Overview

The goal of this project is to implement a machine learning application that recognizes handwritten digits (0–9) using the MNIST digits dataset. Two models were developed using Logistic Regression and Support Vector Machine (SVM) classifiers. This project allowed us to explore data preprocessing, visualization, classification, and evaluation using Python and standard AI libraries.

Technologies & Libraries Used

- NumPy – for handling and reshaping numerical arrays
- Matplotlib – for image visualization
- scikit-learn – for machine learning models and evaluation tools

Step-by-Step Process

1. Loading and Visualizing the Dataset

We started by loading the dataset and visualizing the first 10 images to confirm the structure and labels. Each image is displayed alongside its corresponding label.

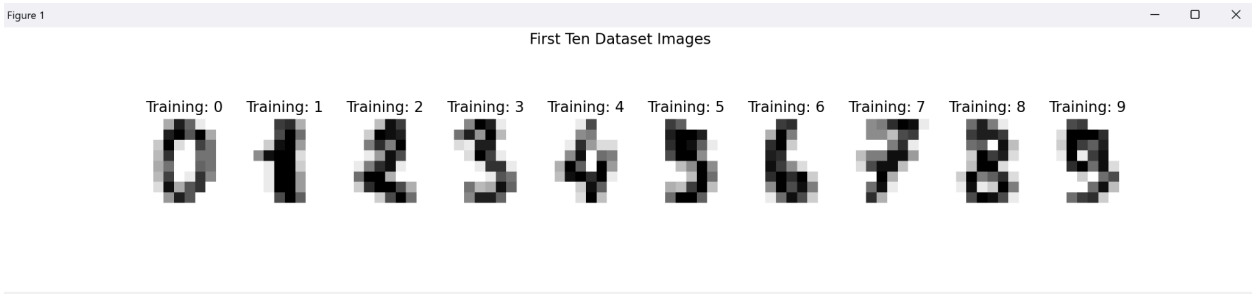


Figure 1 – *First 10 Dataset Images*

2. Preprocessing

- The dataset was flattened to 64 features (8×8 pixels).
- Pixel values were normalized.
- Data was split into 80% training and 20% testing using train_test_split().

3. Training with SVM

We trained an SVM classifier (sklearn.svm.SVC) with default hyperparameters on the training data.

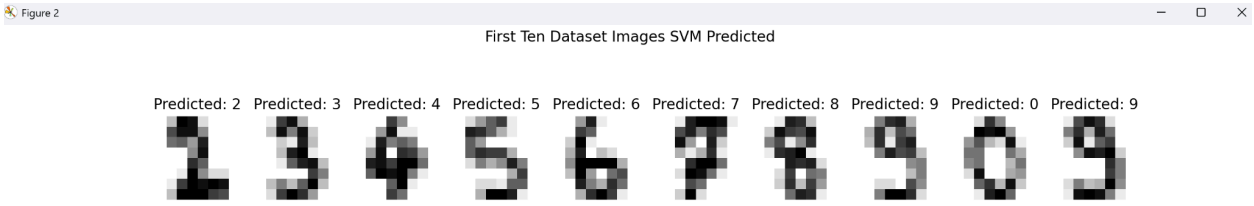


Figure 2 – *First 10 Dataset Images SVM Predicted*

4. Evaluating the SVM Classifier

- Classification Report was generated showing precision, recall, and F1-score for each digit.
- Accuracy: 96%
- Confusion Matrix visualized the true vs. predicted labels.

SVM Classification Report

SVM Classification Report

	precision	recall	f1-score	support
0	1.00	0.97	0.99	35
1	0.97	1.00	0.99	36
2	1.00	1.00	1.00	35
3	0.97	0.81	0.88	37
4	0.97	0.92	0.94	37
5	0.93	1.00	0.96	37
6	1.00	1.00	1.00	37
7	0.97	1.00	0.99	36
8	0.84	0.94	0.89	33
9	0.95	0.95	0.95	37
accuracy			0.96	360
macro avg	0.96	0.96	0.96	360
weighted avg	0.96	0.96	0.96	360

