Linear Regression

Linear regression is a *supervised* machine-learning algorithm that models the relationship between one or more independent variables and a continuous dependent variable. It's one of the first tools every data-scientist masters because the core idea—"fit a straight line that captures the trend"—is both intuitive and mathematically elegant.

Simple Linear Regression

Simple linear regression is the most basic form of linear regression. It involves one input feature (also known as an independent variable) and one output feature (or dependent variable).

Example: College Student Data

Let's consider a dataset of college students where we aim to predict their package (output column) based on their CGPA (input column).

cgpa	package
6.66	3.01

Here, CGPA is the input column, and Package is the output column. Our goal is to predict the package based on the CGPA.
The fundamental equation for simple linear regression is :

$y = \beta_o + \beta_1 x$

where:

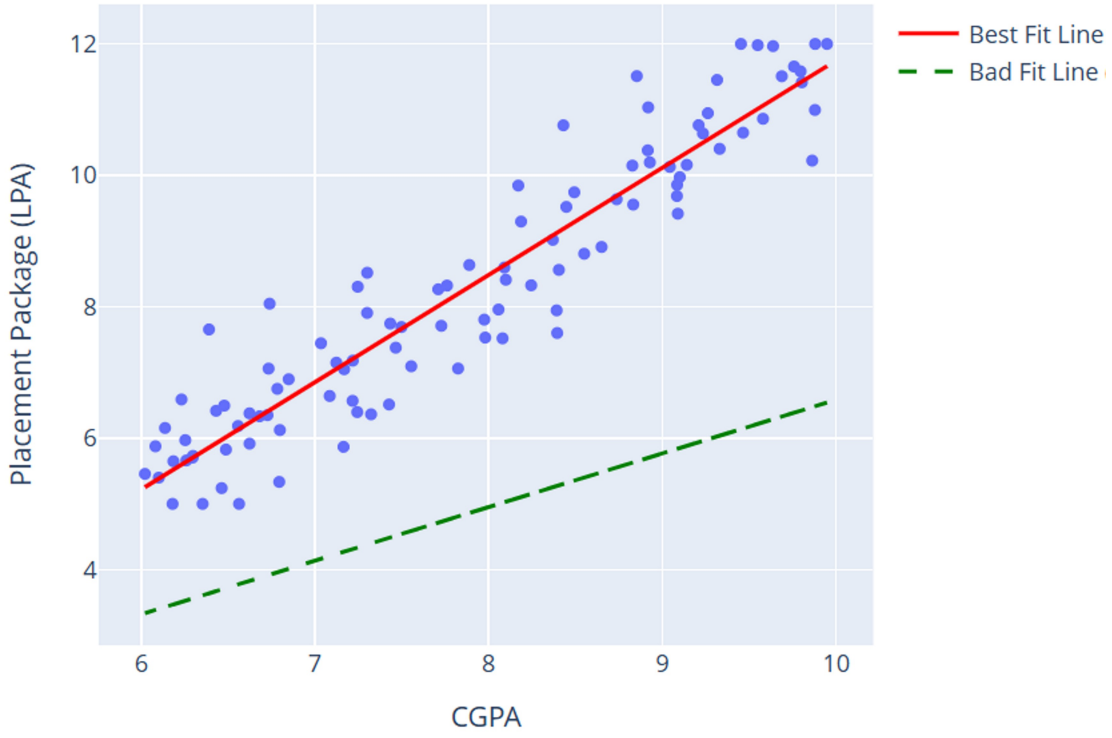
- y is the output (dependent variable)
- x is the input (independent variable)
- β_o is the intercept (the value of y when x is 0)
- β_1 is the slope (the change in y for a one-unit change in x)

Let's consider a more extensive hypothetical dataset of college students:

cgpa	package
7.1	3.5
4.7	1.2

8.9	4.2
8.1	3.9
.	.
.	.

The first step in analyzing such data is to visualize it by plotting the data points.



As observed from the plot, the data exhibits a generally linear trend, but it's not perfectly linear. Real-world data is inherently complex and often influenced by various unobserved factors, leading to what is known as stochastic error or random variability. These errors can be analyzed and modeled mathematically.

