriting styles.

Class Distribution Bar Chart: Create a bar chart showing the distribution of each digit class in the dataset. It helps to verify if the dataset is balanced (similar number of samples for each digit) or imbalanced.

Pixel Intensity Histogram: Plot histograms of pixel intensities to see the distribution of pixel values in the images. This is particularly useful for grayscale images and can help you understand the range of pixel intensities present in the dataset.

Heatmap of Confusion Matrix: After training a model, create a heatmap of the confusion matrix to visualize how well the model performs on each digit class.

Misclassified Samples: Display some misclassified samples by the model to gain insights into the challenges faced by the model in distinguishing certain digits.

Activation Maps: For deep learning models, visualize the activation maps of convolutional layers to see which parts of the image the model focuses on while making predictions.

Data Preprocessing

Loading the Dataset: Load the dataset from the source, whether it's from the official MNIST website, TensorFlow/Keras, scikit-learn, or any other data repository. Split the dataset into training and test sets.

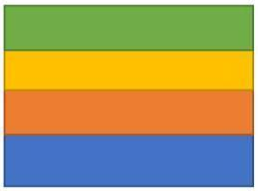
Reshaping the Data: The original images in the MNIST dataset are typically stored as 28x28 pixel arrays. Depending on the specific library you're using, you may need to reshape the data into a 2D array, where each row represents a flattened image (e.g., a 1D array of size 784 for 28x28 images).

Normalizing Pixel Values: Normalize the pixel values to bring them within a common range, usually between 0 and 1. For grayscale images, divide the pixel values by 255, which is the maximum pixel intensity value. This ensures that all the features have a similar scale and helps in the convergence of the model during training.

One-Hot Encoding Labels: Convert the label values (0 to 9) into one-hot encoded vectors. For example, the digit "3" would be represented as [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] in one-hot encoding.

Flatten image

8X8



```
[7]   df.images.shape
      (1797,8 ,8)

[8] df.images[0]

array([[ 0., 0., 5., 13., 9., 1., 0., 0.],

       [0., 0., 13., 15., 10., 15., 5.,0.],

       [0., 3., 15., 2., 0., 11., 8.,0.,]

       [0., 4., 12., 0., 0., 8., 8.,0.],

       [0., 5., 8., 0., 0., 9., 8., 0.],

       [0., 4., 11., 0., 1., 12., 7., 0.],

       [0., 2., 14., 5., 10., 12., 0., 0.],

       [0., 0., 6., 13., 10., 0., 0.,0.]])


[9] df.images[0].shape

(8,8)


[10] len(df.images)

1797

[11]  n_samples=len(df.images)
data=df.images.reshape((n_samples,-1))

[12] data[0]

array([ 0., 0., 5., 13., 9.,1., 0., 0., 0., 0., 13., 15., 10.,

15.,5.,0., 0.,3., 15., 2.,0., 11.,8., 0.,0.,4.,

12.,0., 0., 8., 8., 0.,0.,5., 8.,0 .,0.,9.,8.,

0.,0.,4., 11.,0.,1., 12.,7.,0.,0.,2., 14.,5.,

10.,12.,0.,0.,0., 0.,6.,14.,10.,0.,0.,0.,])

[13] data[0].shape

(64,)

[14]  data.shape
      (1797, 64)
```

Scaling Image data

```
[15] data.min()

0.0

[16] data.max()

16.0

[17] data=data/16

[18] data.min()

0.0

[19] data.max()

1.0

[20] data[0]

array ([0.,0.,0.3125, 0.8125, 0.5625, 0.0625, 0., 0.,

0., 0., 0.8125, 0.9375, 0.625, 0.9375, 0.3125, 0.,

0., 0.1875, 0.9375, 0.125, 0.,0.6875, 0.5, 0.,

0., 0.25, 0.75, 0., 0., 0.5, 0.5, 0.,

0., 0.3125, 0.5, 0., 0., 0.5625, 0.5, 0.,

0., 0.25, 0.6875, 0., 0.0625, 0.75, 0.4375, 0.,

0., 0.125, 0.875, 0.3125, 0.625, 0.75, 0., 0.,

0., 0., 0.375, 0.8125, 0.625, 0., 0., 0.])
```

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

Target Variable (y): The target variable, often denoted as "y," represents the labels or the actual digit values that we want the machine learning model to predict. In this case, the target variable is the digit value (0 to 9) of the handwritten image.

