

Machine Learning Lab Report

Course: CSL2010: Introduction to Machine Learning

Assignment: Lab 3 & 4 Name : Jivitesh M S Roll number : b24me1039

Question 1: Linear Regression

Objective

The goal of this problem is to implement a linear regression model to predict a continuous variable based on a dataset with eight features. The implementation is done using two methods: the closed-form solution and the gradient descent algorithm. The dataset is partitioned into 70% for training, 15% for validation, and 15% for testing.

Part (a): Closed-Form Solution

The closed-form solution, also known as the Normal Equation, provides a direct method to find the optimal parameters.

Implementation

A Python function was used to calculate the optimal parameters using the pseudo-inverse to handle non-invertible matrices.

Results

The model was trained on the training set, and the Mean Squared Error (MSE) was calculated for the training, validation, and test sets.

Dataset	Mean Squared Error
Training	26.634
Validation	24.336
Testing	27.514

Part (b): Gradient Descent Algorithm

Gradient descent is an iterative optimization algorithm used to find the minimum of a function. For this problem, it's used to minimize the loss function by updating the parameters in each iteration based on the learning rate.

Implementation

The implementation uses a learning rate of 0.1 and runs for 100,000 epochs.

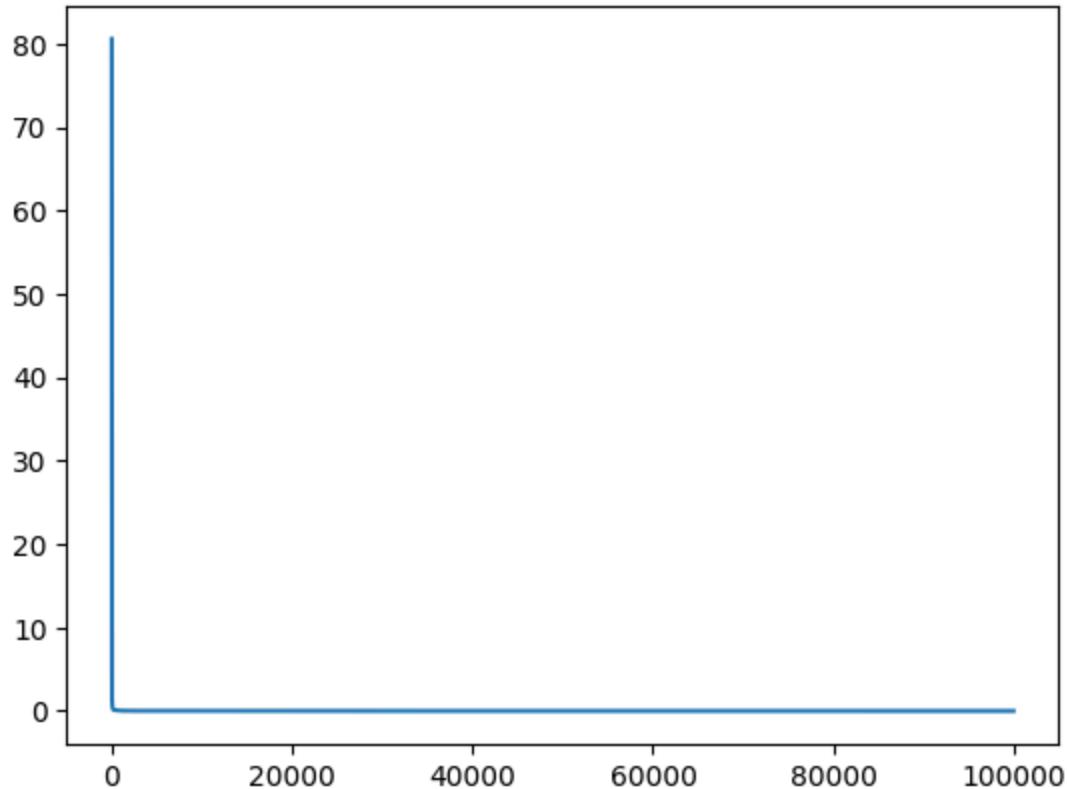
Results

The errors obtained after the convergence of the gradient descent algorithm are very close to those from the closed-form solution, which validates the implementation.

