Multiple Variable Linear Regression

Prerna Tripathi

May 19, 2025

1. Initial Procedures

1.1. Approach

The common process across all the three parts of the assignment is to preprocess the data by removing all the missing values in the dataset which can lead to improper training of the model and also convert the feature 'ocean proximity'into a form the algorithm can understand and process. The dataset is then divided into a 80:20 ratio into training and validation sets. The features are then also scaled using Z-score normalisation across all the three parts. Feature scaling is highly necessary for implementing gradient descent. Few things as in the formation of the feature 'ocean proximity' was done with the help of gpt.

2. Pure Python Implementation

The pure python implementation implements multiple variable linear regression using gradient descent without using any libraries,using only loops and lists.The maximum no. of iterations is 500 and the learning rate is 0.01. The graph of convergence vs iteration shows that our gradient descent has converged well enough to a point of global minimum Use of AI was done in areas using scikit-library to calculate regression metrics and in splitting the data.Otheriwse the implementation of the main gradient descent algorithm was done avoiding the use of AI.

2.1. Analysis

Convergence time: 356.3851 seconds MAE= Training:50623.3644, Validation:50519.6950 RMSE= Training:69824.8406, Validation:69375.6927 R2-score= Training:69824.8406, Validation:0.6405

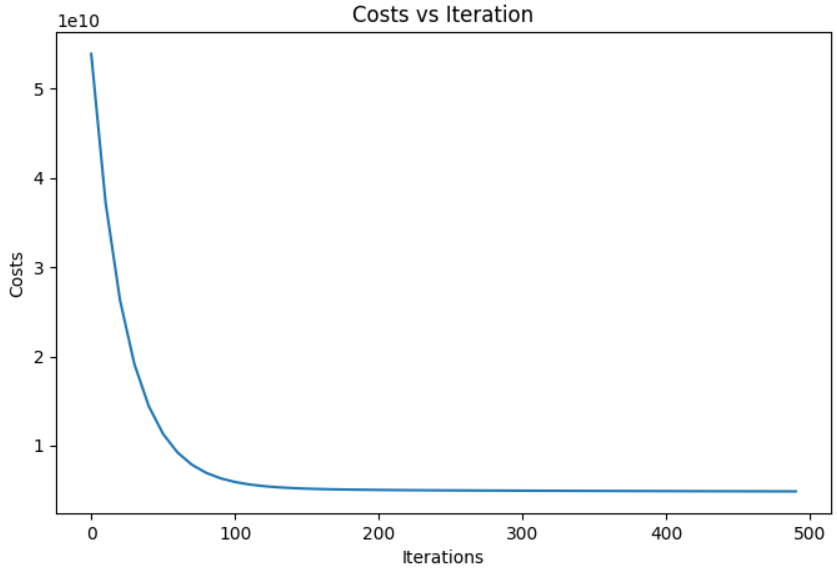

Figure 1: Graph of Convergence vs Iteration:Pure Python

3. Numpy Implementation

In this approach, the algorithm remains the same as the first part including the initial parameters, but instead of loops and lists Numpy library is used which through vectorized algebra and parallel operations improves the execution speed and shortens the code also. Here also the learning rate is 0.01 and the no. of iterations is 500and the graph of convergence vs iterations shows that our gradient decsent has converged well enough to a minimum. Use of AI was again restricted to the use of scikit-library functions and also to a few numpy operations which have been used in the code.

3.1. Analysis

Convergence time: 0.1059 seconds MAE= Training:50480.7466,,Validation:50390.6575 RMSE= Training:69745.5760, Validation:69279.9713 R2-score= Training:69279.9713, Validation:0.6415

4. Scikit-learn Implementation

4.1. Approach

The scikit-learn implementation uses the `LinearRegression` class, which abstracts away the algorithmic details. It provides built-in methods for model fitting, prediction, and perfor-

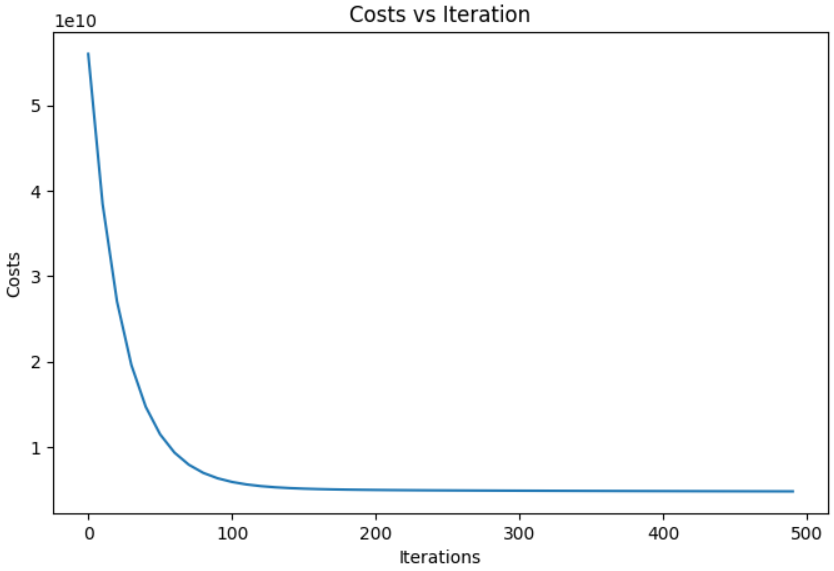

Figure 2: Graph of Convergence vs Iteration: Numpy Optimised

mance evaluation. Again the use of AI has been done to incorporate the scikit's operations.

