# Gradient Descent for Univariate Linear Regression

Daniel Sherman, *daniel.sherman@ryerson.ca, (501034155)*

## I. Implementation Steps

The mark data were loaded from a csv file from a GitHub repository[1]. The data can be seen in Figure 1.

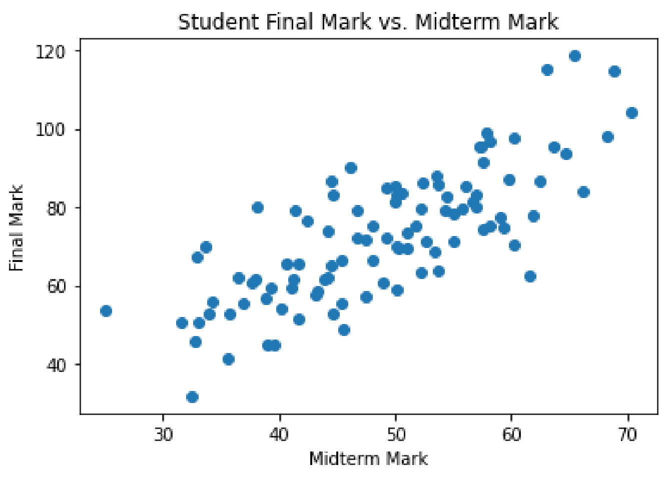

Fig. 1: Raw Midterm Mark vs. Raw Final Mark

The midterm and final marks were standardized according to the Z-Score, shown in Equation 1, where *x'* is the Z-Score, *x* is the raw mark, $\bar{x}$ is the average mark, and $\sigma$ is the standard deviation of the mark.

$$x' = \frac{x - \bar{x}}{\sigma} \tag{1}$$

A linear model between midterm and final mark is being designed, so two weights are being optimized, the slope, *m*, and the intercept, *b*. The initial seed for them was chosen to be -0.5 and 0, respectively. Having defined weights for the model, a predictive value can be initialized for the final mark, $\hat{y}$, based on the midterm mark, *x*, seen in Equation 2.

$$\hat{y} = mx + b \tag{2}$$

The difference between the predicted final mark and the actual final mark is quantified by the sum of squared error, *E*, shown in Equation 3, where *N* is the number of samples.

$$E = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3}$$

To optimize the weights, the slope and intercept are updated according to the learning rate, $\alpha$, and the gradient of the error function. Because the model is relatively simple, only having two weights, the gradient can be found analytically, seen in Equation 4 and Equation 5.

$$\frac{\partial E}{\partial m} = \frac{2}{N} \sum_{i=1}^{N} -x_i(y_i - (mx_i + b)) \tag{4}$$

$$\frac{\partial E}{\partial b} = \frac{2}{N} \sum_{i=1}^{N} -(y_i - (mx_i + b)) \tag{5}$$

To update each weight, $\theta_i$, based on the learning rate, $\alpha$, Equation 6 is utilized.

$$\theta_{i,new} = \theta_{i,old} - \alpha \frac{\partial E}{\partial \theta_i} \tag{6}$$

Two learning rates were explored, 0.1 and 0.0001. Gradient Descent was performed for Raw data and standardized to the Z-Score. Error was quantified over 2000 iterations of Gradient Descent, and the regression line was observed at iteration 100 and 2000.

Results were verified utilizing functions in the Python library `sklearn`.

## II. Discussion

The initial regression line is overlayed on top of the raw mark data and the standardized mark data in Figures 2 and 3, respectively. The initial regression is obviously a poor model for the data.

With the small learning rate (0.0001), the standardized data does not converge even after 2000 iterations (Figure 4 and Figure 5), but the raw data does appear to converge after 100 iterations (Figure 6 and Figure 7). The small learning rate is a large reason why the standardized data did not converge after 2000 iterations. It is possible that the raw data was able to converge because the range in data is so large, the gradients will also be much larger, allowing the model to converge faster as the rate of convergence is proportional to the gradients.

