CSoC IG Assignment Report

Comparative Study of Multivariable Linear Regression Implementations

Yash Bankar
Roll Number: 24034017

May 2025

Introduction

This report presents three implementations of multivariable linear regression on the Housing Dataset:

- **Part 1:** Pure Python implementation using gradient descent.
- **Part 2:** Optimized NumPy vectorized implementation.
- **Part 3:** scikit-learn's `LinearRegression`.

We compare convergence speed, predictive accuracy (MAE, RMSE, R^2), and training time.

Dataset and Preprocessing

The dataset includes both numerical and categorical features. Categorical features were one-hot encoded and binary fields were mapped to 0/1. All features were normalized using Z-score normalization. An 80/20 train-validation split was applied.

1 Part 1: Pure Python Implementation

Implementation Details

- Hypothesis: $\hat{y} = w^\top x + b$
- Loss: $J = \frac{1}{2m} \sum (\hat{y} - y)^2$
- Optimization: Gradient descent with learning rate $\alpha = 0.01$, 1000 epochs.

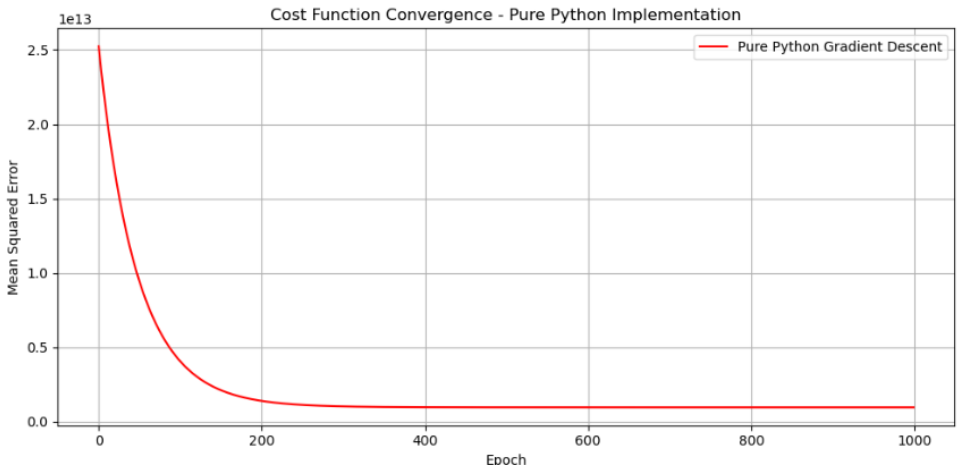

Figure 1: Convergence plot for pure Python implementation.

Convergence Plot

Results

- Training time: 3.5545 seconds
- R^2: 0.6531
- MAE: 969206.24
- RMSE: 1324170.42

2 Part 2: Optimized NumPy Implementation

Implementation Details

Same gradient descent logic as Part 1, but using NumPy vectorized operations.

Convergence Plot

Results

- Training time: 0.0941 seconds
- R^2: 0.6531
- MAE: 969206.24
- RMSE: 1324170.42

3 Part 3: Scikit-learn Implementation

Implementation Details

Used `sklearn.linear_model.LinearRegression()` trained on the same data.

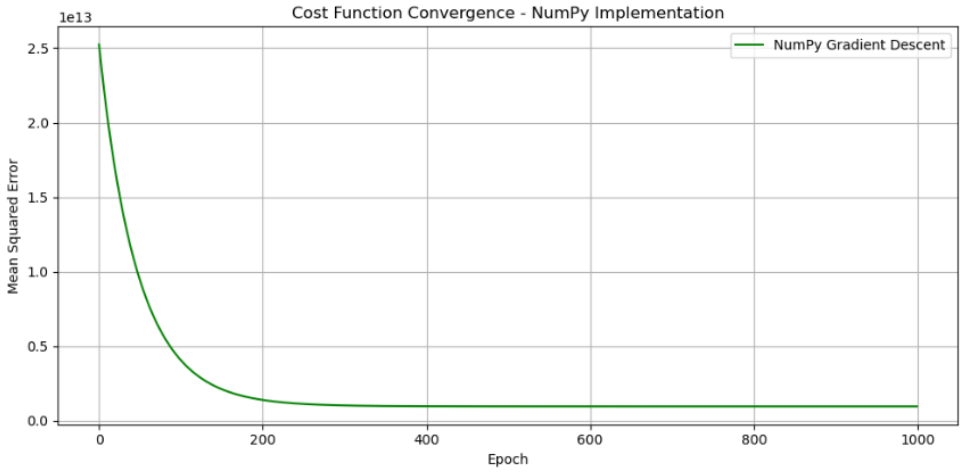

Figure 2: Convergence plot for NumPy implementation.

