Model Summary Report

Vidit Kumar

May 2025

1 Introduction

The task is to develop three machine learning models that utilize multivariate linear regression:

- Model 1: Implemented purely in **Python**
- Model 2: Implemented using **NumPy**
- Model 3: Implemented using **Scikit-learn**

The dataset used is the **California Housing Price** dataset. Initially, I performed data analysis and visualization to better understand the data.

1.1 Data Visualization

Certain categories in **ocean proximity** did not follow a Gaussian distribution and had weak correlation with the target variable. The following plot shows a Q-Q plot after removing data points where `ocean proximity = ISLAND`.

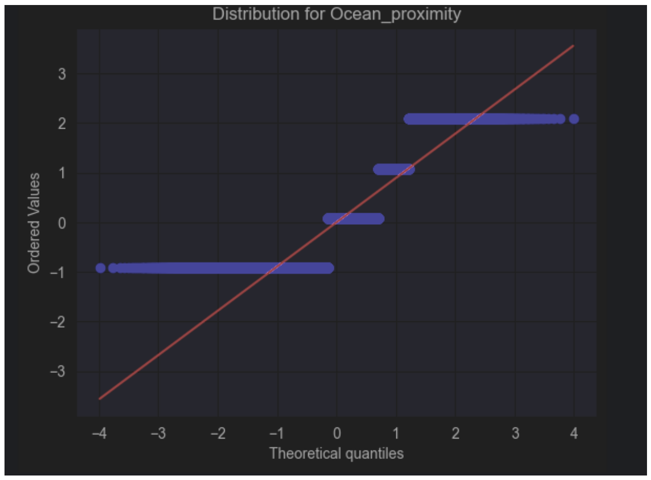

Figure 1: Q-Q plot for ocean proximity after removing 5 datapoints with $ocean\_proximity = ISLAND$

Afterward, I loaded the dataset from Scikit-learn. An initial bar graph of the distribution was not informative. However, a scatter plot between geographical location and median house value did reveal a relationship between location and housing prices.

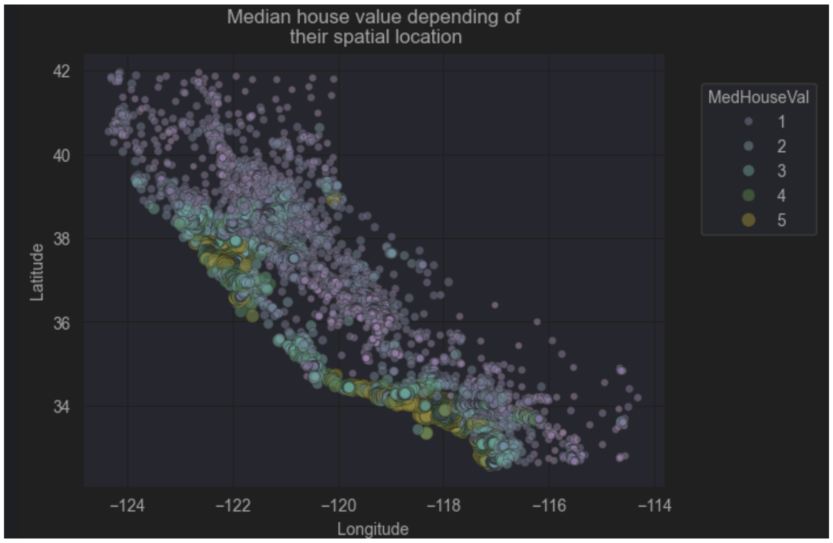

Figure 2: Scatter plot showing relation between location and median house value

Note: Missing values were filled using the median due to the presence of outliers.

2 Model Overview

Model 1: Pure Python implementation inspired by Andrew Ng's approach.

Model 2: NumPy-based implementation using the same logic as Model 1.

Model 3: A simple application of Scikit-learn's `LinearRegression`.

3 Model 1: Pure Python Implementation

This model was developed entirely using Python. It required a significant amount of time to produce results. Training for 1000 epochs took **15 minutes and 42 seconds**, resulting in an **R2 score of** -0.0273.