The Equation of a Line and Best Fit Line

The general equation for a straight line is :

$y = mx + b$

where :

- m represents the slope of the line
- b represents the y-intercept

In linear regression, the best-fit line (also known as the regression line) represents the strongest linear relationship between the independent variable (x) and the dependent variable (y) within the dataset.

The primary objective of linear regression is to find the optimal parameters (β_o and β_1 in our previous equation, or b and m in the general line equation) for this line. These parameters are chosen to minimize the difference (or error) between the predicted y values (from the line) and the actual observed y values in the dataset.

The best-fit line can also be represented by the linear equation:

$y = mx + b$

Where:

- y is the dependent variable (response)
- x is the independent variable (predictor)
- m is the slope of the line
- b is the y-intercept

How to Find m and b (or β_o and β_1) ?

There are generally two main approaches to determine the optimal values for the slope (m or β_1) and intercept (b or β_o) in linear regression: Closed-Form Solutions and Iterative (Non-Closed-Form) Solutions.

1. Closed-Form Solution: This approach involves directly calculating the values of m and b using a predefined mathematical formula. The most common closed-form method for simple linear regression is the Ordinary Least Squares (OLS) method.
2. Iterative (Non-Closed-Form) Solution: This approach uses approximation techniques, starting with initial guesses for m and b and iteratively refining them to minimize the error. The most prominent iterative technique is Gradient Descent.

When to Use Which ?

- Ordinary Least Squares (OLS): OLS is straightforward and computationally efficient for datasets with a relatively small number of features (low-dimensional data). It provides an exact solution.
- Gradient Descent: When dealing with high-dimensional data (a large number of input features), finding a closed-form solution can become

computationally very complex or even infeasible. In such scenarios, gradient descent becomes the preferred method, as it can efficiently find approximate solutions by iteratively moving towards the minimum error.

Implementation in Scikit-learn :

- The LinearRegression class in Scikit-learn, a popular Python machine learning library, primarily uses the Ordinary Least Squares (OLS) method to determine the regression coefficients.
- The SGDRegressor (Stochastic Gradient Descent Regressor) in Scikit-learn, on the other hand, utilizes Gradient Descent to find the values of m and b. This is particularly useful for large datasets where OLS might be too slow.

Formulas to Find the Values of m and b (OLS Method)

For the Ordinary Least Squares (OLS) method, the formulas to calculate the slope (m) and intercept (b) are derived to minimize the sum of squared errors.

The formula for the slope (m) is:

$$m = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

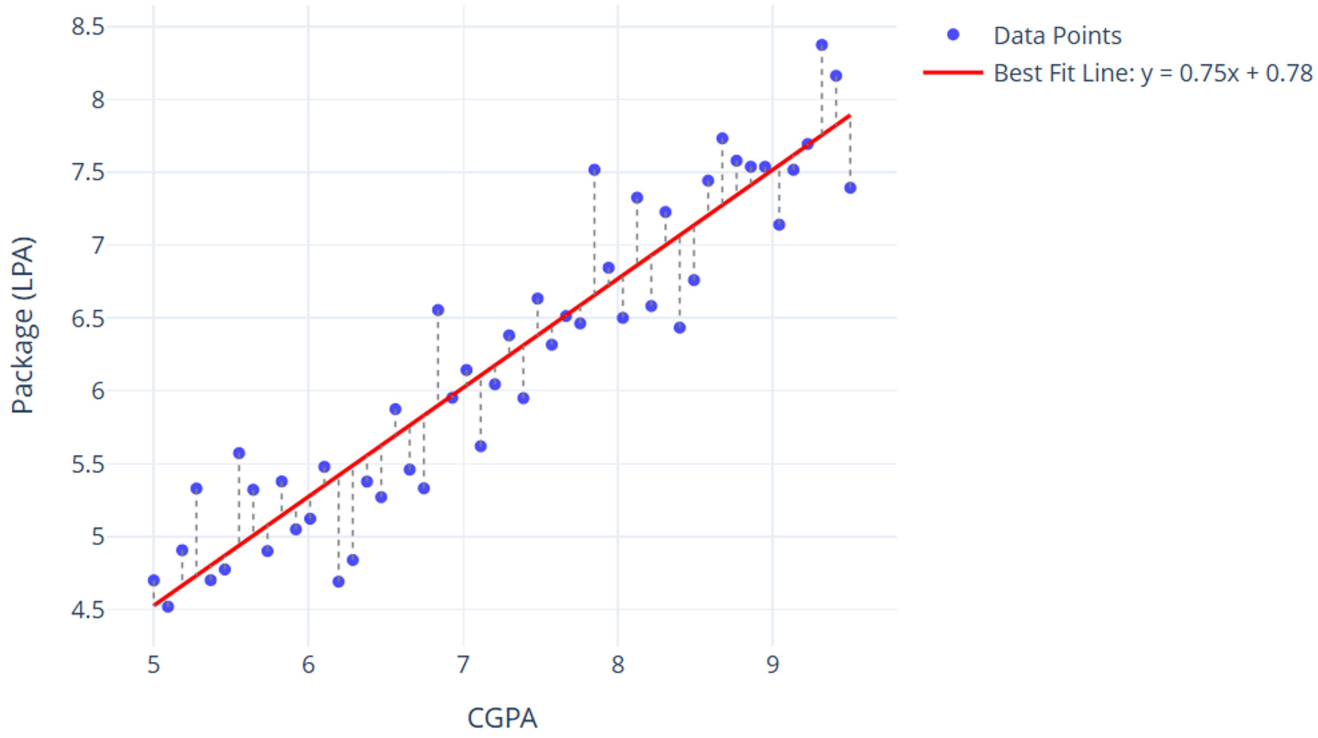
And the formula for the intercept (b) is:

