Linear Regression Model Analysis and Comparison

Satyam Yadav

May 21, 2025

1 Introduction

This report presents a comparative analysis of three different implementations of linear regression on the California housing dataset:

1. Loop-based Gradient Descent
2. Vectorized Gradient Descent using NumPy
3. Scikit-learn's `LinearRegression`

The models are assessed based on convergence behavior, training time, and regression metrics like MAE, RMSE, and R^2.

2 EDA

2.1 Dataset

The dataset consists of 20,640 entries with 10 columns, summarized as follows:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

2.2 Missing Values

The dataset contains missing values only in the `total_bedrooms` column with 207 missing entries. All other columns are complete.

Column	Missing Values
total_bedrooms	207
all others	0

2.3 Summary Statistics

Feature	Mean	Std Dev	Min	Max
Longitude	-119.57	2.00	-124.35	-114.31
Latitude	35.63	2.14	32.54	41.95
Housing Median Age	28.64	12.59	1.00	52.00
Total Rooms	2635.76	2181.62	2.00	39320.00
Total Bedrooms	537.87	421.39	1.00	6445.00
Population	1425.48	1132.46	3.00	35682.00
Households	499.54	382.33	1.00	6082.00
Median Income	3.87	1.90	0.50	15.00
Median House Value	206,855.82	115,395.62	14,999.00	500,001.00

This overview gives a detailed insight into the dataset's composition and variability, helping guide further analysis steps.

2.4 Distribution of Median House Value

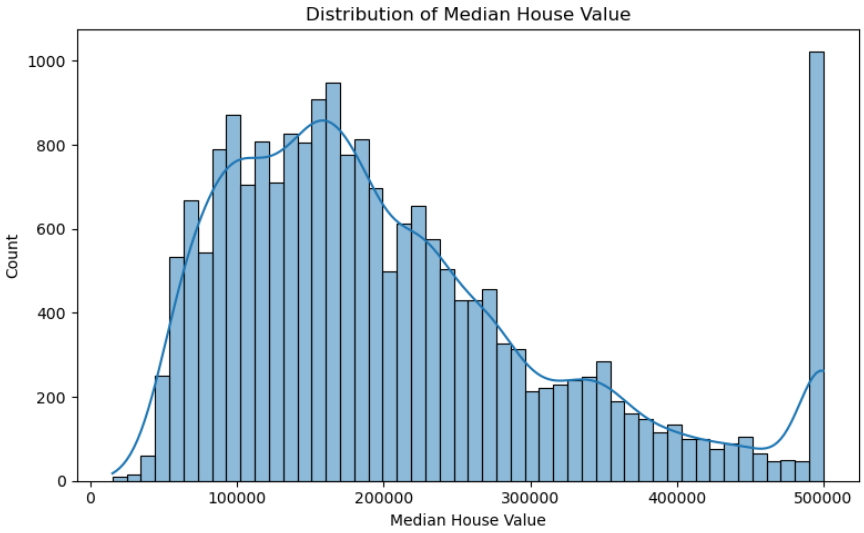

Figure 1: Distribution of median house values in the dataset.

