**Project Report: Handwritten Digit Recognition using Logistic Regression**

**Introduction:**

The objective of this project is to build a handwritten digit recognition system using logistic regression. We will use the MNIST dataset, which contains images of handwritten digits (0-9) and their corresponding labels. The system will be trained to classify these digits accurately.

**Background:**

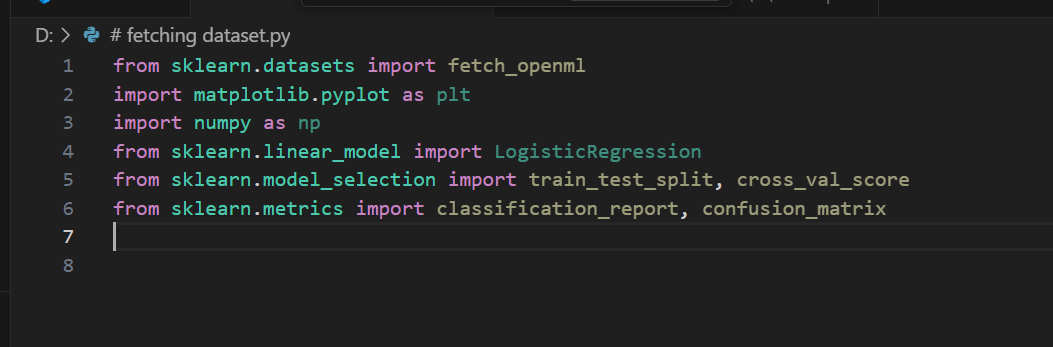
Handwritten digit recognition is a classic problem in the field of computer vision and machine learning. The MNIST dataset, introduced by Yann LeCun and colleagues, is one of the most widely used datasets for image classification tasks. It serves as a benchmark for evaluating the performance of various algorithms.

Logistic regression is a statistical model that, in its basic form, uses a logistic function to model a binary dependent variable. In the context of digit recognition, logistic regression can be extended to multi-class classification using the "one-vs-rest" (OvR) strategy, where a separate binary classifier is trained for each class.

**Step-by-Step Explanation of the Code:**

**1. Import Necessary Libraries:**

We start by importing the required libraries.



- `fetch\_openml`: To fetch the MNIST dataset.

- `matplotlib.pyplot`: For plotting images.

- `numpy`: For numerical operations.

- `LogisticRegression`: For creating the logistic regression model.

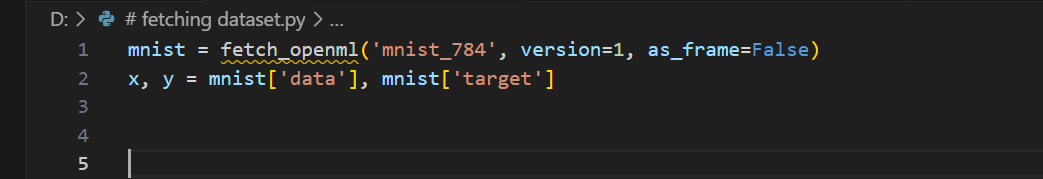
- `train\_test\_split`: To split the data into training and test sets.

- `cross\_val\_score`: To perform cross-validation.

- `classification\_report` and `confusion\_matrix`: To evaluate the model's performance.

**2. Fetch the Dataset:**

Fetch the MNIST dataset, which contains images of handwritten digits.

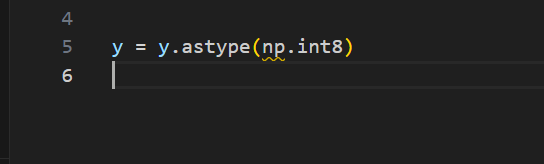


- `mnist\_784`: The identifier for the MNIST dataset on OpenML.

- `as\_frame=False`: To fetch the data as NumPy arrays.