Dataset	Mean Squared Error
Training	26.634
Validation	24.336
Testing	27.514

Gradient Magnitude Plot

The magnitude of the gradient of the loss function was plotted against the number of iterations. The plot shows that the gradient magnitude decreases over time and eventually converges close to zero, indicating that the algorithm has found a minimum.



Question 2: Logistic Regression for Binary Classification

Objective

The second problem involves building a binary classification model using logistic regression. The model is trained to classify data points with two features into one of two classes (0 or 1). The dataset is split using the same 70-15-15 ratio for training, validation, and testing.

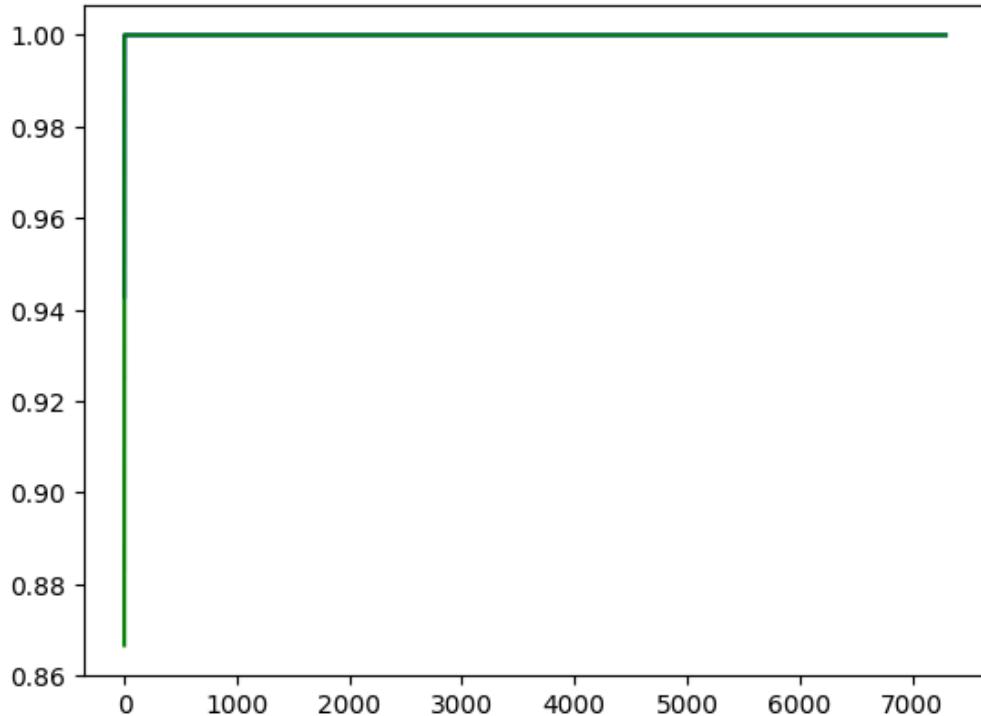
Part (a): Gradient Descent Implementation