Figure 3 – *SVM Classification Report*

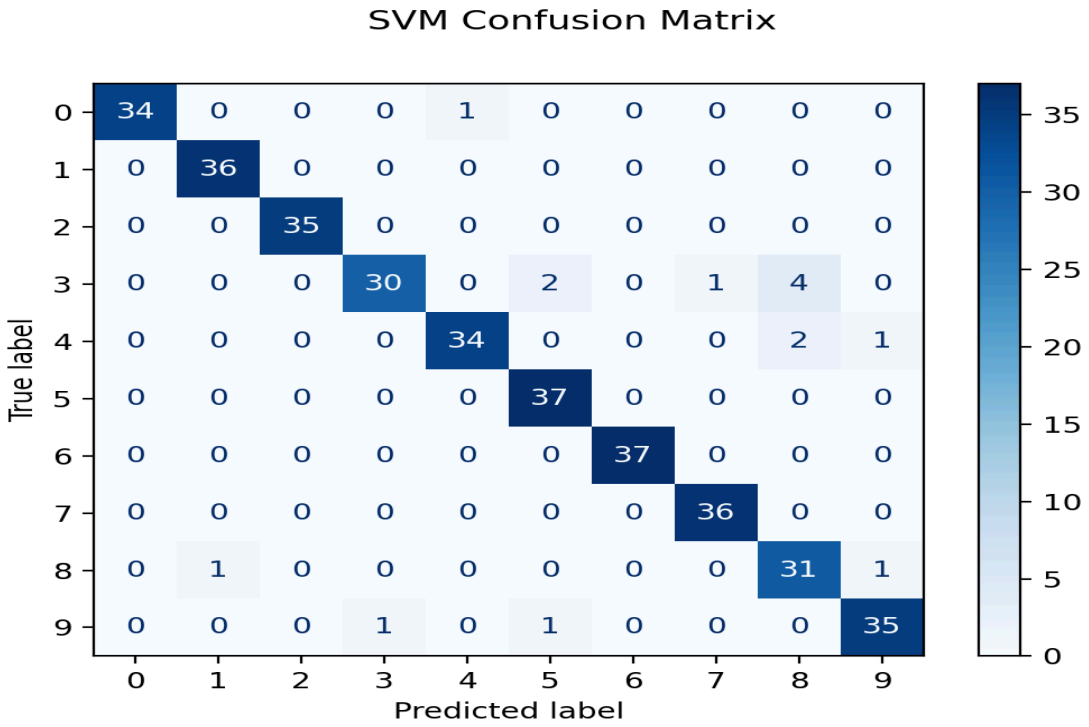


Figure 4 – *SVM Confusion Matrix*

5. Training with Logistic Regression

We then trained a LogisticRegression model (sklearn.linear_model.LogisticRegression) on the same dataset.