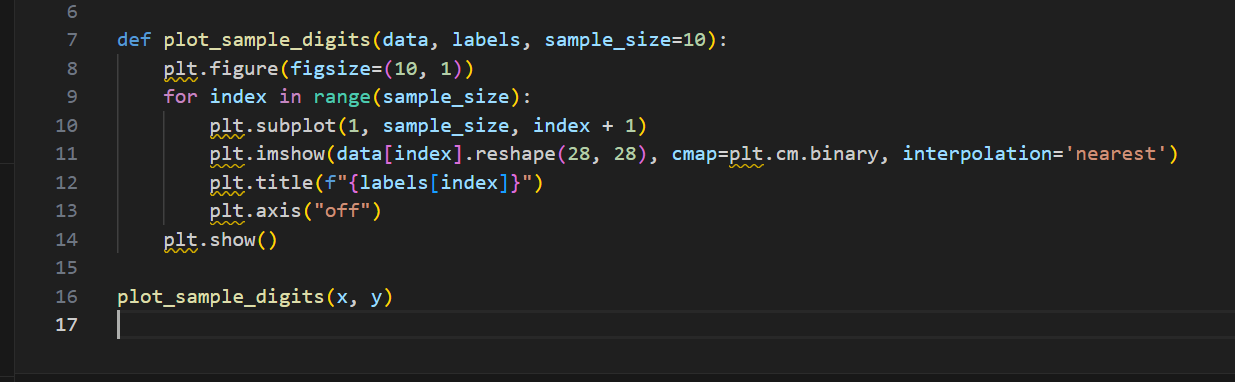
**3. Convert Labels to Integers:**

Convert the labels from string format to integers for compatibility with scikit-learn models.



**4. Display Multiple Sample Digits:**

Plot the first 10 digits from the dataset to visualize the data.



- `plot\_sample\_digits`: A function to plot a specified number of sample digits.

**5. Prepare the Data:**

Split the dataset into training and test sets.

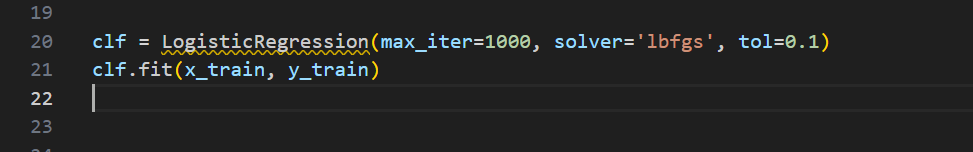


- `train\_test\_split`: To split the data into training (80%) and test (20%) sets.

- `random\_state=42`: Ensures reproducibility of the split.

**6. Train the Logistic Regression Classifier:**

Train a logistic regression model using the training data.



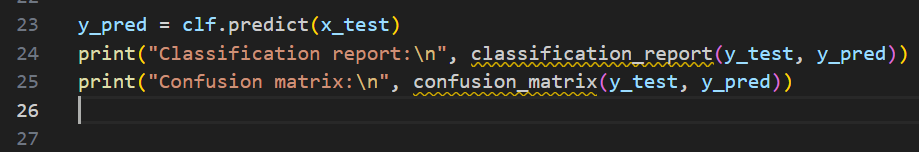
- `max\_iter=1000`: Sets the maximum number of iterations.

- `solver='lbfgs'`: Specifies the optimization algorithm.

- `tol=0.1`: Sets the tolerance for stopping criteria.

**7. Evaluate the Classifier:**

Generate predictions on the test set and evaluate the model's performance.



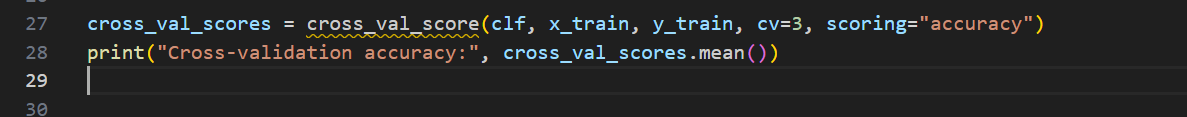
- `y\_pred`: Predictions for the test set.

- `classification\_report`: Provides precision, recall, F1-score, and support for each class.

- `confusion\_matrix`: Shows the number of correct and incorrect predictions.

**8. Cross-validation:**

Perform cross-validation on the training set to assess the model's stability.



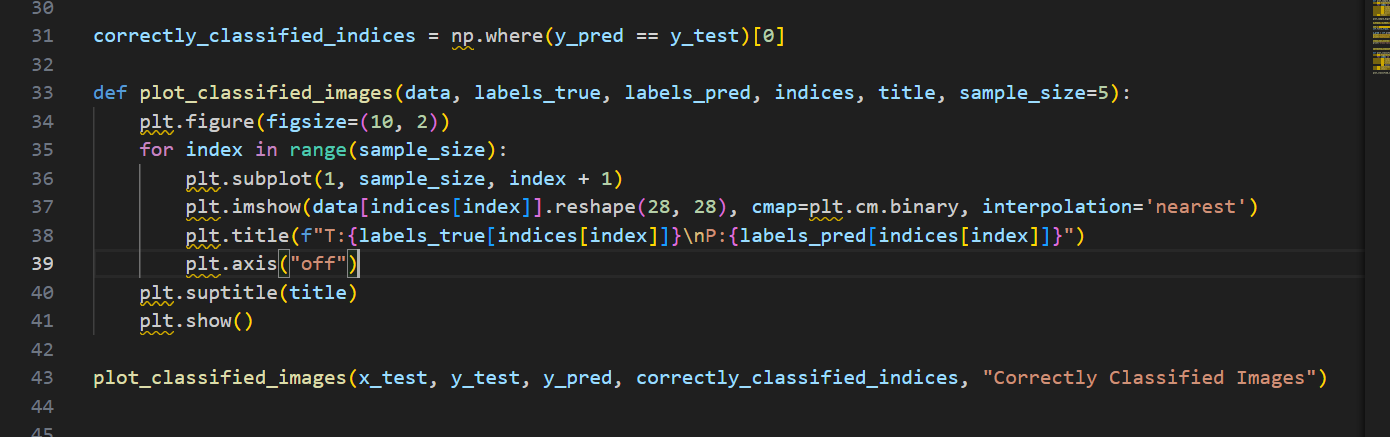
- `cross\_val\_score`: Performs 3-fold cross-validation.

- `cv=3`: Specifies 3-fold cross-validation.

- `scoring="accuracy"`: Measures accuracy.

**9. Display Correctly Classified Images:**

Visualize some correctly classified images from the test set.

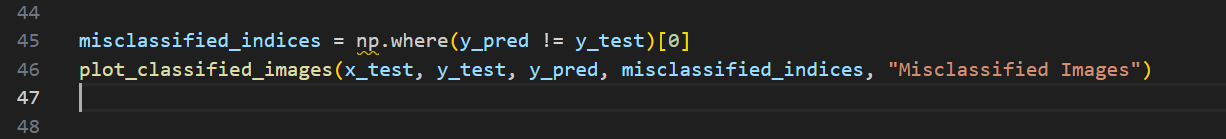


- `np.where(y\_pred == y\_test)`: Identifies correctly classified indices.

- `plot\_classified\_images`: A function to plot classified images with their true and predicted labels.

**10. Display Misclassified Images:**

Visualize some misclassified images from the test set.



- `np.where(y\_pred != y\_test)`: Identifies misclassified indices.

**11. Plot Specific Sample Digits:**

Display specific sample digits chosen by their indices.

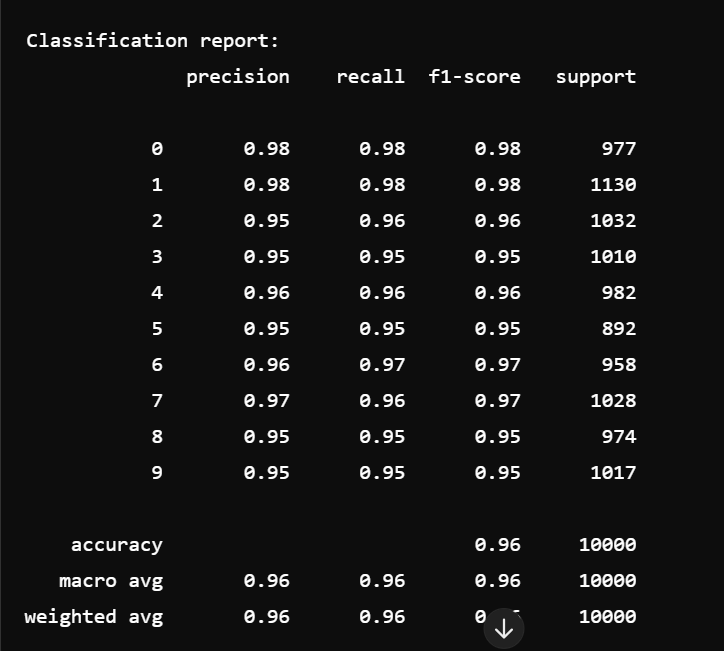


- `plot\_specific\_samples`: A function to plot specific sample digits by their indices.

