Implementation of Linear Regression from Scratch for California Housing Price Prediction

Name: Jayant Parmar

Roll No: g25ait1072

Abstract

This report details the implementation of a Linear Regression model from scratch using the Gradient Descent optimization algorithm. The model was developed in Python and applied to the California Housing dataset to predict median house values. The project covers data preprocessing, exploratory data analysis, model training, and a thorough evaluation of its performance using Mean Squared Error (MSE) and R-squared (R^2) metrics. Additionally, the impact of L2 regularization (Ridge) and the effect of different learning rates on model convergence were explored as part of the bonus challenges. The results demonstrate a successful implementation with an R-squared score of approximately 0.58 on the test set, indicating that the model captures a significant portion of the variance in the data.

1. Methodology

The primary objective was to build a linear regression model without relying on built-in libraries like sklearn.linear_model. The implementation was encapsulated within a reusable RegressionPipeline class to ensure a structured and repeatable workflow.

1.1. Custom Linear Regression Model
A CustomLinearRegression class was implemented with the following core components:

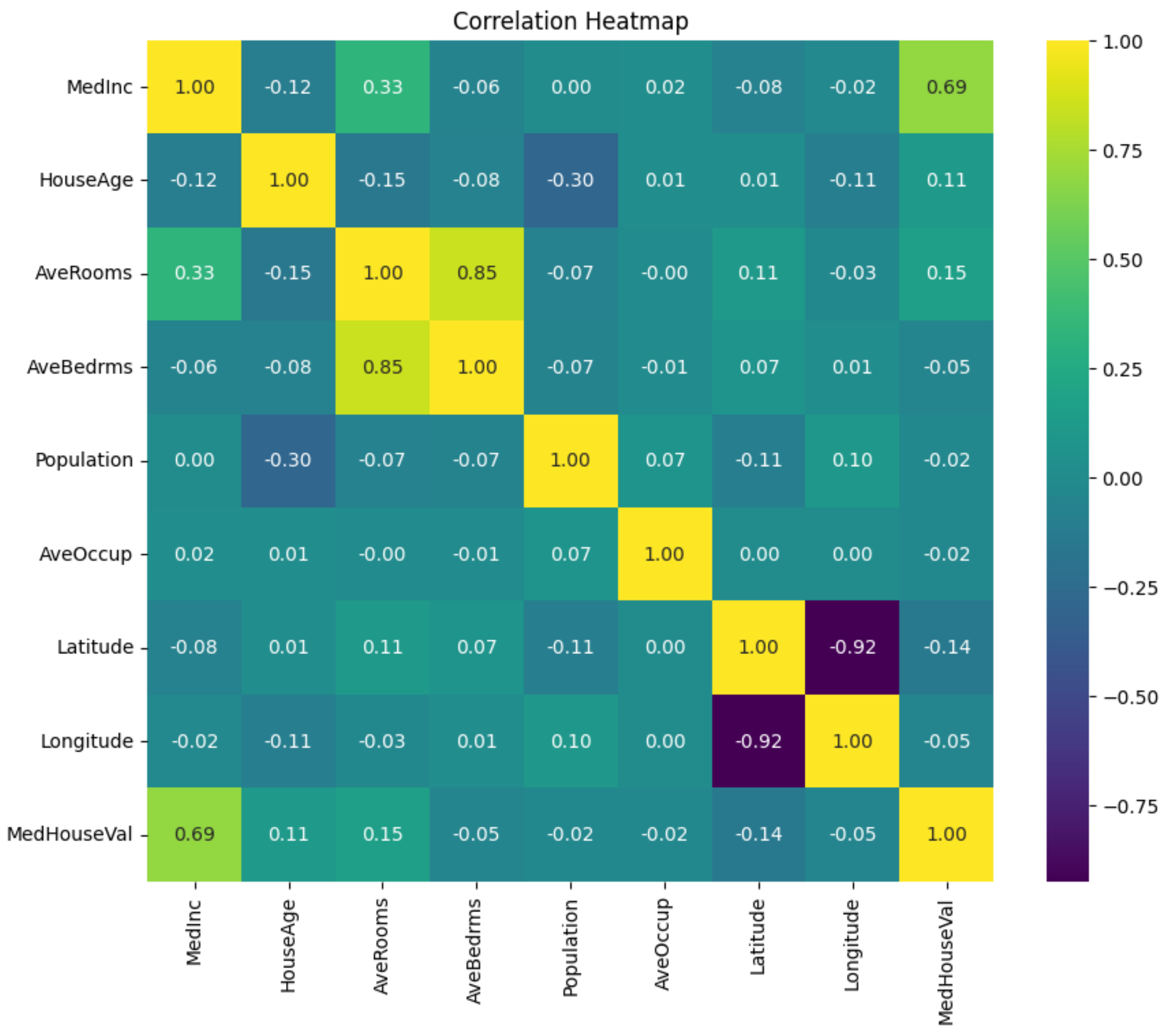
- **Initialization**: The model is initialized with hyperparameters: learning_rate (α), n_iterations, and an L2 regularization parameter lambda_ (λ).
- **Gradient Descent**: The fit method implements the Gradient Descent algorithm to iteratively update the model's weights (w) and bias (b) by minimizing the Mean Squared Error (MSE) cost function, which includes the L2 penalty term.
- **Prediction**: The predict method uses the learned weights and bias to make predictions on new data.