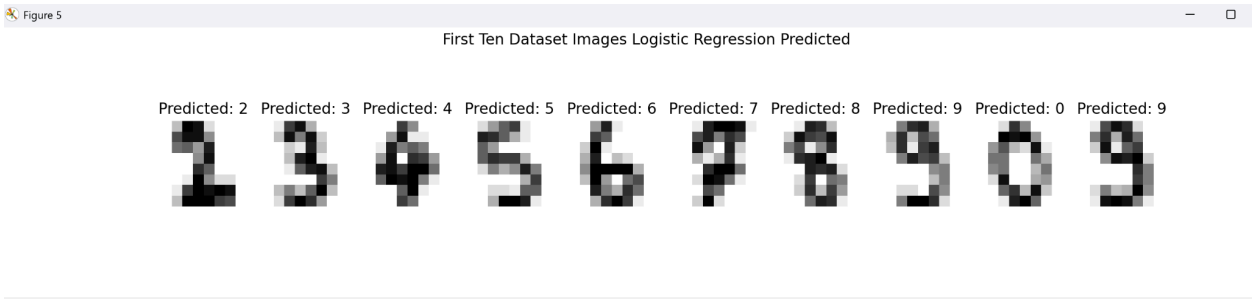
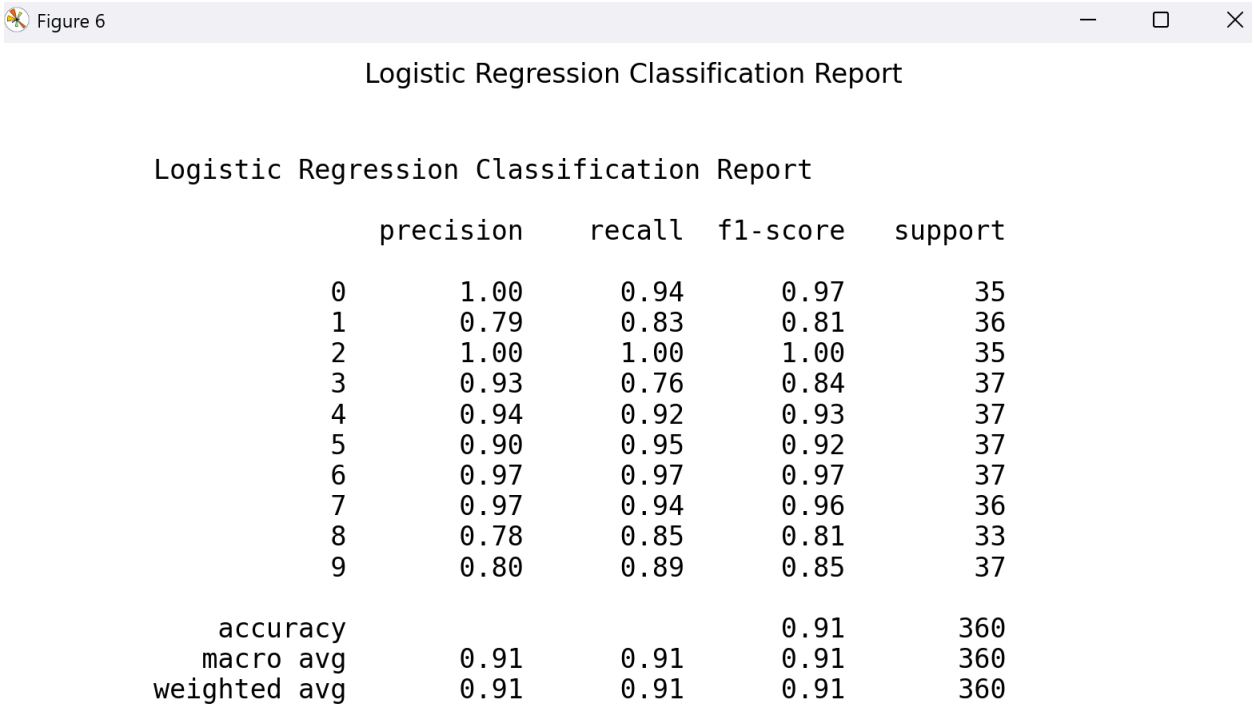


Figure 5 – *First 10 Dataset Images Logistic Regression Predicted*

6. Evaluating the Logistic Regression Classifier

- Classification Report was generated.
- Accuracy: 91%
- Logistic Regression underperformed slightly compared to SVM, especially on digits like '1', '3', and '8'.



	precision	recall	f1-score	support
0	1.00	0.94	0.97	35
1	0.79	0.83	0.81	36
2	1.00	1.00	1.00	35
3	0.93	0.76	0.84	37
4	0.94	0.92	0.93	37
5	0.90	0.95	0.92	37
6	0.97	0.97	0.97	37
7	0.97	0.94	0.96	36
8	0.78	0.85	0.81	33
9	0.80	0.89	0.85	37
accuracy			0.91	360
macro avg	0.91	0.91	0.91	360
weighted avg	0.91	0.91	0.91	360