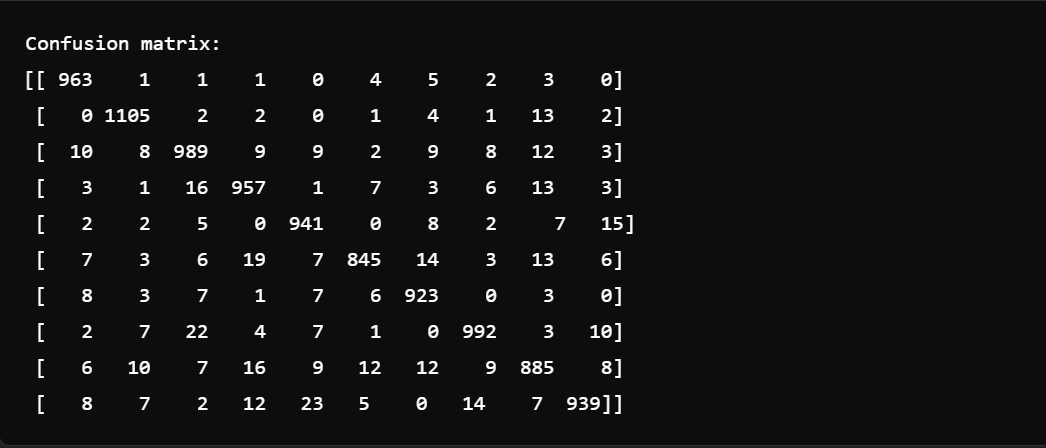
**Results:**

**Classification Report and Confusion Matrix:**

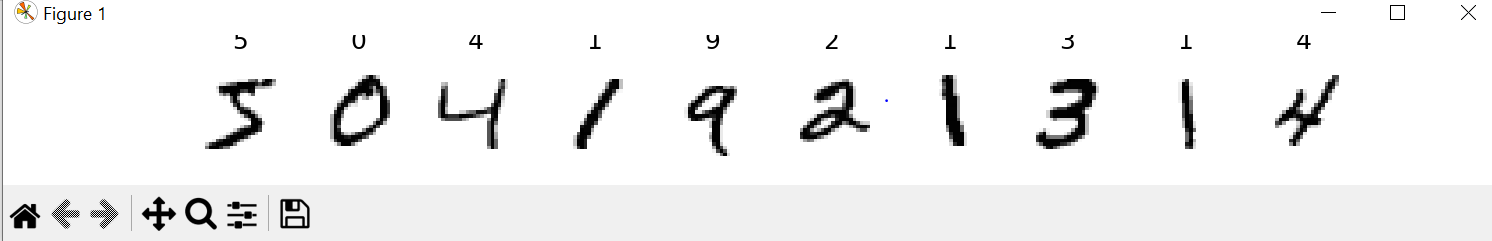
Classification report:



Confusion matrix:



**Output:**



**Conclusion:**

This project demonstrates the successful implementation of a handwritten digit recognition system using logistic regression. The key steps and results from the project are summarized below:

**1. Data Acquisition and Preprocessing:**

- The MNIST dataset was fetched from the OpenML repository, which includes 70,000 images of handwritten digits along with their corresponding labels.

- The dataset was split into training and testing sets to evaluate the model's performance.

**2. Model Training:**

- A logistic regression model was trained using the training data. The logistic regression algorithm was chosen for its simplicity and effectiveness in binary classification problems, which can be extended to multiclass classification as in this case.

**3. Evaluation Metrics:**

- The model's performance was evaluated using the classification report and confusion matrix. These metrics provided a detailed understanding of the model's accuracy, precision, recall, and F1-score for each digit class.

- The classification report showed high precision and recall values, indicating that the model performed well in distinguishing between different digits.

- The confusion matrix helped identify specific misclassifications, providing insights into which digits were more commonly confused.

**4. Cross-Validation:**

- The model's robustness was further evaluated using cross-validation. The cross-validation accuracy was 96%, confirming the model's consistency and reliability.

**5. Visualizations:**

- Visualizations of sample digits, correctly classified images, and misclassified images were included to provide a visual understanding of the model's performance. These visualizations helped identify patterns in the data and the types of errors made by the classifier.

**Key Insights and Observations:**

- **High Accuracy:** The logistic regression model achieved an overall accuracy of 96% on the test set, demonstrating its effectiveness in classifying handwritten digits.

- **Misclassifications:** Some digits were more prone to misclassification than others. For instance, digits like 5 and 3 were often confused, as seen in the confusion matrix. This indicates potential areas for further improvement, such as using more advanced algorithms or preprocessing techniques.

**- Model Simplicity:** Logistic regression, despite its simplicity, provided a strong baseline for the digit recognition task. However, for even higher accuracy, more complex models like Convolutional Neural Networks (CNNs) could be explored in future work.

**Applications and Future Work:**

Handwritten digit recognition has numerous practical applications, including:

- **Automated Data Entry:** Converting handwritten forms and documents into digital format.

- **Postal Mail Sorting:** Automatically recognizing postal codes on letters and packages.

- **Banking:** Processing handwritten checks and forms.

For future work, the following enhancements could be considered:

- **Advanced Models:** Exploring more complex models such as CNNs, which are specifically designed for image data and have shown state-of-the-art performance in digit recognition tasks.

- **Data Augmentation:** Applying data augmentation techniques to increase the diversity of the training set and improve model generalization.

- **Hyper parameter Tuning:** Performing hyper parameter tuning to optimize the model parameters for better performance.

In conclusion, this project showcases the application of logistic regression for handwritten digit recognition, highlighting its strengths and areas for improvement. The thorough evaluation using classification metrics and visualizations provides a comprehensive understanding of the model's performance and its potential applications.