Logistic Regression

June 24, 2025

Problem 1 1. Methodology Description

Problem

Use Logistic Regression on a Loan Default Prediction Dataset using Gradient Descent methods to identify high-risk individuals, enabling timely interventions in financial loan services.

Data Splitting

The data was split using the following strategies:

- Train-Test Splits: {85:15, 60:40, 50:50, 30:70}
- Random and Sequential Splits were applied for evaluation purposes.

Gradient Descent Variants

Implemented and tested the following variants of Gradient Descent:

- SGD (Stochastic Gradient Descent)
- GD (Gradient Descent)
- BGD (Batch Gradient Descent)

The update rules were implemented manually without using built-in optimization functions. Hyperparameters like learning rate and epochs were varied to analyze their impact on performance.

Built-in Algorithm

Additionally, the inbuilt Logistic Regression function from a machine learning library was employed for comparison.

2. Performance Metrics

The following performance metrics were used to evaluate the models:

- Accuracy
- Precision
- Recall
- F1 Score
- Confusion Matrix

Table 1: Confusion Matrix

		Predicted	
		Positive	Negative
2***Actual**	Positive	TP	FN
	Negative	FP	TN

3. Result Compilation

The summarized results for all models under different conditions are shown below:

Table 2: Model Performance Comparison

Train Size	Test Size	Split Type	GD Variant	Epochs	Learning Rate	Accuracy	P
0.15	0.85	Sequential	SGD	500	0.01	0.8710	
0.15	0.85	Random	SGD	500	0.01	0.8589	
0.15	0.85	Sequential	GD	100	0.5	0.8670	
0.15	0.85	Random	GD	100	0.5	0.8669	

... (Other rows omitted for brevity)

4. Visualization

Plots were generated for:

- Loss function over epochs for different Gradient Descent methods.
- Performance metrics comparison across splits.
- Effect of varying learning rates.

5. Analysis of Results

- Mini-Batch GD showed inconsistent behavior in some splits, often underperforming with low recall values.
- SGD exhibited faster convergence but with high variability due to its stochastic nature.
- GD provided stable results with moderate performance but required a higher number of epochs.
- The inbuilt Logistic Regression achieved the highest overall accuracy but struggled with recall due to class imbalance.

6. Conclusion and Insights

- Inbuilt Logistic Regression outperformed custom implementations in terms of accuracy.
- Class imbalance significantly impacted recall, indicating a need for oversampling techniques.
- SGD was the fastest but suffered from performance variability.
- GD was stable but computationally expensive.
- MBGD was less effective due to improper batch size tuning, requiring further experimentation.