With the larger learning rate (0.1), the cases that were unable to converge with the small learning rate were able to converge without difficulty, as seen in Figure 8 and Figure 9.

An interesting effect is seen with the Raw data and the larger learning rate. The error actually diverges because the learning rate is too large. This is seen in Figure 10 and Figure 11, in the later case, the regression line is not able to be plotted on the same axes.

The results of training with `sklearn` are seen in Figure 12 and Figure 13. In both cases, the model was able to converge.

The values for the weights for each case is seen in Table I. The results for weights hardly varied between manually coding Gradient Descent and using `sklearn` for the cases when the descent converged.

Report due July 9, 2020

[1]Find the data on GitHub Here

TABLE I: Weights according to training cases

| **Training** | **Data** | **Slope, m** | **Intercept, b** | **Converged?** |
|---|---|---|---|---|
| 0.0001 | Standardized | -0.08 | -1.82E-17 | No |
| 0.0001 | Raw | 1.476 | 0.158 | Yes |
| 0.1 | Standardized | 0.773 | 2.727 | Yes |
| 0.1 | Raw | NaN | NaN | Diverged |
| sklearn | Standardized | 0.773 | 2.55E-16 | Yes |
| sklearn | Raw | 1.321 | 8.011 | Yes |

## Appendix A
## Results of Gradient Descent

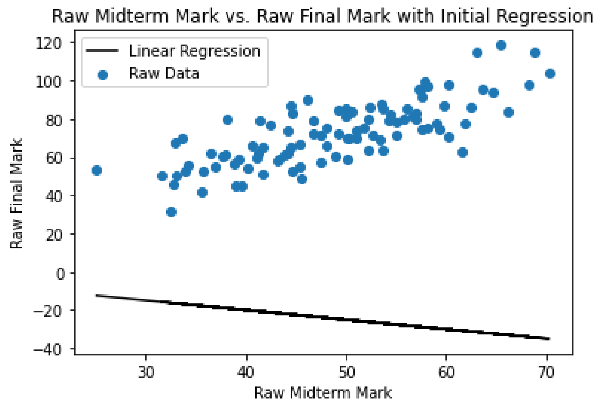

Fig. 2: Raw Mark Data with Initial Regression Line

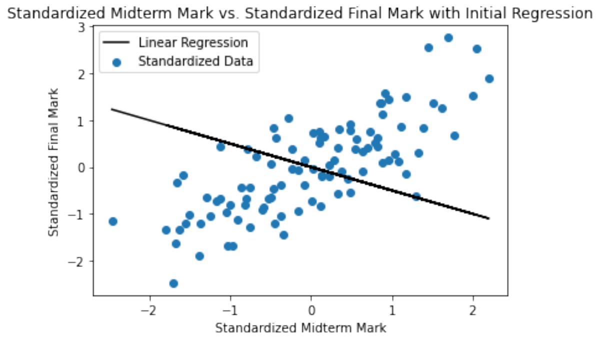

Fig. 3: Standardized Mark Data with Initial Regression Line

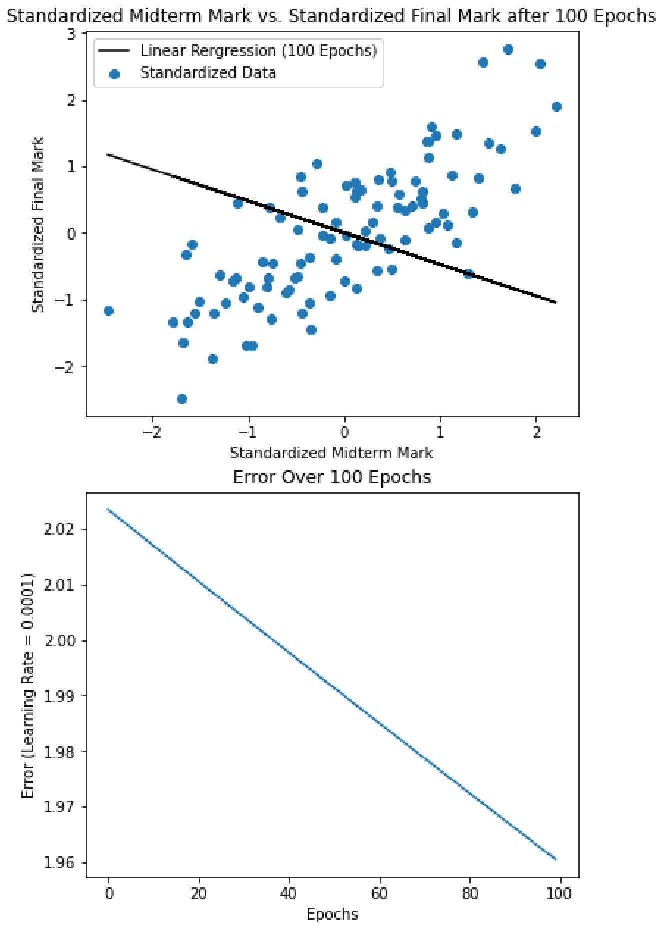

Fig. 4: Standardized Mark Data with Linear Regression (Top) and Error (Bottom) over 100 iterations with $\alpha = 0.0001$

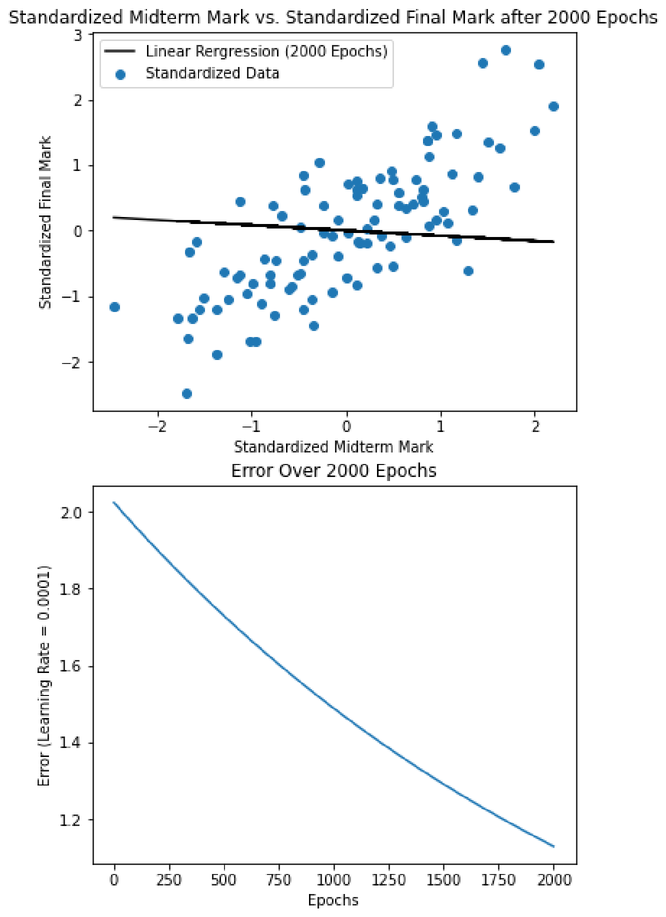

Fig. 5: Standardized Mark Data with Linear Regression (Top) and Error (Bottom) over 2000 iterations with $\alpha = 0.0001$

