# Linear Regression from Scratch - California Housing Dataset

**Roll Number:** g25ait1010
**Date:** October 2025

## 1. Assumptions Made

The following assumptions were made during this implementation:

- All input features are standardized to have mean 0 and variance 1 using StandardScaler
- The target variable (house prices) is left unscaled for interpretability
- The California Housing dataset has no missing values
- A linear relationship exists between features and target variable
- The cost function is convex, ensuring global minimum convergence

## 2. Resources Used

The following resources were utilized during the implementation:

1. **Scikit-learn Library** - For dataset loading and train-test split
2. **NumPy Library** - For numerical computations and matrix operations
3. **Matplotlib Library** - For creating visualizations and plots
4. **StandardScaler** - From scikit-learn for feature standardization
5. **GeeksforGeeks** - Gradient Descent algorithm reference
6. **Machine Learning Course Materials** - For theoretical understanding

## 3. Dataset Information

| Attribute | Details |
|---|---|
| Dataset Name | California Housing Dataset |
| Source | Scikit-learn fetch_california_housing |
| Total Samples | 20,640 |
| Features | 8 numerical features |
| Target | Median house value |
| Train Split | 80% (16,512 samples) |
| Test Split | 20% (4,128 samples) |

## 4. Implementation Details

### 4.1 Linear Regression Model

The Linear Regression model implements the equation: **$\hat{y} = X \cdot w + b$** where $\hat{y}$ represents predicted values, X is the input features matrix, w is the weights vector, and b is the bias term.

### 4.2 Training Algorithm

The implementation uses gradient descent optimization with the following approach:

1. **Parameter Initialization**: Weights and bias are initialized to zero
2. **Forward Pass**: Model predictions are computed using current parameters
3. **Cost Calculation**: Mean Squared Error is calculated between predictions and actual values
4. **Gradient Computation**: Gradients are computed for both weights and bias
5. **Parameter Update**: Parameters are updated using the learning rate and computed gradients
6. **Iteration**: This process repeats for the specified number of iterations

### 4.3 Cost Function

The Mean Squared Error serves as the cost function: **$\text{Cost} = (1/m) \times \Sigma(\hat{y}_i - y_i)^2$** where m is the number of training examples.

### 4.4 Gradient Updates

The gradient descent updates follow these rules:

- **Weight Update**: $w = w - \alpha \times$ (gradient of weights)
- **Bias Update**: $b = b - \alpha \times$ (gradient of bias)

Where $\alpha$ is the learning rate parameter.

### 4.5 Prediction Method

To generate predictions, the learned linear function is applied to test features using the final weights and bias values.

### 4.6 Evaluation Metrics

Model performance is assessed using two primary metrics:

- **Mean Squared Error (MSE)**: Measures average squared difference between actual and predicted values
- **R-squared ($R^2$)**: Measures the proportion of variance explained by the model

## 5. Experimental Results

### 5.1 Base Model Performance

| Dataset | Mean Squared Error (MSE) | $R^2$ Score |
|---|---|---|
| Training Set | 0.5457 | 0.5765 |
| Testing Set | 0.5238 | 0.6405 |

The model demonstrates good generalization as test error is slightly lower than training error, indicating no overfitting.

### 5.2 Ridge Regression Results (Bonus)

| Model Type | $\lambda$ Parameter | Testing MSE | Testing $R^2$ |
|---|---|---|---|
| Base Model | 0 | 0.5238 | 0.6405 |
| Ridge Regression | 10 | 0.5179 | 0.6471 |

L2 regularization with $\lambda=10$ provides marginal improvement in both MSE and $R^2$ scores.

### 5.3 Learning Rate Analysis (Bonus)

| Learning Rate | Convergence Behavior | Final MSE | Status |
|---|---|---|---|
| 1.0 | Diverges/Oscillates | ~5.0 | Too High |
| 0.01 | Smooth Decay | ~0.52 | Optimal |
| 0.0001 | Very Slow Descent | ~1.5 | Too Low |

## 6. Visualization Analysis

### 6.1 Learning Curve (MSE vs. Iterations)

**Figure 1: Learning Curve**

The learning curve demonstrates:

- Steep initial drop: Large errors corrected quickly in first ~100 iterations
- Gradual flattening: Fine-tuning of weights in later iterations
- Convergence: Stable minimum reached around iteration 900-1000

### 6.2 Actual vs. Predicted Values

**Figure 2: Actual vs. Predicted Values**

The scatter plot reveals:

- Good alignment: Points cluster around the 45° diagonal line
- Higher variance: More spread observed for expensive houses
- Model fit: $R^2 \approx 0.64$ indicates model explains 64% of target variance

### 6.3 Learning Rate Comparison

**Figure 3: Effect of Learning Rate on Convergence**

The comparison demonstrates:

- LR = 1.0: Causes divergence due to overshooting the minimum
- LR = 0.0001: Converges too slowly, requiring many more iterations
- LR = 0.01: Optimal balance between speed and stability

## 7. Key Observations

### 7.1 Model Performance

- The implemented linear regression successfully learns patterns in the California Housing dataset
- Test $R^2$ score of 0.6405 indicates reasonably good predictive performance
- Close train-test error values suggest proper generalization without overfitting

### 7.2 Convergence Analysis

- Model converges reliably within 1000 iterations using learning rate 0.01
- Learning curve shows expected exponential decay pattern
- No oscillations or divergence observed with optimal hyperparameters

### 7.3 Regularization Impact

- L2 regularization with $\lambda=10$ provides marginal improvement
- Regularization helps prevent overfitting by penalizing large weight values
- Small improvement suggests the base model was already well-regularized

### 7.4 Hyperparameter Sensitivity

- Learning rate significantly affects convergence behavior
- Too high learning rates cause instability and divergence
- Too low learning rates result in slow training and poor convergence

## 8. Conclusion

The implementation successfully demonstrates Linear Regression using Gradient Descent on the California Housing dataset. The model achieves satisfactory performance with MSE of 0.524 and $R^2$ of 0.641 on the test set. The experiments with different learning rates and regularization provide valuable insights into hyperparameter tuning. The visualization results confirm proper model training and convergence behavior.

**Key Achievements:**

- Complete from-scratch implementation without built-in ML libraries

- Comprehensive evaluation using multiple metrics and visualizations
- Successful implementation of Ridge regression and learning rate analysis
- Professional documentation following assignment requirements

**References:**


1. Scikit-learn Documentation - California Housing Dataset
2. GeeksforGeeks - Linear Regression and Gradient Descent
3. NumPy and Matplotlib Documentation