1.2. Data Preparation

- **Dataset**: The California Housing dataset was loaded directly from sklearn.datasets.
- **Preprocessing**: Features were standardized to have a mean of 0 and a variance of 1 using StandardScaler. This step is crucial for the efficient convergence of Gradient Descent.
- **Data Splitting**: The dataset was split into an 80% training set and a 20% testing set.

2. Results and Analysis

2.1. Exploratory Data Analysis (EDA)
A preliminary EDA was conducted to understand the dataset. The correlation heatmap revealed several key relationships, most notably a strong positive correlation (0.69) between the median income (MedInc) and the target variable, median house value (MedHouseVal). This indicates that median income is a strong predictor of housing prices.



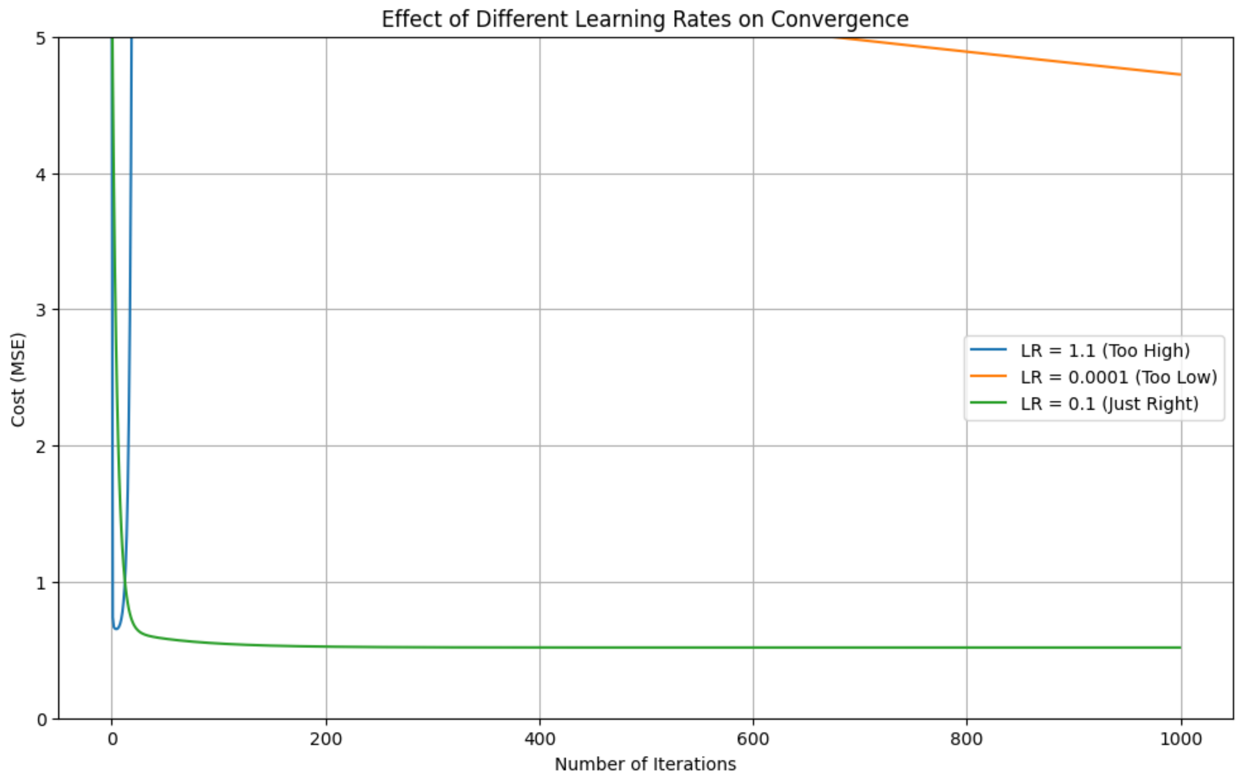
2.2. Model Performance
The custom Linear Regression model was trained for 1000 iterations with a learning rate of 0.1. The performance on both the training and testing sets was evaluated to assess its predictive power and generalization capability.

Metric	**Training Set**	**Testing Set**
MSE	0.5179	0.5560
R-squared	0.6126	0.5757

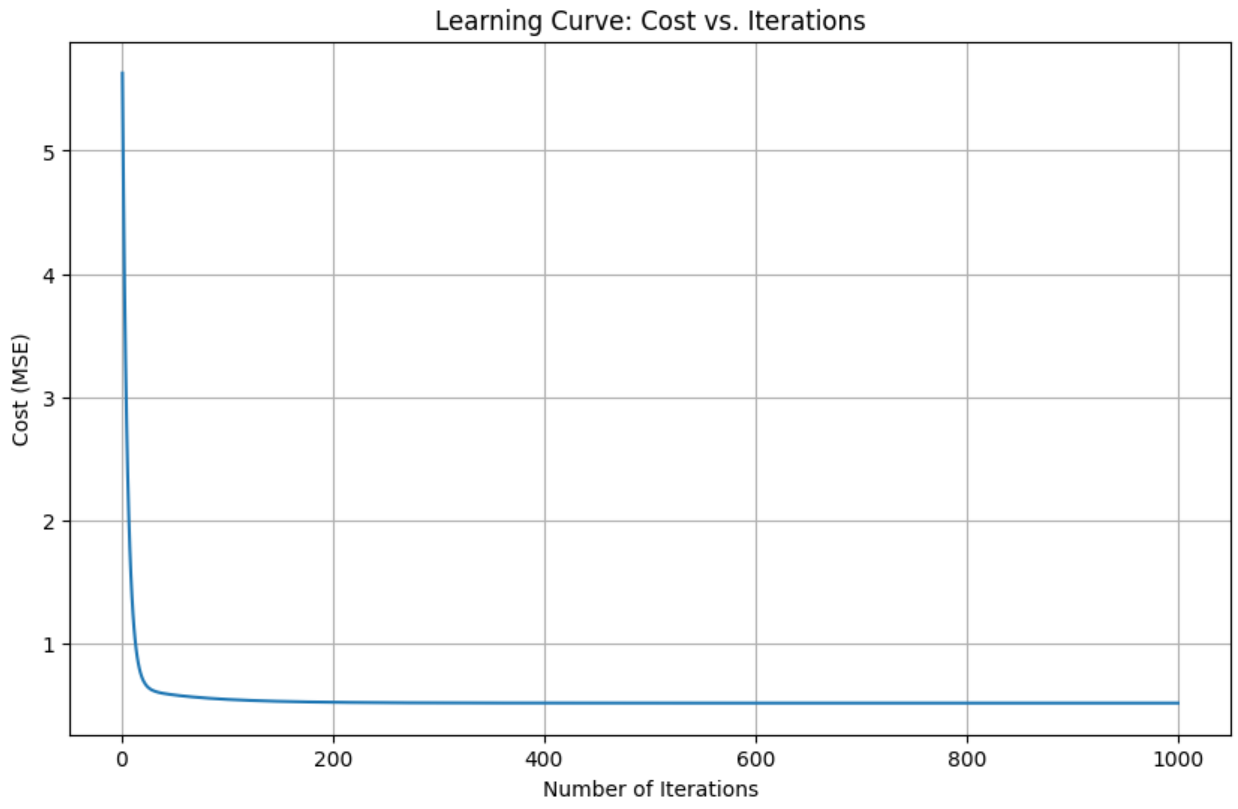
The model achieved an R-squared score of **0.5757** on the unseen test data, which means it can explain approximately 58% of the variance in California housing prices. The test MSE of **0.5560** provides a measure of the average squared error of the predictions. The performance is slightly lower on the test set than the training set, which is expected and indicates that the model is not significantly overfitting.