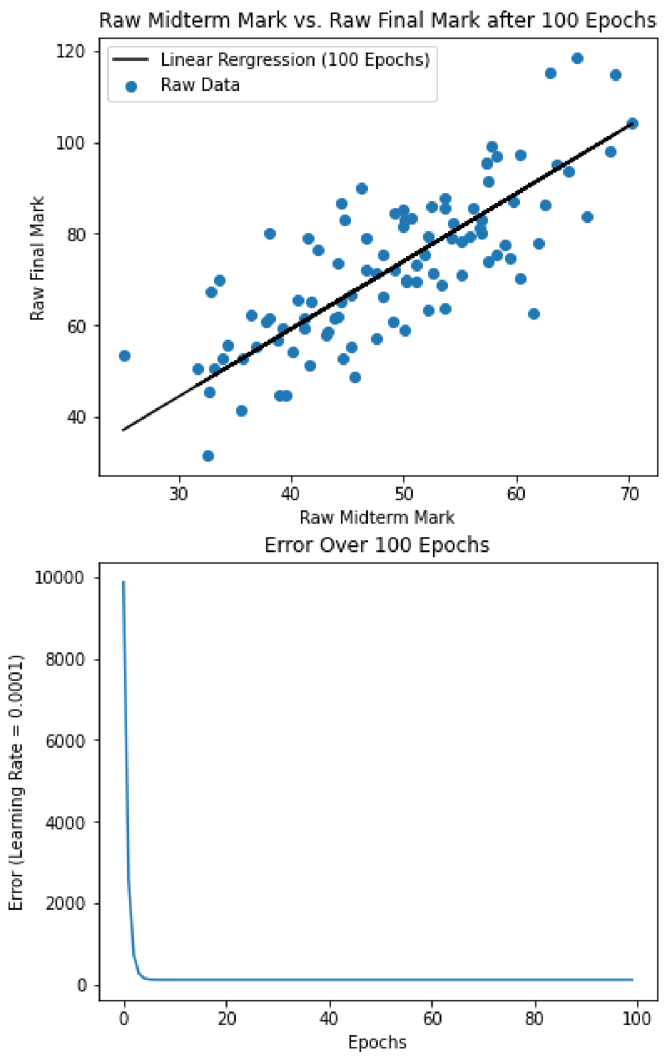

Fig. 6: Raw Mark Data with Linear Regression (Top) and Error (Bottom) over 100 iterations with $\alpha = 0.0001$

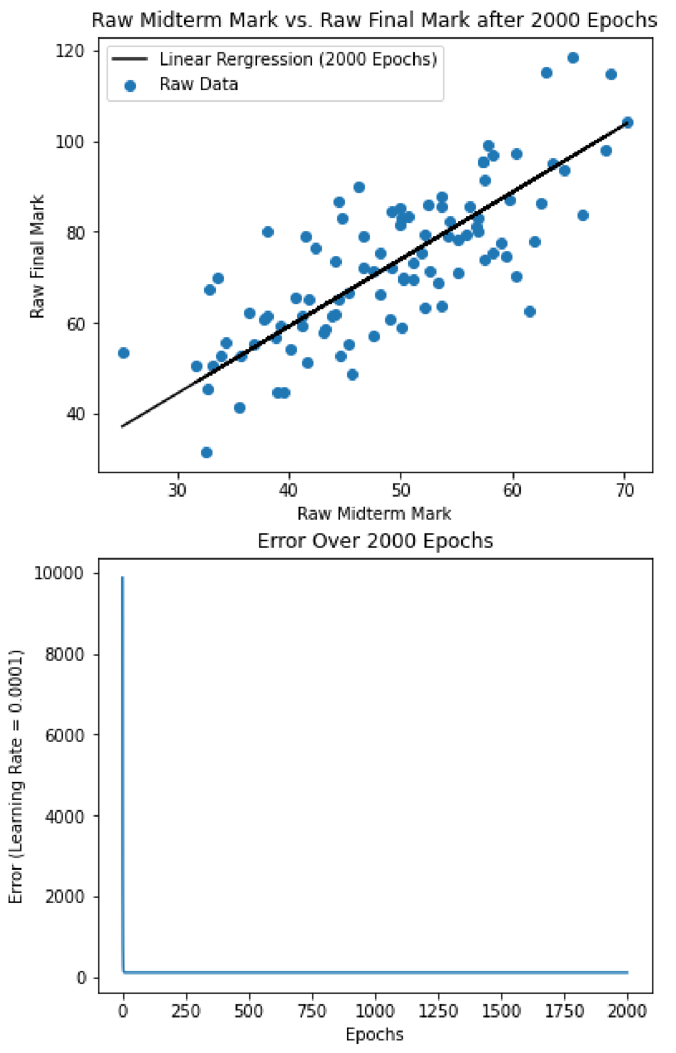

Fig. 7: Raw Mark Data with Linear Regression (Top) and Error (Bottom) over 2000 iterations with $\alpha = 0.0001$

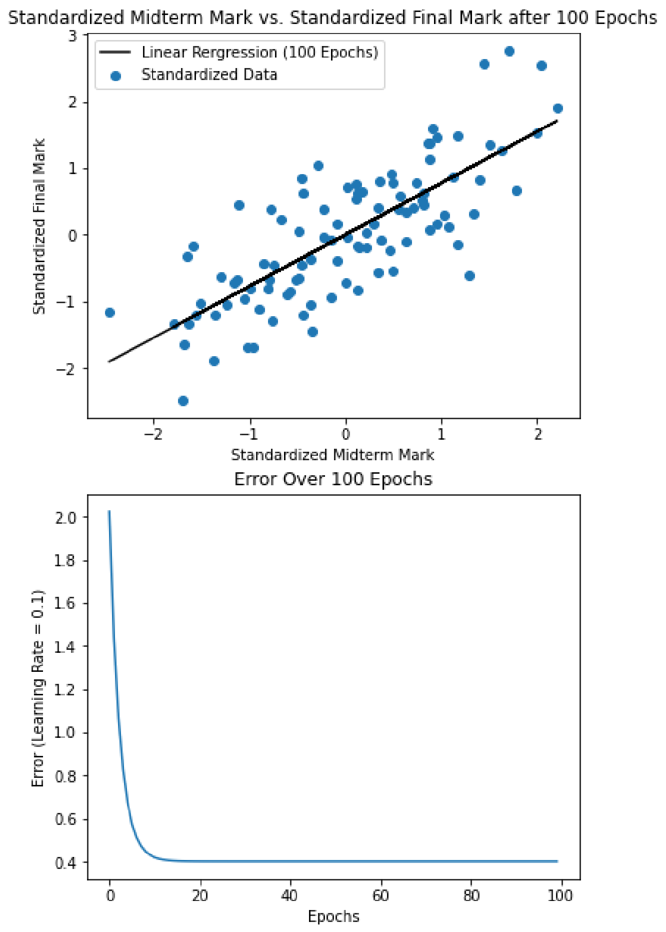

Fig. 8: Standardized Mark Data with Linear Regression (Top) and Error (Bottom) over 100 iterations with $\alpha = 0.1$

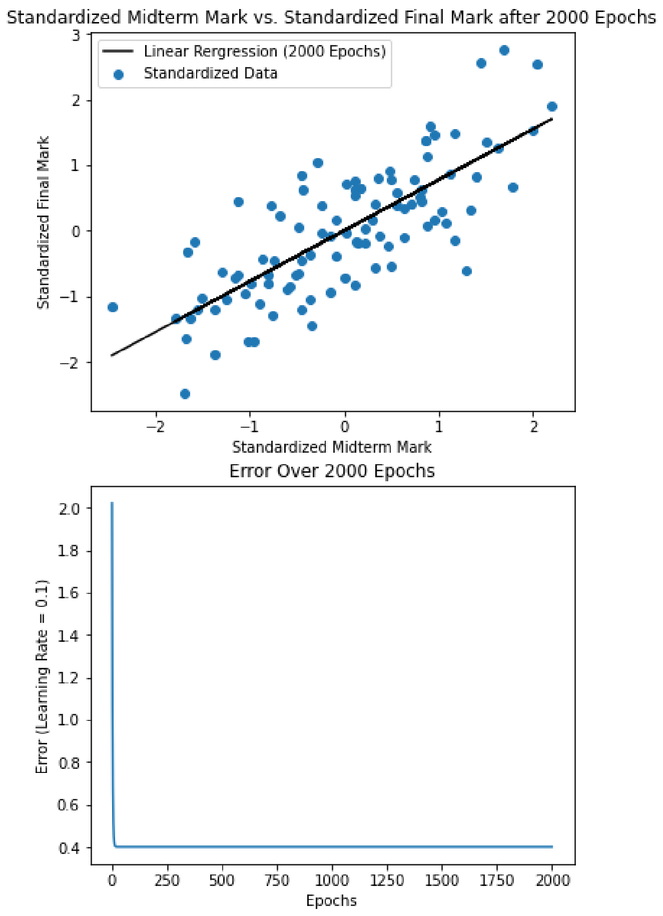

Fig. 9: Standardized Mark Data with Linear Regression (Top) and Error (Bottom) over 100 iterations with $\alpha = 0.1$

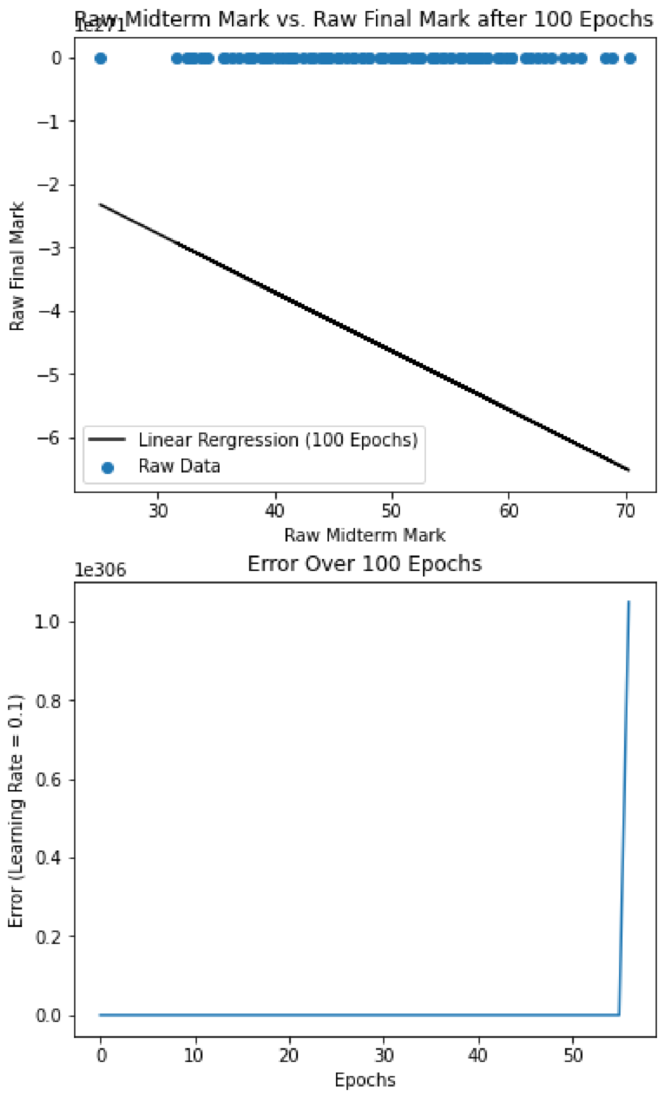

Fig. 10: Raw Mark Data with Linear Regression (Top) and Error (Bottom) over 100 iterations with $\alpha = 0.1$

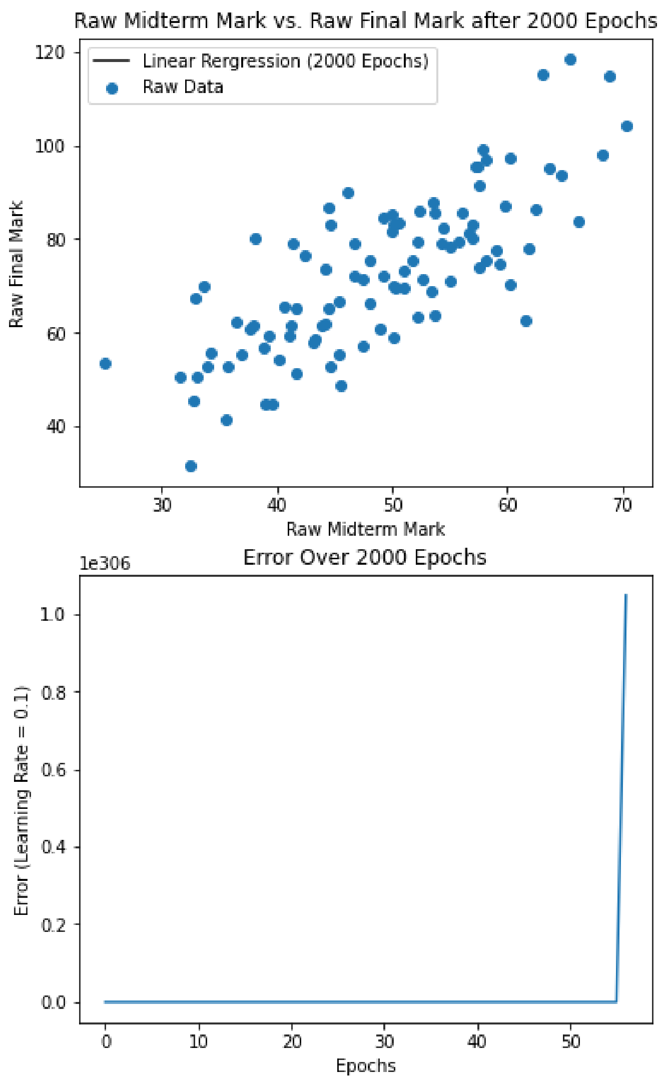

Fig. 11: Raw Mark Data with Linear Regression (Top) and Error (Bottom) over 2000 iterations with $\alpha = 0.1$

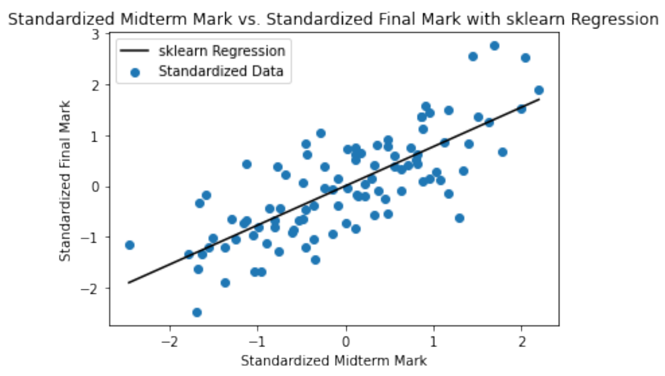

Fig. 12: Standardized Data with Linear Regression after training with `sklearn`

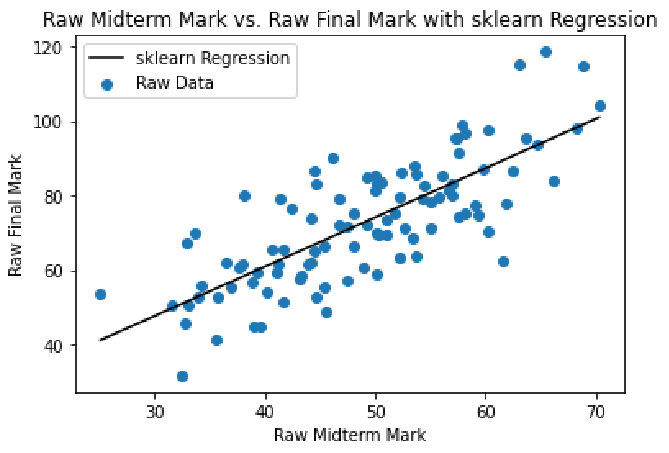

Fig. 13: Raw Data with Linear Regression after training with `sklearn`