Simple Linear Regression Using Gradient Descent: A Predictive Modeling Approach

Overview

The goal of this project is to implement a simple linear regression model using the **gradient descent** algorithm, as described in the lecture notes. The objective is to analyze the relationship between two variables, x (independent variable) and y (dependent variable), and to compute the slope, intercept, and MSE over iterations. The dataset myData.csv contains 100 data points with two attributes, x and y. The model parameters-slope and intercept – are optimized iteratively to minimize to the Mean Squared Error (MSE).

METHODOLOGY

This project assesses the performance of simple linear regression through manual computation. MyData.csv dataset was uploaded into Python environment and processed without using any libraries, except Matplotlib for visualization. The model was initialized with the following parameters:

Slope (m) = 0
Intercept (b) = 0
Learning rate = 0.000001
Number of iterations = 1000

The gradient descent algorithm was applied to iteratively update the slope and intercept values based on the learning rate. The predicted values were computed as:

Predicted = Slope * x + Intercept

Throughout the process, the **Mean Squared Error (MSE)** was logged at each iteration, enabling evaluation of model performance using **R-squared (R^2)**. Finally, the regression line was generated alongside a scatter plot, and MSE was plotted over iterations to visualize convergence.

IMPLEMENTATION

- The dataset is read manually from myData.csv.
- Gradient descent is implemented iteratively for optimization.
- MSE for each iteration is logged in SLRTraining[Iterations][LearningRate]MSEs.txt.
- Final model parameters, including slope, intercept, final MSE, and R-squared, are saved in SLRModelParameters.txt.
- The regression line is plotted and saved as RegressionPlot.png.
- The MSE per iteration is plotted and saved as MSEPlot.png