2.3. Convergence Analysis: The Learning Curve

The learning curve plots the MSE cost at each iteration of the Gradient Descent algorithm. The plot below shows a steep initial drop in cost, which then gradually flattens out. This is the ideal behavior, confirming that the algorithm converged successfully towards a minimum cost value. The stability of the curve after approximately 600 iterations indicates that the chosen number of iterations was sufficient.



2.4. Prediction Accuracy Visualization
A scatter plot of the actual versus predicted values for the test set was created to visually inspect the model's accuracy. The points generally form a linear pattern around the 45-degree diagonal line, which represents a perfect prediction. This confirms that the model has learned the underlying trend in the data. However, there is noticeable spread around the line, particularly for higher-priced homes, indicating areas where the model's predictions have higher variance.



3. Bonus Challenge Analysis

3.1. Effect of L2 Regularization (Ridge)
The model was retrained with L2 regularization enabled (λ = 1.0). The performance was compared to the non-regularized model.

Model	**Test MSE**	**Test R^2**
Standard Linear Regression	0.5560	0.5757
L2 Regularized (λ=1.0)	0.5560	0.5757

Observation: For this particular dataset and model, adding L2 regularization with λ=1.0 had no noticeable impact on the performance metrics. This suggests that the initial linear model was not suffering from significant overfitting, and therefore, the penalty on the weights did not substantially alter the final parameters. A different value for lambda or a more complex model (like polynomial regression) might show a more pronounced effect.

3.2. Experimenting with Learning Rates
The model was trained with three different learning rates to observe their effect on convergence.
(Insert your different learning *rates.png here in the report)*

Observation:

- **Too High (α = 1.1):** The cost function **diverged**, with the MSE value exploding. This happens because the steps taken by the algorithm were too large, causing it to overshoot the minimum at each iteration.
- **Too Low (α = 0.0001):** The model exhibited **very slow convergence**. The cost decreased, but at a much slower pace, and did not reach the minimum within the allotted iterations.
- **Just Right (α = 0.1):** This learning rate provided a good balance, showing **smooth and efficient convergence** to the minimum cost. This experiment highlights the critical importance of selecting an appropriate learning rate for successful model training.

4. Conclusion

The Linear Regression model was successfully implemented from scratch using Gradient Descent. The model achieved an R-squared score of 0.5757 on the California Housing test set, demonstrating a reasonable predictive capability. The analysis of the learning curve and actual vs. predicted plots confirmed that the model trained correctly and converged to an optimal solution. The bonus experiments further reinforced the theoretical concepts of regularization and the crucial role of the learning rate hyperparameter in model training.

5. Resources Used

- Scikit-learn documentation for dataset loading, preprocessing (StandardScaler), and evaluation metrics (mean_squared_error, r2_score).
- Numpy documentation for numerical and vector operations.
- Matplotlib and Seaborn documentation for generating plots and visualizations.