Logistic Regression from Scratch on Titanic Dataset

Vaibhav Chourasia

October 2, 2025

1 Introduction

This report describes the implementation of Logistic Regression from scratch in Python, without using scikit-learn. The Titanic dataset is used to predict passenger survival. The implementation includes feature preprocessing, logistic regression training using gradient descent, visualization of the sigmoid function, and evaluation with and without regularization.

2 Libraries Used

The following Python libraries were used:

- **NumPy**: For numerical computations, array manipulation, and matrix operations.
- **Pandas**: For loading the dataset and handling tabular data.
- **Matplotlib**: For plotting graphs such as the sigmoid function, training loss, and logistic curves.

3 Dataset and Preprocessing

The dataset was obtained from `https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv`. The selected columns are:

Survived, Pclass, Sex, Age, SibSp, Parch, Fare.

Preprocessing steps:

1. Missing values in `Age` and `Fare` were filled with their median values.
2. The categorical feature `Sex` was mapped as: male = 1, female = 0.
3. A bias column (all ones) was added to the features.
4. Features were standardized:
$$x' = \frac{x - \mu}{\sigma}$$
where μ is the mean and σ is the standard deviation of each feature.
5. The dataset was split into training (80%) and testing (20%).

4 Logistic Regression Model

Logistic regression models the probability of survival as:

$$p(y = 1|x) = \sigma(z), \quad z = w^\top x$$

where w is the weight vector and $\sigma(z)$ is the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

4.1 Loss Function

The cross-entropy loss is:

$$L(w) = -\frac{1}{n} \sum_{i=1}^{n} \Big[y^{(i)} \log p^{(i)} + (1 - y^{(i)}) \log(1 - p^{(i)}) \Big],$$

where n is the number of samples and $p^{(i)} = \sigma(w^\top x^{(i)})$.

4.2 Gradient

The gradient of the loss with respect to weights is:

$$\nabla L(w) = \frac{1}{n} X^\top (p - y),$$

where X is the feature matrix, p the predicted probabilities, and y the true labels.

Derivation:

For a single sample, the loss is:

$$\ell^{(i)} = - \left[y^{(i)} \log p^{(i)} + (1 - y^{(i)}) \log(1 - p^{(i)}) \right], \quad p^{(i)} = \sigma(w^\top x^{(i)})$$

Differentiating with respect to w:

$$\frac{\partial \ell^{(i)}}{\partial w} = (p^{(i)} - y^{(i)}) x^{(i)}$$

Thus, the gradient over the entire dataset is:

$$\nabla L(w) = \frac{1}{n} \sum_{i=1}^{n} (p^{(i)} - y^{(i)}) x^{(i)} = \frac{1}{n} X^\top (p - y)$$

4.3 Regularization

To avoid overfitting, L2 regularization is added:

$$L_{\text{reg}}(w) = L(w) + \frac{\lambda}{2n} \sum_{j=1}^{d} w_j^2,$$

where λ is the regularization parameter and the bias weight w_0 is not penalized.

Derivation:

Adding L2 regularization penalizes large weights:

$$L_{\text{reg}}(w) = L(w) + \frac{\lambda}{2n} \sum_{j=1}^{d} w_j^2$$

Gradient for $j \geq 1$:

$$\frac{\partial L_{\text{reg}}}{\partial w_j} = \frac{1}{n} \sum_{i=1}^{n} (p^{(i)} - y^{(i)}) x_j^{(i)} + \frac{\lambda}{n} w_j$$

The bias w_0 is not regularized:

$$\frac{\partial L_{\text{reg}}}{\partial w_0} = \frac{1}{n} \sum_{i=1}^{n} (p^{(i)} - y^{(i)})$$

The gradient with regularization becomes:

$$\nabla L_{\text{reg}}(w)_j = \begin{cases} \frac{1}{n} X^\top (p - y), & j = 0 \\ \frac{1}{n} X^\top (p - y) + \frac{\lambda}{n} w_j, & j \geq 1 \end{cases}$$

5 Training Procedure

The model is trained using batch gradient descent:

$$w := w - \eta \cdot \nabla L(w),$$

where η is the learning rate.

Training was performed for 3000 iterations with a learning rate $\eta = 0.05$.

6 Results and Graphs

6.1 Sigmoid Function

Figure 1 shows the sigmoid function, mapping real numbers into probabilities between 0 and 1.