2.5 Pairwise Relationships

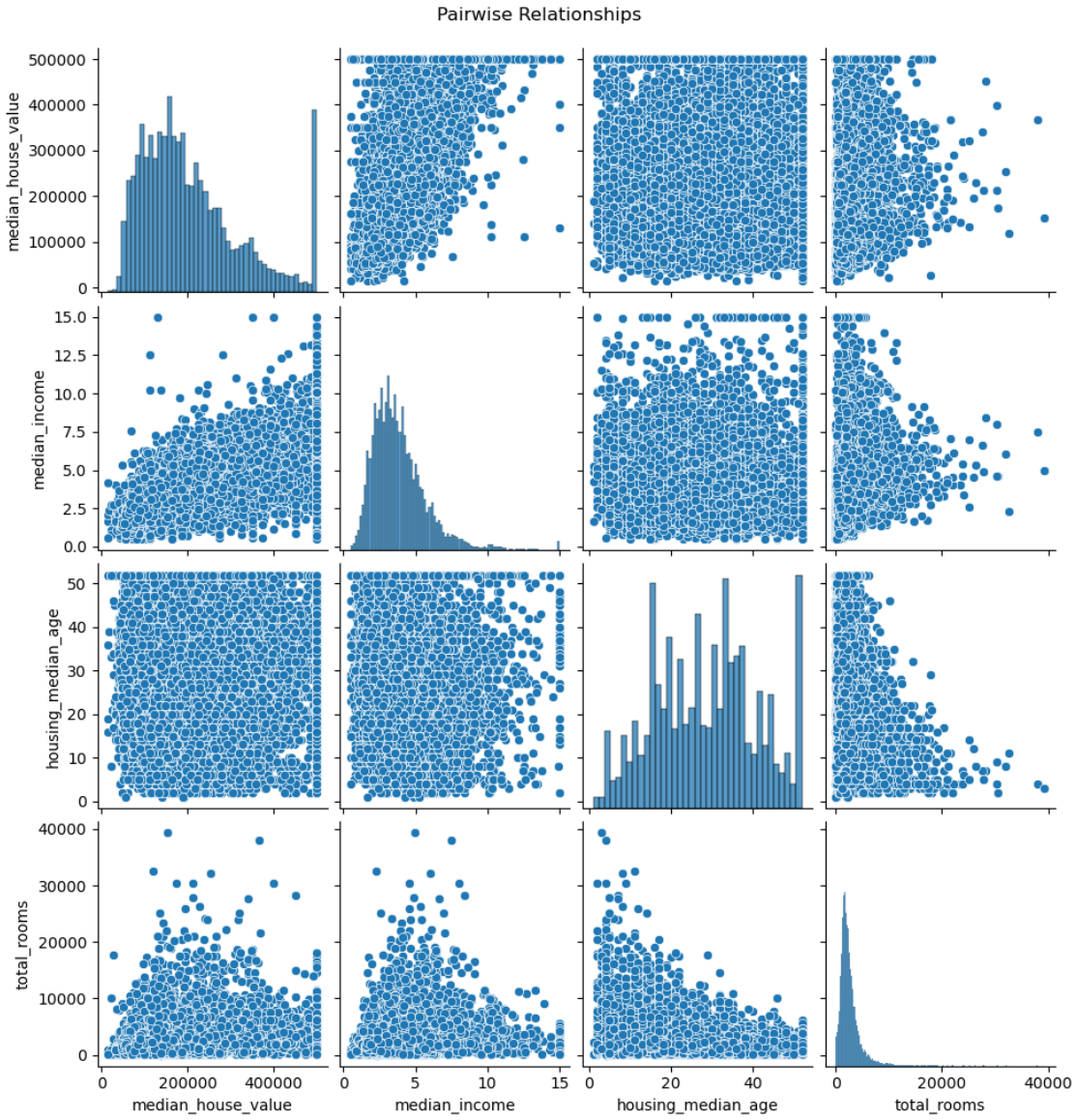

Figure 2: Pairplot showing relationships between income, age, total rooms, and house value.