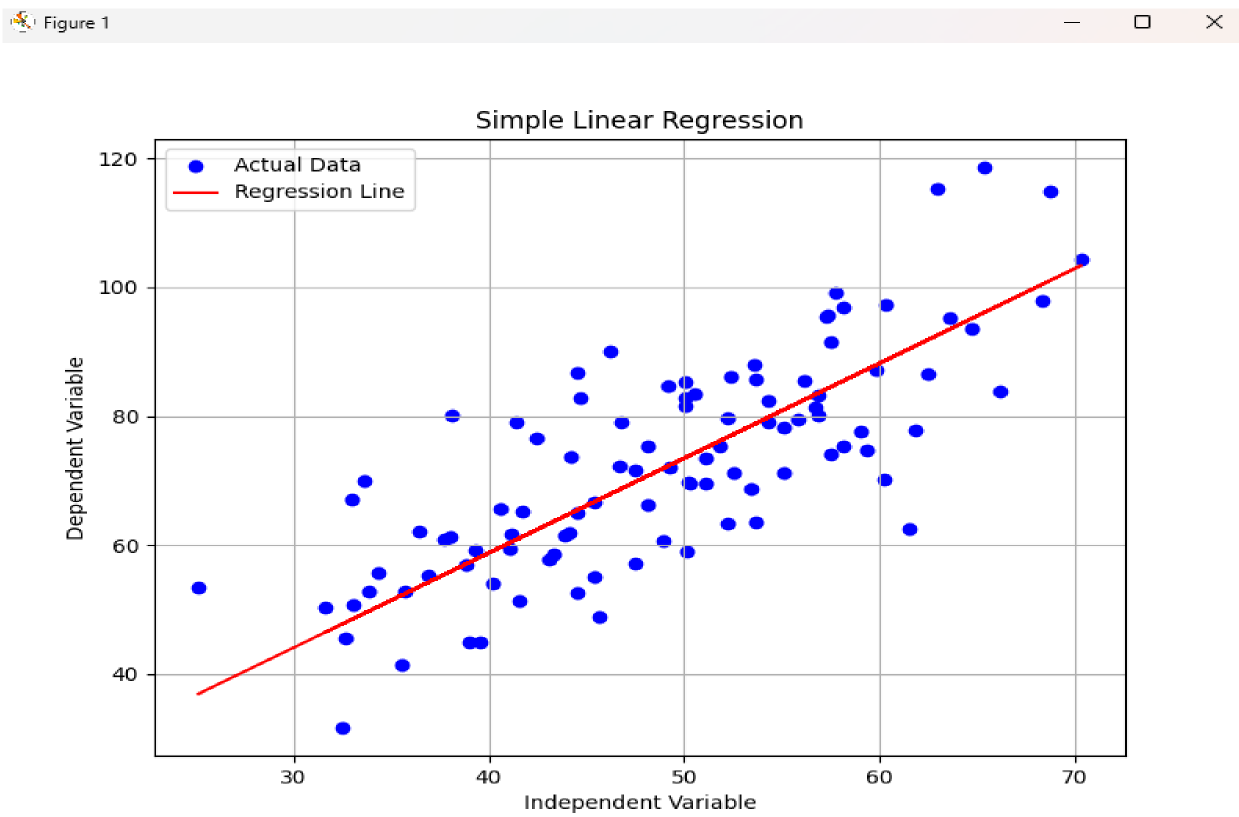
RESULT

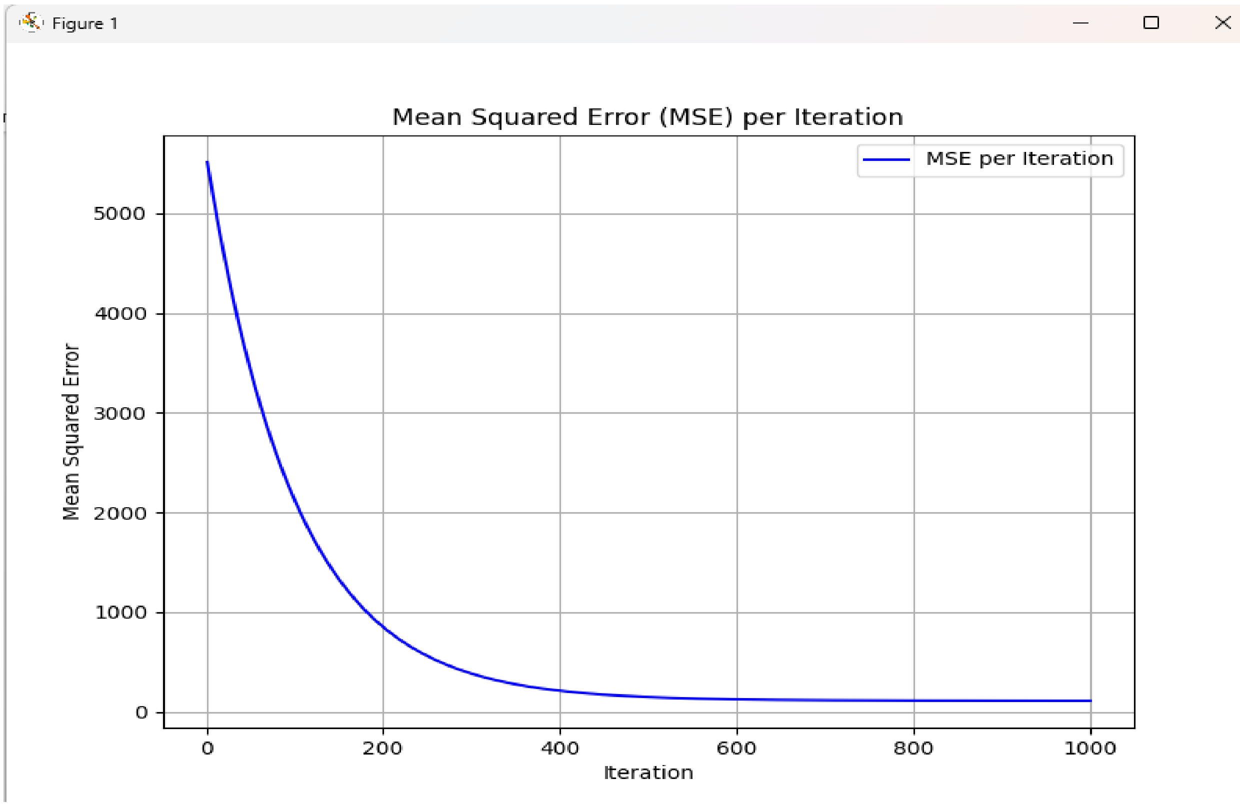
The following metrics were computed for the linear regression model:

Slope (m): 1.4689072254572764
Intercept (b): 0.029472081221912246
R-squared (R^2): 0.5890376513646982
MSE Log File: SLRTraining[1000][1e-06]MSEs.txt
Model Parameters File: SLRModelParameters.txt
MSE Plot File: MSE_Per_Iteration_Plot.png
Regression Plot File: RegressionPlot.png

- The slope of 1.4689 suggests that for each unit increase in the independent variable, the dependent variable increases by approximately 1.47 units.
- The intercept of 0.0295 indicates that when the independent variable is zero, the predicted value of the dependent variable is close to zero.
- The R-squared value of 0.5890 means that about 58.9% of the variance in the dependent variable is explained by the independent variable, indicating a moderate fit.

Regression Plot





DISCUSSION

The regression line indicates a moderate positive relationship between the dependent and the independent variables as R-squared = 0.5890. The intercept line almost passed through the origin. The MSE curve demonstrates the learning process of the gradient descent algorithm. Initially, the MSE value was very high (~ 5500), indicating a poor model fit. Over time, MSE value decreases rapidly as the model is improving and finally, the curve flattened out around iteration between 600 – 800 indicating that convergence is at optimal. The final MSE was relatively low, meaning that the model has effectively minimized the predicted errors.

Challenges:

1. **Data Preprocessing:** Parsing the dataset and handling potential issues like missing or malformed data was time-consuming. This was solved by carefully parsing the data and validating it before using it.

2. **Numerical Stability:** With large-scale data, numerical stability was a concern. Scaling the data before training would have helped, but it was not implemented in this version.

3. **Gradient Descent Tuning:** Finding the right learning rate and number of iterations took time.

Straightforward Steps:

- Implementing the gradient descent algorithm was straightforward as it followed a standard formula.
- Plotting and saving graphs using Matplotlib took minimal time due to Matplotlib's built-in features.
- Writing results to files was easily implemented using Python's file handling capabilities.

Conclusion

This implementation successfully demonstrates a simple linear regression model using gradient descent. The results show a moderate correlation between the independent and dependent variables. Future improvements may involve feature scaling, tuning hyperparameters, or exploring more complex models to enhance prediction accuracy.

References

- Course materials from DATA 527.
- Online resources on linear regression and gradient descent.
- Python documentation for mathematical computations and plotting.