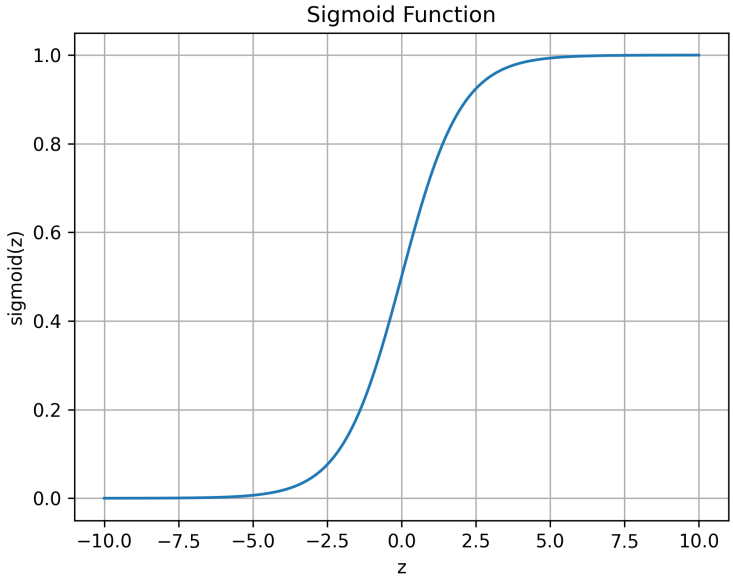

Figure 1: Sigmoid Function

6.2 Training Loss

The training loss decreases with iterations, as shown in Figure 2.

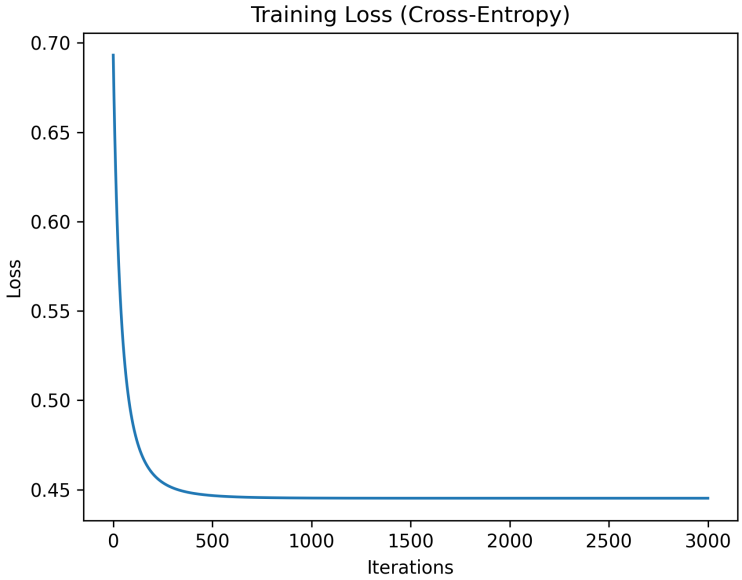

Figure 2: Training Loss (Cross-Entropy)

6.3 Logistic Curve for a Feature

Figure 3 shows the logistic curve plotted against a single feature (e.g., Age after scaling), with survival data points.

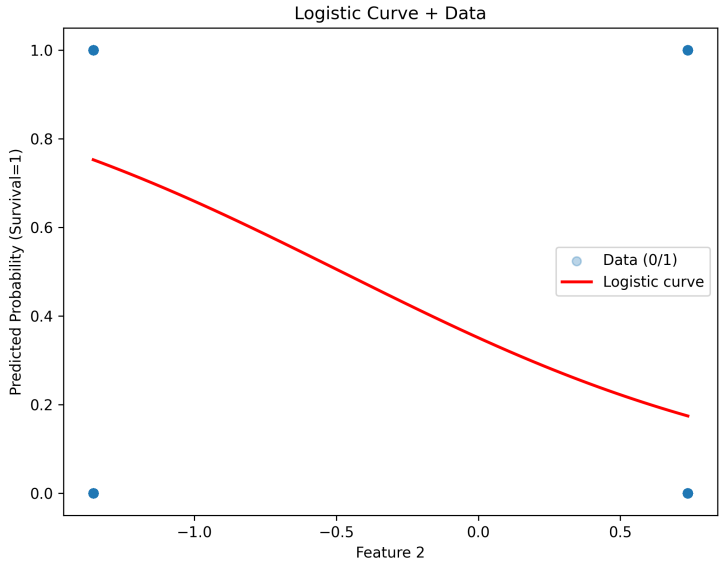

Figure 3: Logistic Curve with Data

6.4 Log-Odds vs Feature

Figure 4 shows the log-odds plotted against a feature, demonstrating the linear relationship.