2.6 Feature Correlation

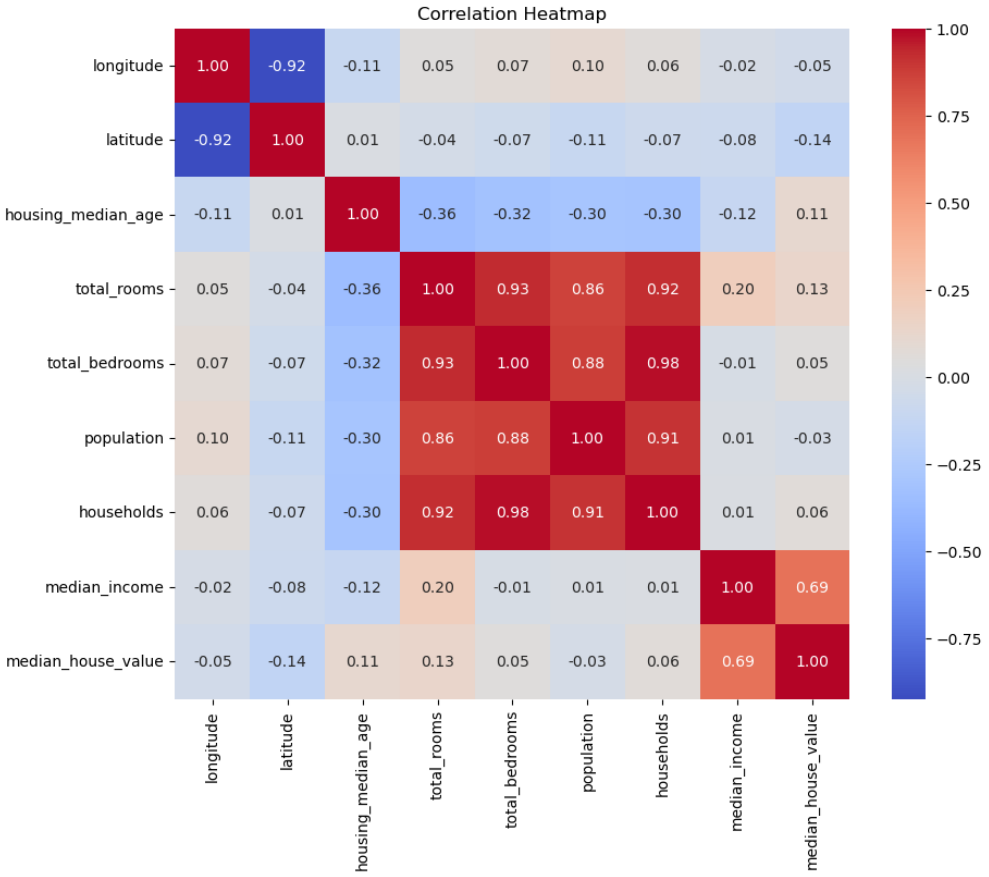

Figure 3: Heatmap of correlation among numerical features.

2.7 House Value by Ocean Proximity

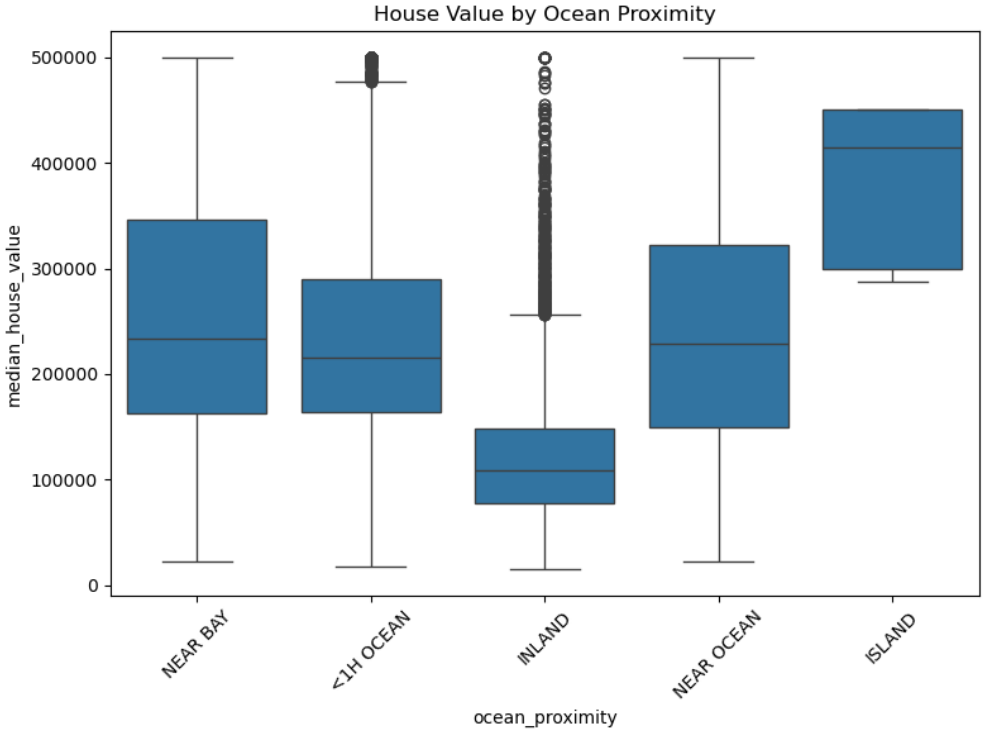

Figure 4: Median house value by proximity to the ocean.

2.8 Relationship Between Income and House Value

Figure 5: Scatter plot of median income vs. house value.

