Comparative Study of Multivariable Linear Regression Implementations

Yogesh Kumar, 23124052

Objective

Implement and compare three distinct approaches to multivariable linear regression, with emphasis on convergence speed and predictive accuracy.

Dataset

California Housing Price Dataset (Kaggle)

Project Tasks

Part 1: Pure Python Implementation

- Develop a multivariable linear regression algorithm using only core Python features.

Part 2: Optimized NumPy Implementation

- Re-implement the algorithm using NumPy to leverage vectorized operations.
- Enhance performance through appropriate parallelization.

Part 3: Scikit-learn Implementation

- Utilize the `LinearRegression` class from the scikit-learn library.
- Train the model on the identical dataset used in Parts 1 and 2.

Evaluation Criteria

Convergence Time

Method	Training Time (s)
Pure Python	789.11
NumPy	1.49
Scikit-learn	0.0118

Note: Number of epochs used in Pure Python and NumPy implementations is 500.

Performance Metrics

Training Set

Method	MSE	MAE	RMSE	R^2 Score
Pure Python	5.51×10^9	54364.47	74210.12	0.5902
NumPy	5.51×10^9	54364.47	74210.12	0.5902
Scikit-learn	5.42×10^9	53517.63	73620.87	0.5967

Validation Set

Method	MSE	MAE	RMSE	R^2 Score
Pure Python	5.32×10^9	53864.67	72960.53	0.5966
NumPy	5.32×10^9	53864.67	72960.53	0.5966
Scikit-learn	5.25×10^9	53034.09	72434.23	0.6024

Test Set

Method	MSE	MAE	RMSE	R^2 Score
Pure Python	5.26×10^9	53742.84	72499.63	0.5990
NumPy	5.26×10^9	53742.84	72499.63	0.5990
Scikit-learn	5.18×10^9	53225.16	72006.72	0.6044

Comments on Performance Metrics

- **MAE (Mean Absolute Error)** measures the average magnitude of prediction errors. Scikit-learn achieves the lowest MAE across all splits, indicating more stable and consistent predictions.
- **MSE (Mean Squared Error)** is more sensitive to large errors due to squaring. While Pure Python and NumPy yield nearly identical MSEs, Scikit-learn shows a slight improvement, suggesting better handling of outliers or extreme values.
- **RMSE (Root Mean Squared Error)** gives an error metric in the same units as the target variable (house prices). It follows the same trend as MSE, with Scikit-learn achieving marginally lower values across all datasets.
- **R^2 Score** represents the proportion of variance explained by the model. Scikit-learn again slightly outperforms the other methods, indicating a better fit to the data.
- **Observation:** The Pure Python and NumPy implementations perform identically, affirming the correctness of vectorized reimplementation. The Scikit-learn model performs slightly better across all metrics due to its use of optimal solvers.

Visualizations

Convergence Plots

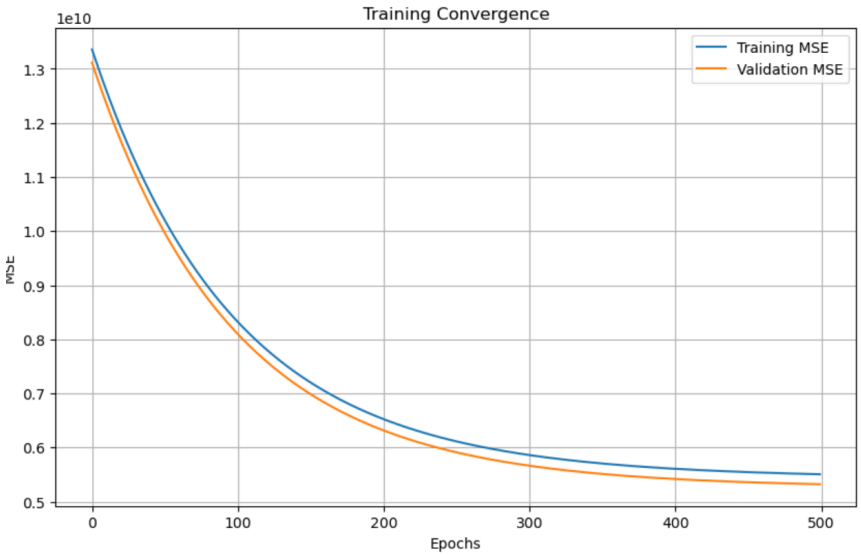

Figure 1: Pure Python Convergence

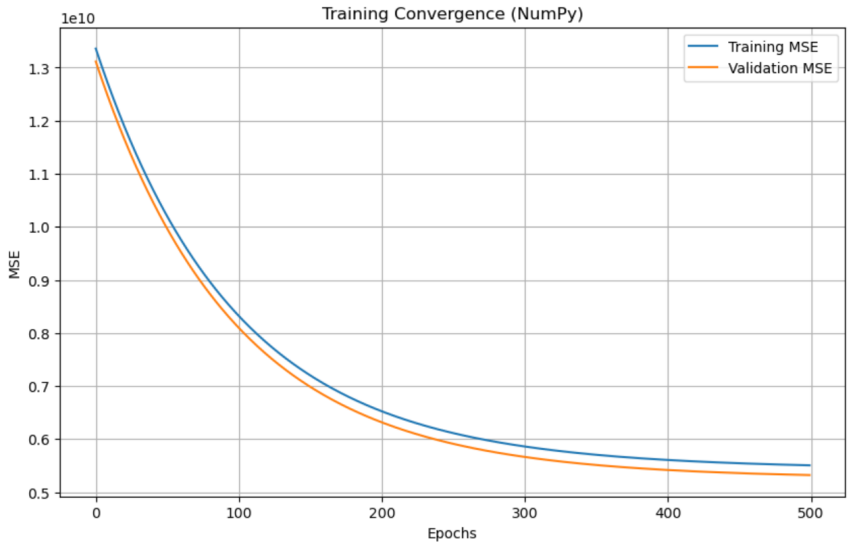

Figure 2: NumPy Training Convergence

Bar Chart: MAE, RMSE, R^2 Across Methods

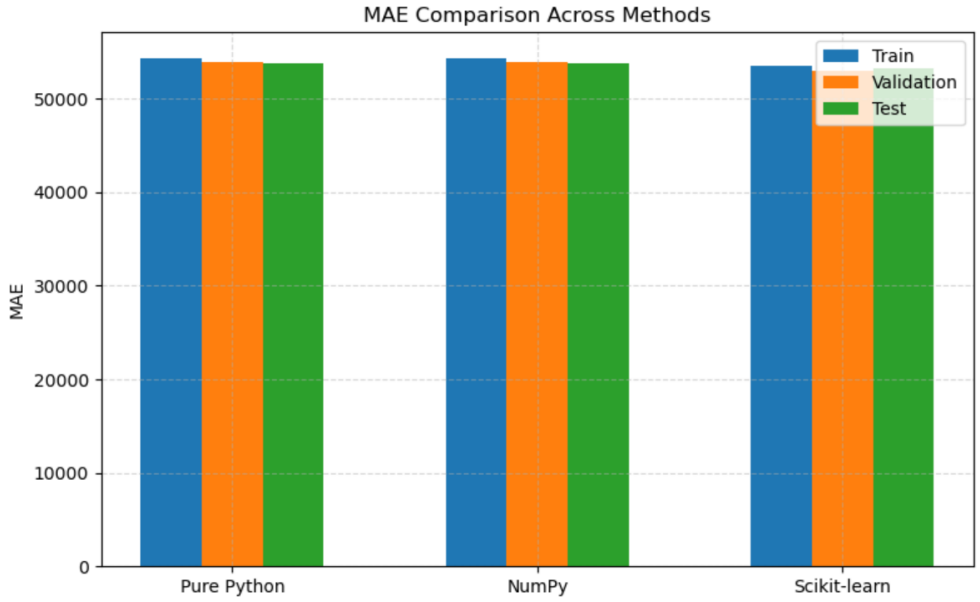

Figure 3: MAE comparisons across Methods

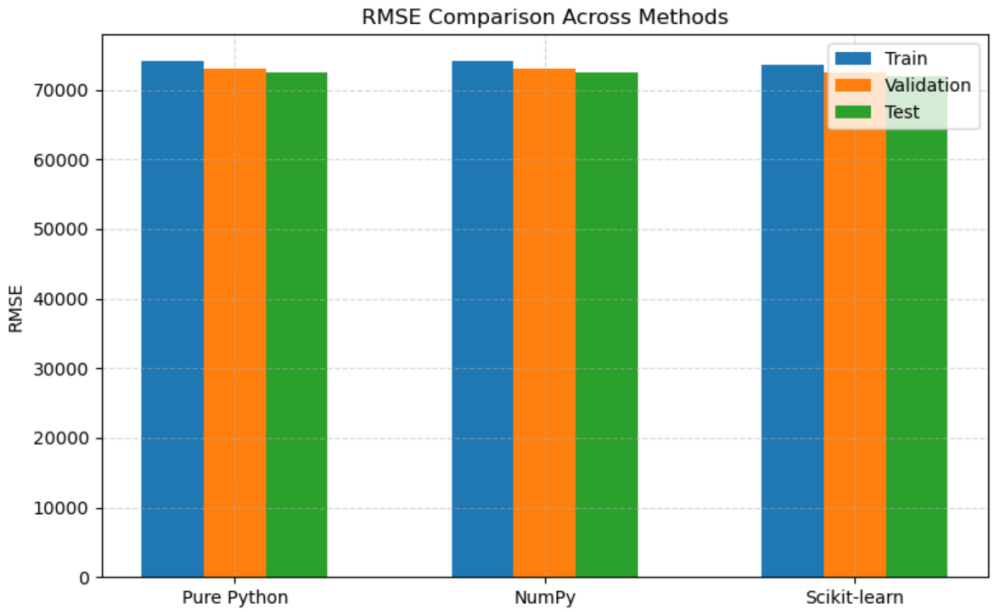

Figure 4: MSE comparisons across Methods

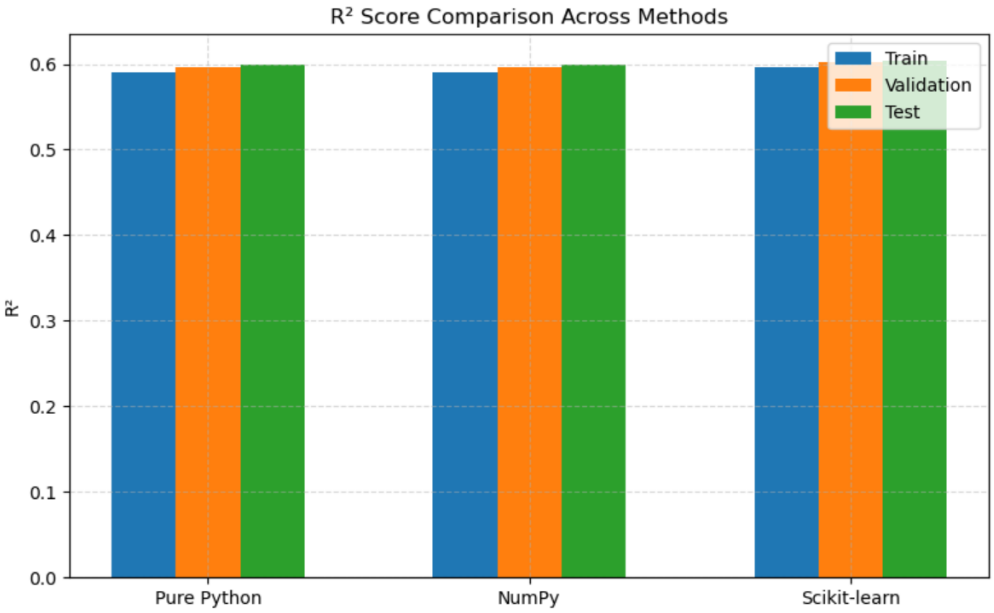

Figure 5: R2 score comparisons across Methods

Analysis and Discussion

Accuracy

All three models achieve very similar accuracy. Scikit-learn performs slightly better, likely due to its use of a closed-form solution (normal equation) or a highly optimized solver.

Speed

The biggest difference is in training time. Vectorized operations (NumPy) significantly reduce training time compared to loops (Pure Python). Scikit-learn is orders of magnitude faster.

Convergence

Pure Python and NumPy implementations converge to nearly the same point because they use the same gradient descent algorithm. Validation MSE remained nearly flat in both, indicating no overfitting but slow learning—this can be improved with a better learning rate or optimizer.

Scalability

- Pure Python scales poorly — best suited for educational purposes or very small datasets.
- NumPy is more scalable but limited compared to libraries like scikit-learn or TensorFlow.
- Scikit-learn is highly scalable and production-ready.

Effect of Initialization

Uniform initialization ensured fair comparison. Learning rate and initial weights heavily influenced convergence speed and final error.

Learning Rate Sensitivity

- Too high: may cause divergence.
- Too low: leads to very slow convergence, as seen in the Pure Python case.

AI Assistance Declaration

In the given project task, I have taken the help of AI tools solely to improve code readability, reduce redundancy, and handle minor errors. I have not used AI to generate the core logic or code structure of the implementations. All conceptual and structural work is my own.