4.2. Analysis

Fitting Duration:0.0132 seconds MAE= Training:49752.9160, Validation:49964.0128 RMSE= Training:68758.3663, Validation:68201.5212 R2-score= Training: 0.6448, Validation=0.6526

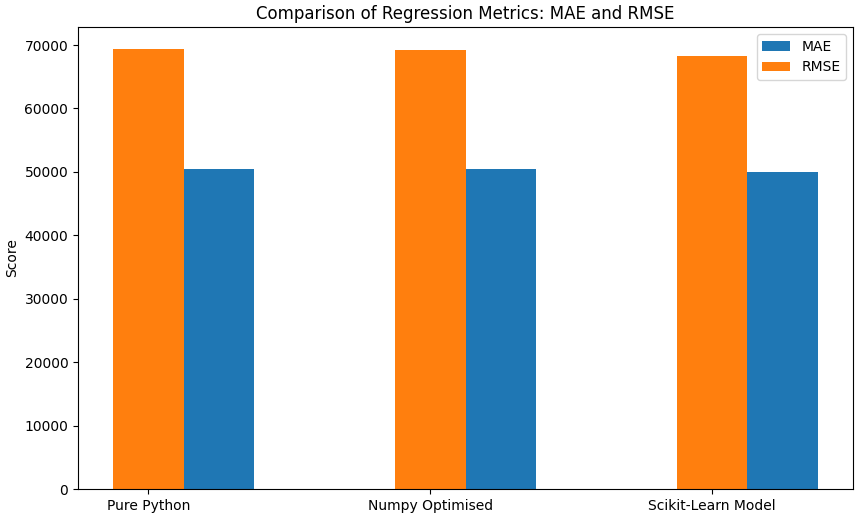

Figure 3: Comparsion Of MAE and RMSE over the three parts

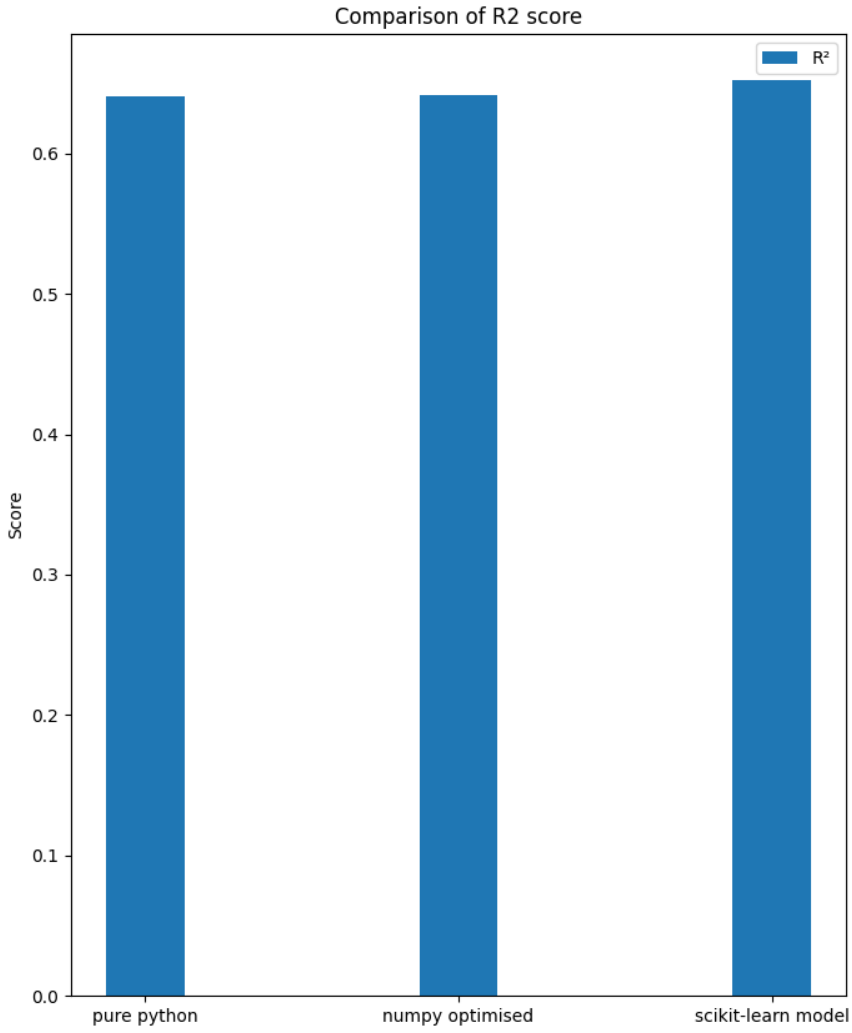

Figure 4: Comparsion Of R2 score over the three parts