For example, if we have a handwritten image of the digit "5," the corresponding target variable (y) would be 5. Similarly, for an image of the digit "2," the target variable (y) would be 2.

In practice, the target variable is typically represented as a one-dimensional array or a vector containing the digit labels for all the samples in the dataset. For instance, if we have "N" samples in the dataset, the target variable (y) would be an array of length "N" containing the corresponding digit labels.

Feature Variables (X): The feature variables, often denoted as "X," represent the input data or the characteristics of the handwritten images that the machine learning model will use to make predictions. In this case, the feature variables are the pixel values of the 28x28 grayscale images.

Each handwritten digit image can be represented as a 1D array of length 784 (28x28), where each element in the array represents the pixel intensity value (ranging from 0 to 255) of a specific pixel in the image.

In practice, the feature variables (X) are represented as a 2D array or matrix, where each row corresponds to an image and contains the pixel values of that image. If we have "N" samples in the dataset, the feature variables (X) would be a 2D array of shape (N, 784).

With the target variable (y) and feature variables (X) defined, we can use them to train a machine learning model to predict the handwritten digit labels based on the pixel values of the images. The model learns the patterns and features in the feature variables to make accurate predictions of the corresponding target variables.

Train Test Split Data

```
[21]from sklearn.model_selection import train_test_split

[22]X_train, X_test, y_train, y_test = train_test_split
(data, df.target, test_size=0.3)



[23]X_train.shape, X_test.shape, y_train.shape, y_test.shape
((1257, 64), (540, 64), (1257,), (540,))
```

Random Forest Model

```
[24] from sklearn.ensemble import RandomForestClassifier

[25] rf =RandomForestClassifier()

[26] rf.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

Predict Test Data

```
[27] y_pred = rf.predict(X_test)

[28] y_pred

       array([6, 5, 2, 6, 8, 8, 2, 7, 5, 1, 1, 4, 3, 7, 2, 0, 0, 7, 0, 2, 5, 3,
              2, 5, 2, 7, 5, 9, 1, 8, 3, 3, 1, 1, 8, 4, 0, 4, 2, 1, 3, 1, 6, 9,
              8, 8, 6, 7, 5, 4, 3, 8, 5, 0, 6, 0, 7, 1, 1, 1, 6, 1, 9, 3, 8, 8,
              8, 1, 5, 0, 6, 2, 5, 9, 3, 5, 6, 0, 3, 5, 4, 0, 4, 1, 9, 1, 1, 3,
              0, 3, 7, 1, 6, 3, 9, 7, 2, 9, 6, 2, 7, 0, 6, 9, 9, 3, 5, 4, 4, 6,
              3, 6, 2, 3, 9, 6, 3, 7, 2, 7, 3, 1, 3, 6, 1, 9, 0, 5, 2, 9, 6, 1,
              3, 4, 1, 6, 7, 2, 8, 8, 2, 7, 7, 8, 7, 1, 2, 5, 0, 0, 3, 1, 5, 8,
              0, 3, 0, 1, 5, 5, 4, 8, 0, 7, 9, 5, 5, 5, 6, 4, 3, 6, 9, 3, 6, 0,
              2, 5, 9, 5, 7, 6, 4, 2, 2, 5, 7, 7, 2, 0, 8, 1, 0, 7, 6, 7, 2, 7,
              6, 6, 1, 4, 4, 7, 5, 0, 2, 9, 8, 4, 0, 0, 8, 1, 9, 4, 2, 4, 6, 1,
              0, 4, 5, 1, 6, 3, 1, 4, 8, 3, 7, 3, 3, 8, 9, 1, 2, 5, 5, 0, 7, 1,
              2, 7, 3, 0, 7, 9, 4, 7, 3, 5, 8, 1, 3, 3, 3, 6, 8, 3, 3, 6, 1, 1,
              3, 7, 8, 6, 9, 9, 6, 7, 0, 3, 9, 8, 8, 7, 9, 2, 7, 1, 5, 3, 2, 2,
              1, 4, 2, 5, 2, 0, 9, 5, 1, 2, 1, 8, 8, 3, 0, 8, 4, 2, 9, 8, 3, 2,
              6, 7, 1, 1, 4, 8, 9, 5, 1, 5, 6, 4, 0, 7, 3, 7, 4, 7, 7, 9, 5, 0,
              5, 0, 9, 2, 0, 1, 1, 9, 3, 7, 4, 7, 0, 0, 3, 9, 6, 2, 8, 8, 9, 0,
              6, 7, 1, 2, 6, 3, 9, 8, 1, 6, 9, 3, 5, 1, 6, 6, 7, 9, 1, 0, 1, 2,
              5, 2, 1, 4, 2, 8, 9, 5, 1, 7, 3, 2, 0, 9, 8, 3, 6, 4, 2, 2, 9, 1,
              2, 8, 4, 0, 4, 2, 8, 3, 8, 0, 6, 2, 0, 1, 5, 7, 6, 5, 5, 8, 6, 5,
              5, 6, 2, 5, 8, 8, 4, 7, 4, 0, 7, 0, 0, 2, 0, 8, 4, 8, 1, 6, 4, 2,
              7, 0, 4, 7, 1, 1, 1, 2, 6, 2, 8, 6, 1, 9, 1, 2, 0, 3, 6, 0, 5, 5,
              9, 7, 0, 0, 6, 3, 4, 5, 3, 9, 9, 8, 1, 2, 4, 4, 5, 4, 2, 5, 0, 6,
              9, 3, 1, 5, 0, 5, 7, 2, 2, 6, 5, 2, 4, 9, 7, 3, 8, 5, 1, 2, 0, 5,
              2, 9, 2, 3, 4, 8, 5, 7, 7, 4, 0, 0, 1, 9, 2, 1, 6, 7, 3, 8, 7, 5,
```

