📌 Simple Linear Regression – Mathematical Formulation (No Code)

---

1. Objective

We want to find a straight line:

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

Where:

- y^: predicted value (e.g., predicted salary)
- x: input feature (e.g., years of experience)
- m: slope of the line (rate of change)
- b: intercept (value of yy when x=0x = 0)

---

2. How to Find m (slope) and b (intercept)?

These values are **calculated mathematically** using **least squares minimization**, which reduces the total prediction error.

💡 ***Formulas:***

1. **Slope (m):**

$$m = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2}$$

2. **Intercept (b):**

$$b = \frac{\sum y - m\sum x}{n}$$

Where:

- n = total number of data points
- $\sum x\,y$ = sum of products of x and y values
- $\sum x^2$ = sum of squares of x values
- $\sum x$, $\sum y$ = sum of x and y values

Example:

Suppose we have small dataset like:

X: [1, 2, 3]

Y: [2, 4, 5]

We calculate:

- $\sum x = 6, \sum y = 11, \sum xy = 25, \sum x^2 = 14, n = 3$

Plug into formulas to compute m and b.

---

3. Why these formulas?

They're derived from minimizing the **Mean Squared Error (MSE)**:

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

We differentiate the error formula with respect to m and b to get optimal values (calculus-based derivation, skipped here).

---

4. Coding from Scratch

Though we're not covering code here, the idea is:

- Loop through the dataset to compute the above summations.
- Apply the formulas to calculate m and b.
- Use them to predict y^ for any new x.

5. How the Line is Used

Once we have:

- m=1.5 , b = 2

Then the line becomes:

$$\hat{y} = 1.5x + 2$$

So:

- For $x = 4$, predicted $y = 1.5(4) + 2 = 8$

Summary Table

Term	Meaning
m	Slope: Controls the steepness of the line
b	Intercept: Where line crosses y-axis
Least Squares	Technique to minimize total squared errors
y^=mx+b	Prediction formula
Goal	Fit best line to minimize prediction errors

📘 Simple Linear Regression: Step-by-Step Mathematical Derivation

🎯 Goal:

We want to fit a **straight line**:

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

Where:

- y^: Predicted value
- x: Input feature
- m: Slope of the line
- b: Intercept (value of y when x=0)

Step 1: Dataset Assumption

Suppose we have nn data points:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

We want a line y^= mx + b that best fits this data.

Step 2: Define the Error Function

We want to minimize the difference between the actual yi and the predicted y^i.
So the error for each point is:

$$\text{Error}_i = y_i - \hat{y}_i = y_i - (mx_i + b)$$

To avoid negative errors canceling out, we square them.

The **total error** (called the *Sum of Squared Errors*, SSE):

$$SSE = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

This is the function we want to minimize.

Step 3: Use Calculus to Minimize Error

To find the best mm and bb, we take **partial derivatives** of the SSE and set them to **zero**.

Partial derivative with respect to b:

$$\frac{\partial}{\partial b} \sum (y_i - mx_i - b)^2 = -2 \sum (y_i - mx_i - b)$$

Set it to 0:

$$\sum (y_i - mx_i - b) = 0$$

$$\sum y_i - m \sum x_i - nb = 0$$

Solve for b:

$$b = \frac{\sum y_i - m \sum x_i}{n}$$

Partial derivative with respect to m:

$$\frac{\partial}{\partial m} \sum (y_i - mx_i - b)^2 = -2 \sum x_i (y_i - mx_i - b)$$

Set it to 0:

$$\sum x_i (y_i - mx_i - b) = 0$$

$$\sum x_i y_i - m \sum x_i^2 - b \sum x_i = 0$$

Plug in equation (1) into this and solve — or we derive a closed-form solution directly (next step).

Step 4: Final Formulas for m and b

These formulas are derived from solving the above equations:

Slope (m):

$$m = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2}$$

Intercept (b):

$$b = \frac{\sum y_i - m \sum x_i}{n}$$

These are called the **normal equations** for simple linear regression.

Step 5: Predicting New Values

Once you have m and b, the predicted value for a new x is:

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

Summary Table

Quantity	Meaning
x_i	Input (independent variable)
y_i	Output (dependent variable)
m	Slope of line (rate of change)
b	Intercept (value at x=0x = 0)
SSE	Sum of squared prediction errors
Goal	Minimize SSE using calculus