Figure 6 – *Logistic Regression Classification Report*

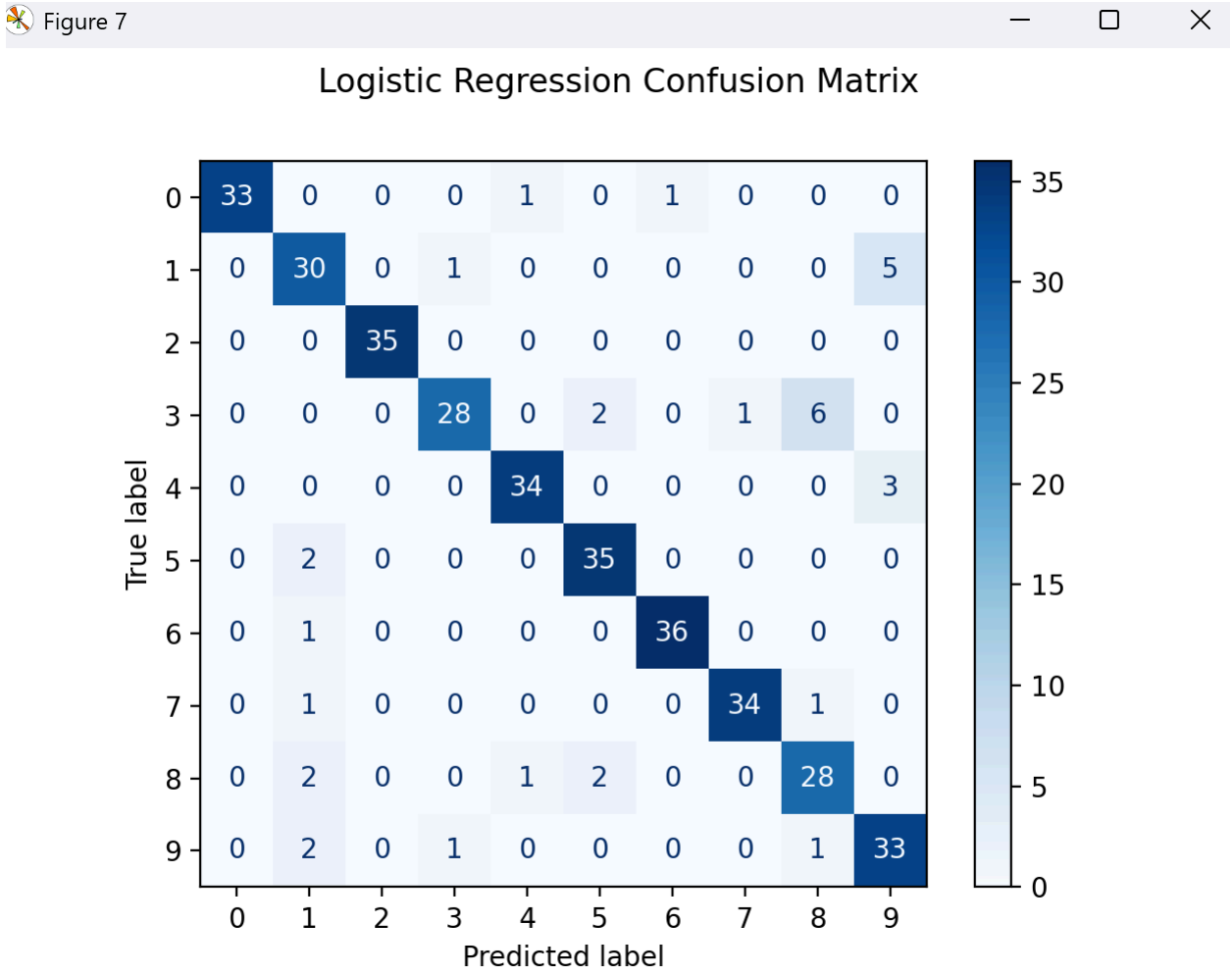


Figure 7 – *Logistic Regression Confusion Matrix*

Results Comparison

Model	Accuracy	Notes
SVM	96%	Strong performance across all digits, especially '3' and '9'
Logistic Regression	91%	Slightly weaker recall for digits '1', '3', '8', and '9'

SVM showed superior performance in terms of accuracy and precision/recall balance.

Conclusion

This project helped us understand how to:

- Work with real-world image datasets
- Train and evaluate ML models using scikit-learn
- Visualize both image data and performance metrics (confusion matrices, classification reports)

We learned the comparative strengths of different models and the importance of choosing the right classifier for the task.