3.1 Advantages

- Educational and helps understand the internal workings
- Easy to modify for experimenting with optimizers (e.g., Momentum, RMSProp)

3.2 Disadvantages

- Very slow training due to lack of parallelization and GPU support
- Inefficient numerical computations compared to optimized libraries

4 Model 2: NumPy Implementation

This model uses NumPy to speed up computations through vectorized operations.

4.1 Advantages

- Faster computation using NumPy's parallel capabilities
- Multi-threading support reduces CPU load

4.2 Disadvantages

- Less optimized compared to frameworks like PyTorch or TensorFlow
- Higher memory usage; caused IDE crashes
- Does not leverage GPU; frameworks like CuPy are better suited

5 Model 3: Scikit-learn Implementation

Implemented using Scikit-learn's `LinearRegression`.

5.1 Advantages

- Extremely fast training
- Requires minimal code and is user-friendly

5.2 Disadvantages

- Lack of transparency regarding inner workings

6 Comparisons

This section discusses training time and convergence behavior for each model.

6.1 Convergence Times

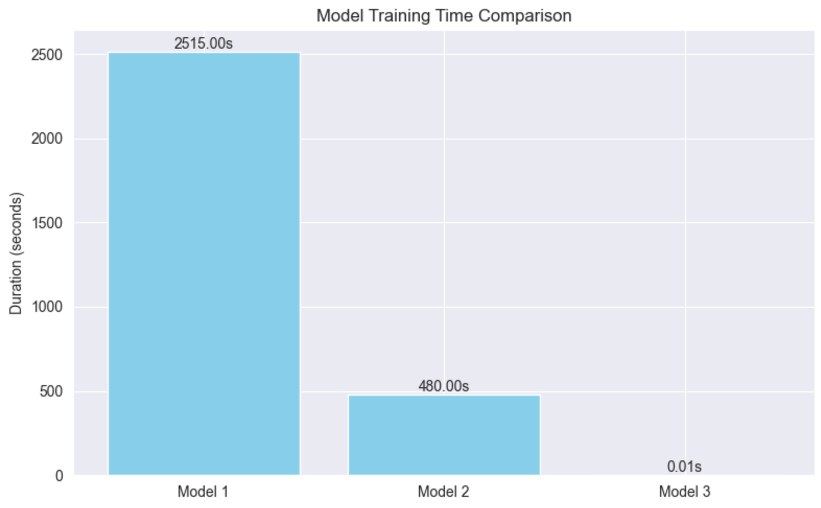

Figure 3: Training times for Model 1 (30000 epochs) and Model 2 (3000000 epochs)