2.9 Distribution of Housing Age

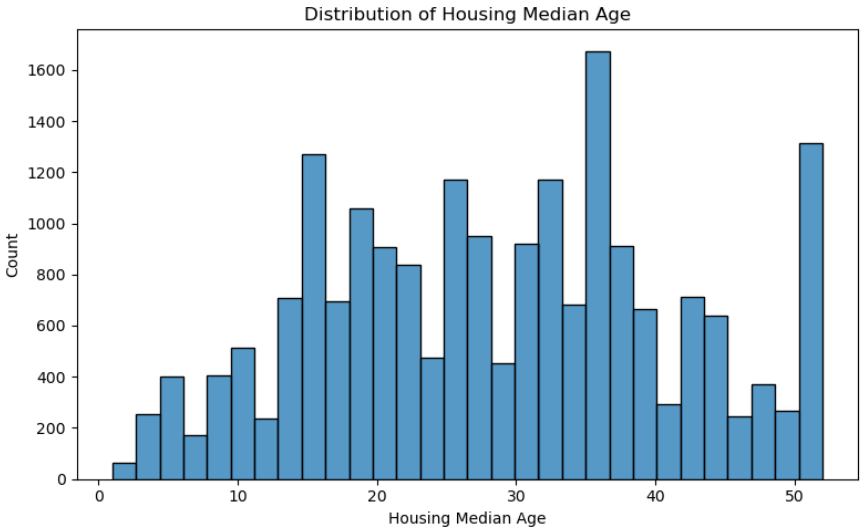

Figure 6: Distribution of housing median age across the dataset.

3 Data Preprocessing

1. First thing i did was removing all the zeroes in the data so it does not create any nan value.

2. Then there were missing data which i have used mean values to fill them.
3. Also i have used feature calling because some features had larger ranges than others.
4. Some feature like ocean proximity,lat,long i wasn't able to think of a way to implement them in training the model so i just dropped them.

4 Mathematical Foundations

4.1 Linear Regression Model

Given training data with m samples and n features, the linear model predicts the output $\hat{y}$ as:

$$\hat{y}^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)} = \boldsymbol{x}^{(i)} \cdot \boldsymbol{\theta}$$

where:

- $\boldsymbol{x}^{(i)}$ is the feature vector of the i-th sample.
- $\boldsymbol{\theta}$ is the vector of parameters (weights).

4.2 Cost Function (Mean Squared Error)

The objective is to minimize the cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

where $y^{(i)}$ is the true output and $\hat{y}^{(i)}$ is the predicted output.

4.3 Gradient Descent

To minimize $J(\boldsymbol{\theta})$, we update weights using:

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial J}{\partial \theta_j}$$

where α is the learning rate and the partial derivative is:

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right) x_j^{(i)}$$

In vectorized form:

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \cdot \frac{1}{m} \boldsymbol{X}^T (\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})$$

4.4 Normal Equation (Used by Scikit-learn)

The closed-form solution is:

$$\boldsymbol{\theta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

This is numerically efficient when n (number of features) is small.

5 Regression Metrics

5.1 Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{m} \sum_{i=1}^{m} \left| y^{(i)} - \hat{y}^{(i)} \right|$$

5.2 Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}$$

5.3 R-squared Score

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{m}(y^{(i)} - \bar{y})^2}$$

6 Performance Results

Method	MAE	RMSE	R^2 Score
Loop-Based GD	53192.05	72545.69	0.6152
Vectorized GD	53192.05	72545.69	0.6152
Scikit-learn LR	51372.67	70156.12	0.6401

Table 1: Performance Comparison Across Models

7 Observations and Discussion

7.1 Convergence Times and Accuracies

- **Loop-based GD** took longer due to explicit iteration and was computationally expensive.
- **Vectorized GD** achieved identical accuracy but was significantly faster due to matrix operations.
- **Scikit-learn** provided the best performance and accuracy using optimized internal solvers.

7.2 Causes for Differences

- **Vectorization** reduces overhead from Python loops, enabling faster convergence.
- **Solver Type:** Scikit-learn uses closed-form solutions or QR/SVD decompositions which are more stable.
- **Initialization and Learning Rate:** Poor choices in θ_0 or α can prevent convergence in gradient descent.

7.3 Scalability and Efficiency

- Loop-based GD is impractical for large datasets.
- Vectorized GD scales well for large m and n.
- Scikit-learn is efficient for most use cases but may not scale to massive datasets unless using online variants like `SGDRegressor`.

7.4 Effect of Learning Rate and Initialization

- Choosing a small learning rate can slow down convergence significantly.
- Large learning rates might cause oscillation or divergence.
- Initialization (e.g., all zeros or random) can influence the number of epochs needed.

8 Conclusion

This analysis highlights that while all three methods can produce reliable linear models, their usability depends on the context. Gradient descent teaches foundational concepts but is less practical. Scikit-learn offers the best combination of accuracy, efficiency, and ease of use for most applications.