Project Report: MNIST Classification Model

Name: Mudasir Sohail **Course: Data Science**

Website: *https://v0-project-report-website-weld.vercel.app/*

1. Introduction

The objective of this project is to implement a handwritten digit classification model using the MNIST dataset from Kaggle. The dataset consists of grayscale images of handwritten digits (0–9), each of size 28x28 pixels. The model is trained to recognize and classify these digits accurately.

This project demonstrates the use of core data science libraries such as NumPy, Pandas, Matplotlib, and Scikit-learn for loading, processing, visualizing, and classifying the dataset.

2. Dataset Description

The MNIST dataset contains:

- **60,000 training images**
- **10,000 test images**

Each image is stored in a binary file (.ubyte format) and paired with corresponding label files that identify which digit (0–9) the image represents.

The dataset files used:

- train-images.idx3-ubyte
- train-labels.idx1-ubyte
- t10k-images.idx3-ubyte
- t10k-labels.idx1-ubyte

3. Libraries Used

- **NumPy:** For numerical computations and reshaping image arrays.
- **Pandas:** For organizing data if needed.
- **Matplotlib:** For visualizing handwritten digits.
- **Scikit-learn:** For implementing and evaluating the classification model.
- **Struct:** For reading binary dataset files.

4. Data Loading and Preprocessing

Since the dataset is stored in binary format, two custom functions were written to load the data:

Function 1: load_images()
Reads the .idx3-ubyte files containing image data using the struct library.
Each image is reshaped from a 1D array to 28×28 pixels.

Function 2: load_labels()
Reads the .idx1-ubyte files containing labels corresponding to each image.

Example:

```
X_train = load_images('train-images.idx3-ubyte')
y_train = load_labels('train-labels.idx1-ubyte')
X_test = load_images('t10k-images.idx3-ubyte')
y_test = load_labels('t10k-labels.idx1-ubyte')
```

After loading, the pixel values are **normalized** (divided by 255) to scale them between 0 and 1 for better model performance.

5. Data Visualization

To understand the dataset better, several sample images were plotted using Matplotlib:

```
plt.figure(figsize=(5,5))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(X_train[i].reshape(28,28), cmap='gray')
    plt.axis('off')
plt.show()
```

This visualization confirms that each 28×28 image represents a clear handwritten digit.

6. Model Implementation

A **Logistic Regression** model was used for classification. It is a simple yet effective algorithm for multi-class classification tasks.

```
model = LogisticRegression(solver='lbfgs', multi_class='auto', max_iter=1000)
model.fit(X_train, y_train)
```

Here:

- solver='lbfgs' ensures efficient optimization.
- multi_class='auto' allows automatic handling of multiple classes (digits 0–9).
- max_iter=1000 ensures the model has enough iterations to converge.

7. Model Evaluation

After training, predictions were made on the test dataset:

y_pred = model.predict(X_test)

Then, performance metrics were computed:

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

8. Results and Discussion

- The model achieved **high accuracy** on the test data (typically above 90%).
- The **confusion matrix** was also plotted to visualize the prediction results:

```
cm = confusion_matrix(y_test, y_pred)
plt.imshow(cm, cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

The diagonal dominance of the confusion matrix confirms that most predictions were correct, proving the effectiveness of the logistic regression model for this dataset.

9. Conclusion

This project successfully demonstrates the implementation of a **handwritten digit classifier** using the MNIST dataset.
Key takeaways:

- Preprocessing binary image data manually deepens understanding of dataset structure.
- Logistic Regression performs surprisingly well even on image data when flattened and normalized.
- Visualization and evaluation steps are crucial to verify model accuracy and reliability.

10. References

- Kaggle MNIST Dataset: https://www.kaggle.com/datasets/hojjatk/mnist-dataset
- Scikit-learn Documentation: https://scikit-learn.org/
- NumPy Documentation: https://numpy.org/