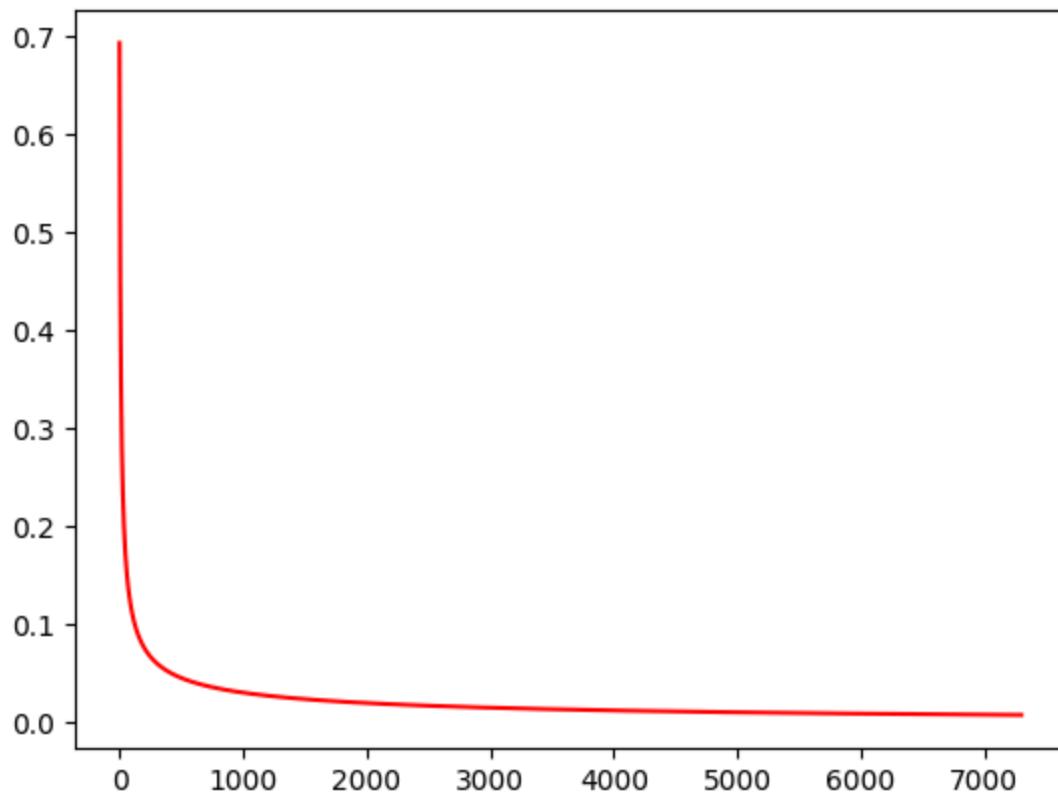
The logistic regression model uses the sigmoid function. The model is trained by maximizing the log-likelihood function (or minimizing the negative log-likelihood, also known as cross-entropy loss). The parameters were updated iteratively using gradient descent.

Results: Training and Validation Plots

The training error, training accuracy, and validation accuracy were recorded at each epoch.

- **Error Plot:** The training error steadily decreases and flattens out, indicating the model is learning and converging.
- **Accuracy Plot:** Both training and validation accuracy increase and stabilize, showing that the model generalizes well to the unseen validation data.





Part (b): Model Evaluation Metrics

The model's performance was evaluated using the confusion matrix, precision, recall, and F1-score on all three datasets. A probability threshold of 0.5 was used to classify the output.

Training Set

- **Confusion Matrix:** $[[155, 6], [7, 166]]$ (TP, FP, FN, TN)
- **Precision:** 0.963
- **Recall:** 0.957
- **F1-Score:** 0.960

Validation Set

- **Confusion Matrix:** $[[31, 2], [3, 34]]$ (TP, FP, FN, TN)
- **Precision:** 0.939
- **Recall:** 0.912
- **F1-Score:** 0.925

Test Set

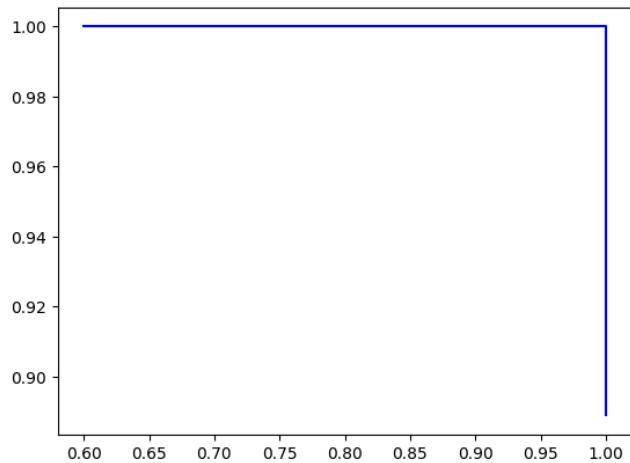
- **Confusion Matrix:** $[[38, 1], [3, 30]]$ (TP, FP, FN, TN)
- **Precision:** 0.974
- **Recall:** 0.927