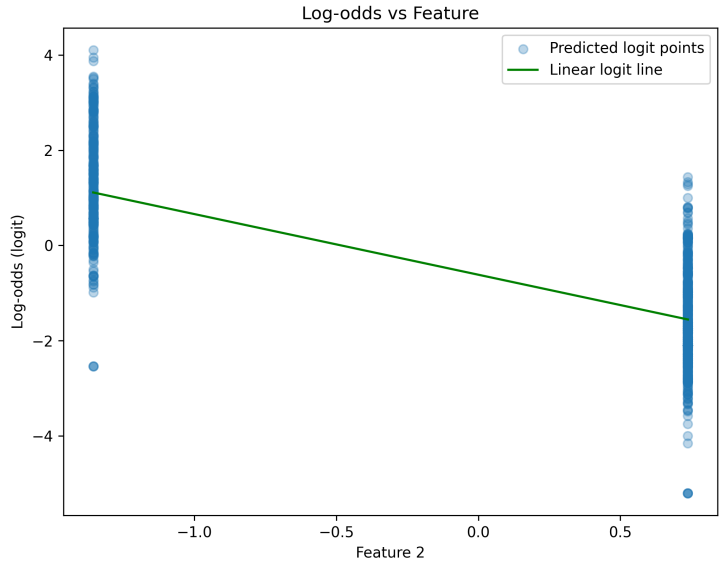

Figure 4: Log-Odds vs Feature

6.5 Regularized Training Loss

Figure 5 shows the effect of L2 regularization on the training loss.

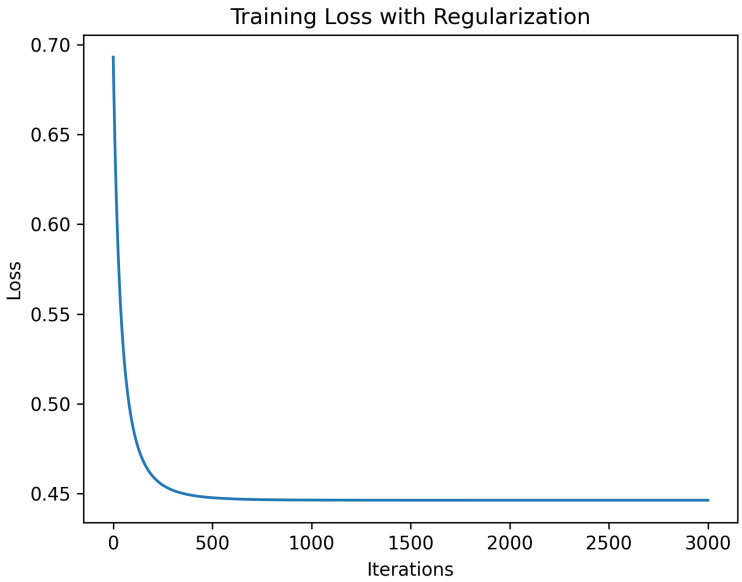

Figure 5: Training Loss with Regularization

7 Model Evaluation

Predictions were made on the test set using a threshold of 0.5:

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(w^\top x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The test accuracy obtained without regularization was approximately:

$$\text{Accuracy} = 0.78$$

(Actual accuracy may vary slightly depending on random seed).

With L2 regularization, the accuracy was:

$$\text{Accuracy}_{\text{reg}} = 0.79$$

8 Conclusion

This project demonstrates logistic regression implemented from scratch. The following were achieved:

- Implemented sigmoid, cross-entropy loss, gradient descent, and L2 regularization without external ML libraries.
- Preprocessed the Titanic dataset for binary classification.
- Visualized the sigmoid function, training loss, logistic curve, and log-odds.
- Achieved test accuracy of around 78–79%.

This shows that logistic regression, even without libraries, is effective for binary classification tasks such as survival prediction.