7 Loss Convergence Curves

7.1 Model 1

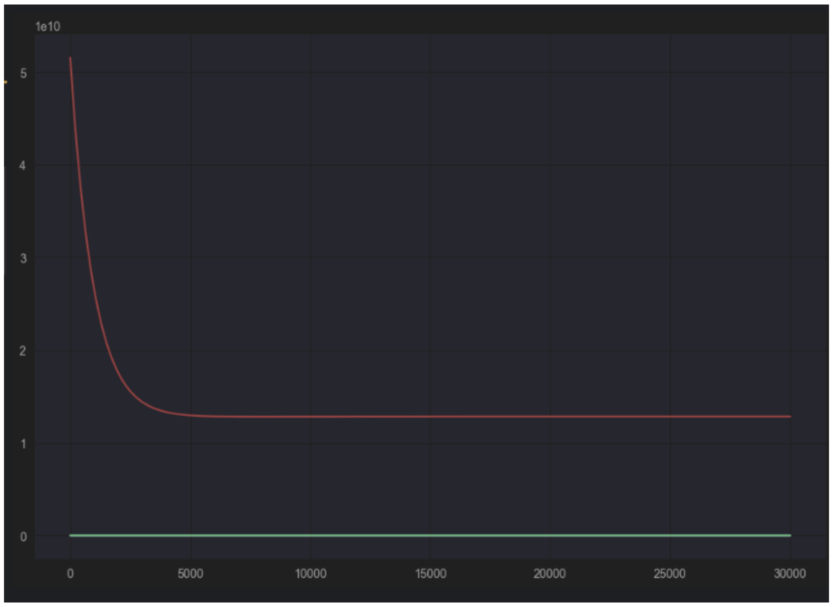

Figure 4: Loss curve for Model 1: 30000 epochs, learning rate = 0.001

7.2 Model 2

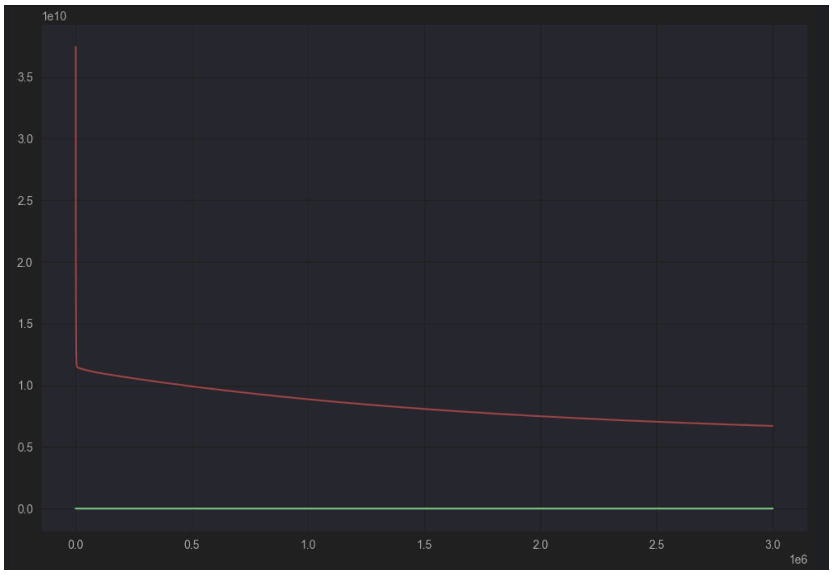

Figure 5: Loss curve for Model 2: 3000000 epochs, learning rate = 0.001

8 R2 Scores

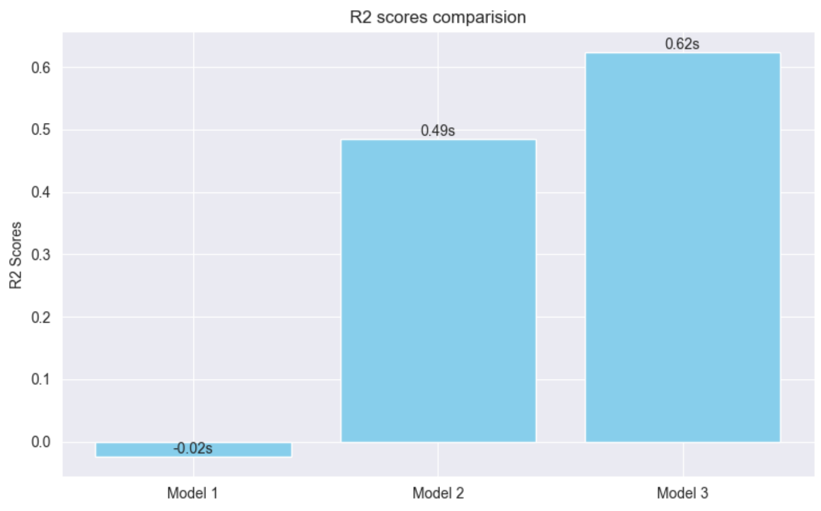

Figure 6: Comparison of R2 scores for all models

9 Conclusion

Missing values were imputed using the median due to many outliers. Then, RobustScaler was used to mitigate the effect of outliers and scale features appropriately.
Here the initializing parameters also play an important role as if there value is nearer the convergence value model will take less time to train.
To make a model more scalable we have to sacrifice efficiency.
Model 1, though insightful, had the longest training time. Due to personal constraints, it could not be trained extensively. Model 2 achieved convergence thanks to faster computations with NumPy. Model 3, being pre-built, trained almost instantly.

10 Assistance and Tools

- Plotting and visualization support provided by ChatGPT for syntax assistance.
- Model logic and implementation were done by me.