```
            2, 0, 5, 0, 0, 4, 8, 5, 5, 7, 9, 3]
```

Model Accuracy

```
[29] from sklearn.metrics import confusion matrix, classification_report

[30] confusion matrix (y_test, y_pred)

 Array ([[60, 0, 0, 0, 1, 0, 0, 0, 0,0],
 [0, 60, 0, 1, 0, 0, 0, 0, 0, 0],
 [0, 0, 61, 0, 0, 0, 0, 0, 0, 1],
 [0, 0, 0, 54, 0, 0, 0, 1, 0, 0],
 [0, 0, 0, 0, 40, 0, 0, 1, 0, 0],
 [0, 0, 0, 0, 0, 58, 1, 0, 1, 0],
 [0, 1, 0, 0, 1, 0, 50, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 52, 0, 0],
 [0, 2, 0, 0, 1, 1, 0, 0, 46, 0],
 [0, 0, 0, 0, 0, 0, 0, 1, 0, 45]])


 [31] print (classification_report(y_test, y_pred))
```

Explanation

Handwritten Digit Prediction is a machine learning task in which the goal is to create a model capable of recognizing and predicting the handwritten digits (0 to 9) present in images. The task falls under the broader category of image classification, where the model learns to classify input images into specific categories or classes.

The primary steps involved in Handwritten Digit Prediction are as follows:

Data Collection: The first step is to gather a dataset of handwritten digit images along with their corresponding labels. One of the most commonly used datasets for this task is the MNIST dataset, which contains 28x28 grayscale images of handwritten digits.

Data Preprocessing: Before feeding the data into the machine learning model, preprocessing steps are performed. These steps may include reshaping the images into a suitable format, normalizing pixel values to a common scale, and one-hot encoding the digit labels.

Model Selection: There are various machine learning algorithms that can be used for image classification tasks like Handwritten Digit Prediction. Popular choices include Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Random Forests, and deep learning models like Convolutional Neural Networks (CNNs).

Model Training: The model is trained using the preprocessed data. During training, the model learns to recognize patterns and features in the images associated with each digit. It optimizes its internal parameters to minimize prediction errors.

Model Evaluation: After training, the model is evaluated using a separate test set that it has never seen before. Evaluation metrics such as accuracy, precision, recall, and F1-score are used to assess the model's performance in correctly predicting the digits.

Prediction: Once the model is trained and evaluated, it can be used to make predictions on new, unseen handwritten digit images. The model takes the pixel values of an image as input and outputs a predicted digit label.

Deployment: If the model achieves satisfactory performance, it can be deployed in real-world applications to automatically recognize handwritten digits. This technology finds applications in fields like postal services (reading postal codes), digitizing documents, and digit recognition in various forms.

Deep learning models, particularly CNNs, have shown exceptional performance in Handwritten Digit Prediction due to their ability to automatically learn hierarchical representations of the images. They have significantly contributed to achieving high accuracy on the MNIST dataset and similar tasks.

Handwritten Digit Prediction serves as a foundational task in image classification and computer vision. It not only helps in developing technologies for recognizing digits but also forms the basis for more complex image recognition tasks, such as recognizing objects, faces, and scenes in images.