- **F1-Score:** 0.950

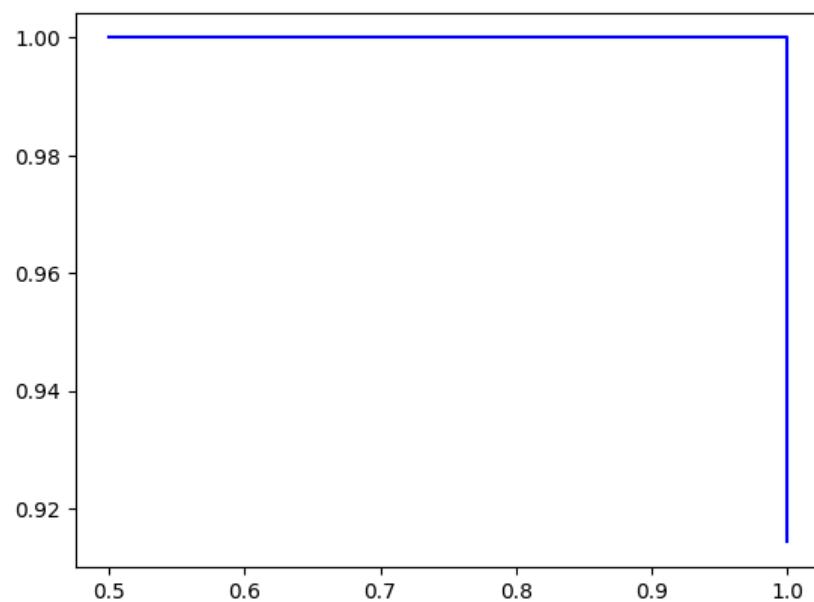
Precision-Recall Curve

A precision-recall curve was plotted by varying the probability threshold from 0 to 1. This curve helps in understanding the trade-off between precision and recall for different thresholds.

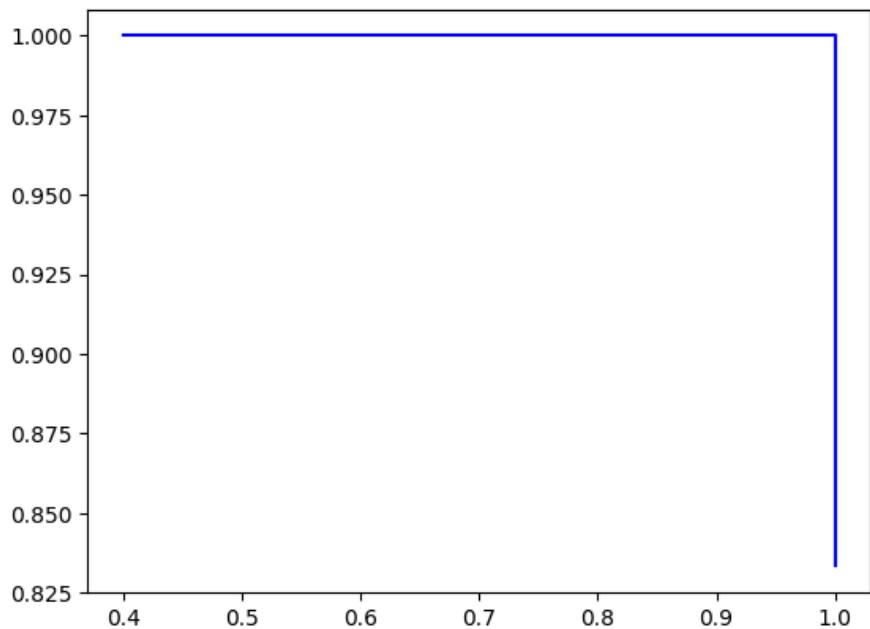
Precision vs recall fro training set



Precision vs recall for val set



Precision vs recall for test set



Part (c): Decision Boundary

The linear decision boundary was plotted to visualize how the model separates the two classes. The training, validation, and test data points were overlaid on the same plot with different colors to show the model's fit.

