For the implementation of the univariate linear regression model, multiple functions were created to facilitate the training of the model. The actual function for the prediction is modeled using a simple expression where theta[0] is the intercept and theta[1] is the slope; it should be noted that with this being *univariate* linear regression, we can only utilize one feature which is how it follows this algebraic model. The coefficients for theta[1] (m) and theta[0] (b) are initialized at a given value and adjusted using gradient descent. The gradient descent approach for the theta adjustments utilizes a cost function, based off a selected metric – *Mean Squared Error* (MSE) in this case, along with a defined *learning rate*. MSE is defined as follows where *m* is defined as the total number of samples from the summation:

The gradient descent used to modify the value of each theta is as follows where *lr* denotes the set learning rate:

Alongside these methods, methods were defined for creating train/dev/test splits of the data, retrieving the feature/label associated with the dataset, and a validation check for data leaking. The dataset was retrieved using sklearn’s built in datasets, specifically their load\_diabetes dataset. Data and Target were loading into independent variables, then feature selection was performed on the input. The SelectKBest method from sklearn was used with k=1 to select a single feature. The selection metric was f\_regression to determine feature ranking off of the Pearson’s R score for each input. After validating that the input was reduced from a shape of (422, 10) to (422, 1), input and target were processed into an array then shuffled. From there the train, dev, and test splits were generated with respective percentages of 70%, 15%, 15%. The data was checked for leaks, with some runs having a single instance of data leakage and others having no instances of data leakage. The initial testing was done with theta initialized at [1.0, 1.0] but was quickly adjusted to [50.0, 1000.0] in initial tuning against the dev set, and again to [500.0, 1800.0] by the end of tuning. The learning rate was determined initially to be 0.01 and reduced at each instance of divergence by a factor of 10. Once it was reduced to a point without divergence, it was stepped up slowly by a rate of the last non-divergent learning rate used. By the end of optimization, the learning rate was set as 0.0040.

Optimizing training iterations was also a point of contention in the training of the model. Utilizing too high of a learning rate, or too high of an iteration count resulted in a horizontal line around 150. This worked to reduce the cost of the model to nearly 0 but removed any sense of correlation from the model. Instead, iterations that were set to 1000 initially was slowly stepped down to 325, sticking to increments of 25 for the purpose of displaying a 5x5 grid of graphs during training to verify model improvement visually. In instances where the cost of the model (MSE) approached single digit numbers, the model performed incredibly off that metric in dev and train but would have massive MSE for the test set. This was an indicator that lowest MSE did not mean that it was the greatest accuracy model for performance, as it could. As additional validation against the test set, R-Squared and Explained Variance were used to check against sacrificing an accurate model in the pursuit of low MSE. For the best MSE performance on test set, 130.2 was the best MSE; however, for the most accurate model in terms of correlation, the MSE was 4980221.84 and is included in the attached printout of the colab document.

Table

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Table

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated