

## Slope (`m`) in Linear Regression

In simple linear regression, the slope (`m`) of the best-fit line is calculated using the formula:

$$m = \frac{n \sum (X_i Y_i) - \sum X_i \sum Y_i}{n \sum (X_i^2) - (\sum X_i)^2}$$

Where:

- $X_i$  and  $Y_i$  are the individual data points of the independent and dependent variables, respectively.
- $n$  is the number of data points.

## Breaking Down `numerator\_m`

The numerator of the slope formula is calculated as:

$$\text{numerator\_m} = n \sum (X_i Y_i) - \sum X_i \sum Y_i$$

In Python code, this is written as:

```
numerator_m = (n_train * np.sum(X_train.flatten() * y_train)) - (np.sum(X_train) * np.sum(y_train))
```

- ``np.sum(X_train.flatten() * y_train)`` calculates the sum of the products of corresponding elements of ``X_train`` and ``y_train`` (i.e.,  $\sum (X_i Y_i)$ ).
- ``n_train * np.sum(X_train.flatten() * y_train)`` multiplies the above sum by the number of training data points  $n$ .
- ``np.sum(X_train)`` and ``np.sum(y_train)`` calculate the sums of the ``X_train`` and ``y_train`` arrays, respectively.
- The expression subtracts the product of the sums of ``X_train`` and ``y_train`` from the product of  $n$  and the sum of the products of corresponding elements.

## Breaking Down `denominator\_m`

The denominator of the slope formula is calculated as:

$$\text{denominator\_m} = n \sum (X_i^2) - (\sum X_i)^2$$

In Python code, this is written as:

```
denominator_m = (n_train * np.sum(X_train.flatten()**2)) - (np.sum(X_train)**2)
```

- ``np.sum(X_train.flatten()**2)`` calculates the sum of the squares of the ``X_train`` values (i.e.,  $\sum (X_i^2)$ ).
- ``n_train * np.sum(X_train.flatten()**2)`` multiplies this sum by the number of training data points  $n$ .
- ``np.sum(X_train)**2`` calculates the square of the sum of the ``X_train`` values.
- The expression subtracts the square of the sum of ``X_train`` from the product of  $n$  and the sum of the squares of ``X_train``.

The expression for ``b`` in your code calculates the **intercept** of the best-fit line in simple linear regression. The intercept (``b``) is the point where the line crosses the y-axis, and it represents the predicted value of ``y`` when ``x`` is 0.

## Intercept (``b``) in Linear Regression

In simple linear regression, the formula to calculate the intercept ``b`` is:

$$b = \bar{y} - m \cdot \bar{X}$$

Where:


- $\bar{y}$  is the mean of the dependent variable (target), i.e.,  $y$ .
- $\bar{X}$  is the mean of the independent variable (feature), i.e.,  $X$ .
- $m$  is the slope of the line.

This formula comes from rearranging the linear equation  $y = mX + b$  to solve for ``b``.

## Breaking Down the Code

In Python, this is implemented as:

python

 Copy code

```
b = (np.sum(y_train) - m * np.sum(X_train)) / n_train
```

This formula is an alternative way to compute the intercept using the sum of the data points, rather than their means. Let's break it down:

1. **Sum of `y_train`:** `np.sum(y_train)` calculates the sum of all the target values  $y_i$ . This is equivalent to  $n \cdot \bar{y}$ , where  $\bar{y}$  is the mean of `y_train` and  $n$  is the number of data points.
2. **Product of Slope and Sum of `X_train`:** `m * np.sum(X_train)` calculates the product of the slope  $m$  and the sum of the feature values  $X_i$ . This is equivalent to  $m \cdot n \cdot \bar{X}$ , where  $\bar{X}$  is the mean of `X_train`.
3. **Difference:** The difference `np.sum(y_train) - m * np.sum(X_train)` represents the total difference between the sum of the target values and the predicted values based on the slope.
4. **Divide by Number of Data Points:** Finally, dividing by `n_train` gives the average difference between the actual `y` values and the predicted values from the slope. This average difference is the intercept `b`.

## Intuition

The intercept `b` adjusts the regression line vertically so that it best fits the data. It's calculated based on how far off the predicted values (based solely on the slope) are from the actual values, on average.

- If `b` is positive, the line is shifted upwards.
- If `b` is negative, the line is shifted downwards.

This ensures that the line not only has the correct slope but also passes through the data in a way that minimizes the overall error.


Key metrics to evaluate the performance of your manually implemented simple linear regression model

1. **R-squared ( $R^2$ ):** A measure of how well your model explains the variance in the dependent variable (target).
2. **Mean Squared Error (MSE):** A measure of the average squared difference between the observed and predicted values.

Let's break down each component.

## 1. Total Sum of Squares (SST or `ss\_total`)

python

 Copy code

```
ss_total = np.sum((y_test - np.mean(y_test))**2)
```

- **Explanation:** `ss\_total` represents the **total variation** in the data. It measures how much the actual `y\_test` values deviate from the mean of `y\_test` (i.e., the average value of the target variable).


- **Formula:**

$$SS_{total} = \sum (y_i - \bar{y})^2$$

Where  $y_i$  are the actual values and  $\bar{y}$  is the mean of the actual values. This is the sum of the squared differences between each actual value and the mean value.

## 2. Residual Sum of Squares (SSR or `ss\_residual`)

python

 Copy code

```
ss_residual = np.sum((y_test - y_test_pred_manual)**2)
```

- **Explanation:** `ss\_residual` represents the **unexplained variation** or the sum of the squared differences between the actual values (`y\_test`) and the predicted values (`y\_test\_pred\_manual`). This measures how far off the predictions are from the actual values.


- **Formula:**

$$SS_{residual} = \sum (y_i - \hat{y}_i)^2$$

Where  $y_i$  are the actual values and  $\hat{y}_i$  are the predicted values from your model. This is the sum of the squared errors or residuals.

### 3. R-squared ( $R^2$ )

python

 Copy code

```
r2_manual = 1 - (ss_residual / ss_total)
```


- **Explanation:** `r2\_manual` is the **R-squared** value, which indicates the proportion of the variance in the dependent variable that is predictable from the independent variable.
- **Formula:**

$$R^2 = 1 - \frac{SS_{residual}}{SS_{total}}$$

- **Interpretation:**
  - If  $R^2 = 1$ , it means the model perfectly explains the variance in the data (i.e., all the data points lie perfectly on the regression line).
  - If  $R^2 = 0$ , it means the model does not explain any of the variance in the data (i.e., the model is no better than using the mean of the dependent variable as a prediction).
  - If  $R^2$  is negative, it means the model is worse than simply predicting the mean.

### 4. Mean Squared Error (MSE)

python

 Copy code

```
mse_manual = np.mean((y_test - y_test_pred_manual)**2)
```

- **Explanation:** `mse\_manual` is the **Mean Squared Error**, which is the average of the squared differences between the actual values (`y\_test`) and the predicted values (`y\_test\_pred\_manual`).
- **Formula:**

$$MSE = \frac{1}{n} \sum (y_i - \hat{y}_i)^2$$

- **Interpretation:**
  - A lower MSE indicates a better fit of the model to the data, as it means the predictions are closer to the actual values.
  - MSE is particularly useful because it penalizes larger errors more heavily, which helps in understanding how the model performs in predicting values that are far off from the actual ones.

## Summary

- **R-squared ( $R^2$ ):** Tells you how much of the variance in the target variable is explained by your model. It ranges from 0 to 1, with higher values indicating a better fit.
- **Mean Squared Error (MSE):** Tells you the average squared difference between the predicted and actual values. Lower values indicate better predictions.