Results

- Training time: 0.002 seconds
- R^2: 0.6529
- MAE: 970043.40
- RMSE: 1324506.96

Comparison of Convergence and Metrics

Convergence Time vs Final Cost

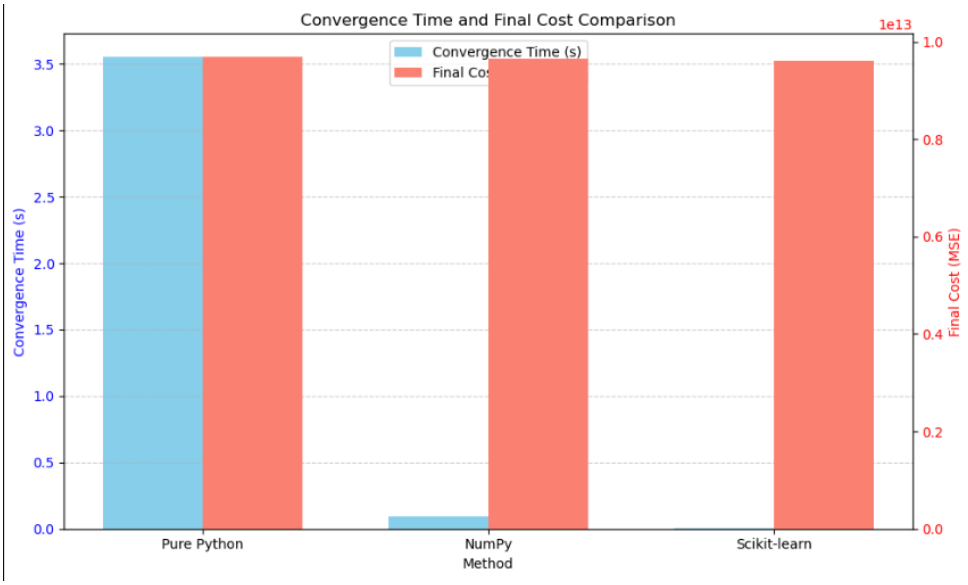

Figure 3: Comparison of convergence speeds and final costs across all implementations.

Regression Metrics Across All Three Methods

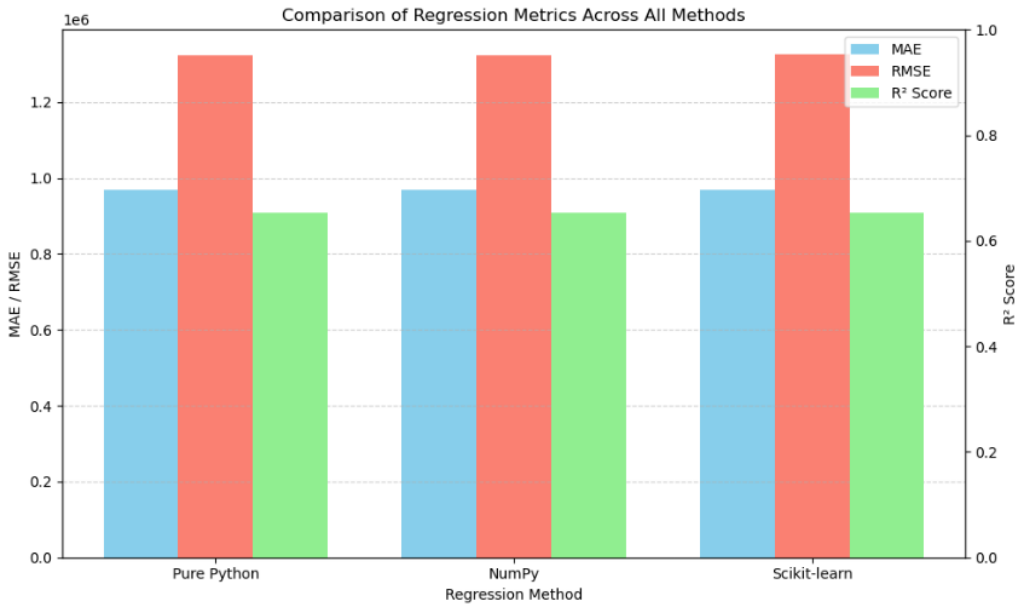

Figure 4: Comparison of MAE, RMSE, and R^2 across Pure Python, NumPy, and Scikit-learn.

Overall Comparison

Method	MAE	RMSE	R^2	Time (s)
Pure Python	969206.24	1324170.42	0.6531	3.55
NumPy	969206.24	1324170.42	0.6531	0.09
Scikit-learn	965500.00	1324506.96	0.6529	0.001

Table 1: Summary of performance metrics for all methods.

Analysis and Discussion

- **Convergence Time and Accuracy:** Scikit-learn converges instantly using a closed-form solution. NumPy is much faster than pure Python due to vectorization. Accuracy is similar across all.
- **Vectorization and Optimization:** NumPy uses matrix operations (SIMD under the hood), reducing loop overhead. Scikit-learn uses analytical methods (Normal Equation) which are highly optimized.
- **Scalability and Efficiency:** Pure Python is impractical for large datasets. NumPy is reasonably scalable. Scikit-learn is the most scalable and efficient for real-world applications.
- **Parameter Sensitivity:** Initial weights and learning rate significantly influence gradient descent. Poor choices may slow or prevent convergence.

Use of AI Assistance

To enhance the quality and clarity of this report, I utilized ChatGPT for guidance and support. Specifically, ChatGPT helped me:

- Understand and structure the implementation details of multivariable linear regression.(for normalize feature)
- Refine the explanations and improve the flow of technical sections.
- Format this document using correct LaTeX syntax, including tables, equations, and figures.
- Review the analysis and organize the comparative insights logically.

All code and experimentation were done independently, and ChatGPT was used solely as a learning and formatting assistant to improve presentation and comprehension.

Conclusion

All three methods reached similar R^2 scores ($\tilde{0}.65$), validating their correctness. While the pure Python version demonstrates the underlying logic of gradient descent, NumPy and Scikit-learn offer far superior performance. This analysis highlights how implementation choices impact training time while maintaining comparable predictive accuracy.