$$b = \bar{y} - m\bar{x}$$

Where:

- n is the total number of data points.
- x_i represents the individual CGPA values (input).
- y_i represents the individual Package values (output).
- $\bar{x}$ is the mean (average) of the CGPA values.
- $\bar{y}$ is the mean (average) of the Package values.

Error (Loss Function)



In linear regression, we define the error as the difference between the actual observed value and the value predicted by our regression line. For each data point i, the error is denoted as d_i .

$d_i = (y_i - \hat{y}_i)$

Where:

- y_i is the actual observed value for the i-th data point.
- $\hat{y}_i$ is the predicted value for the i-th data point from the regression line.

The total error, or the loss function (often denoted as E), is a measure of how well our line fits the data. Ideally, we want this error to be as small as possible. Initially, one might think of summing the individual errors:

$E = d_1 + d_2 + d_3 + \ldots + d_n$

However, simply summing these errors can be problematic because positive and negative errors (points above and below the line) would cancel each other out, potentially leading to a misleadingly small total error even if the line fits poorly. To address this, and to emphasize larger errors, we square each individual error and then sum them. This is known as the Sum of Squared Errors (SSE) or Residual Sum of Squares (RSS):

$E = d_1^2 + d_2^2 + d_3^2 + \ldots + d_n^2$

Or, more compactly:

$E = \sum_{i=1}^{n} d_i^2$

This Sum of Squared Errors (SSE) is our loss function.

Why Squared Errors ?

- Avoids Cancellation: Squaring ensures that all errors contribute positively to the total sum, preventing positive and negative deviations from canceling out.
- Penalizes Large Errors More: Squaring disproportionately penalizes larger errors. For instance, an error of 2 becomes 4 when squared, while an error of 4 becomes 16. This means the model works harder to reduce larger errors. This also helps in making the model less sensitive to outliers compared to using absolute errors.
- Differentiability: The squared error function is continuous and differentiable, which is crucial for using calculus-based optimization methods (like those used to derive the OLS formulas or in gradient descent) to find the minimum error. If we used absolute values ($|d_i|$), the function would not be differentiable at zero, making it harder to find the minimum mathematically.

The core objective of simple linear regression (and more broadly, Ordinary Least Squares) is to find the specific values of m and b that minimize this Sum of Squared Errors (E).

Deriving the Formulas for m and b (Minimizing the Loss Function)

Our goal in simple linear regression is to find the values of m(slope) and b(y-intercept) that minimize the Sum of Squared Errors (SSE), our loss function, E.

The predicted value ($\hat{y}_i$) for any given x_i is given by :
$\hat{y}_i = mx_i + b$
Therefore, our loss function, E, can be expressed as:
$E = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
$E = \sum_{i=1}^{n}(y_i - (mx_i + b))^2$

To find the minimum value of E with respect to m and b, we use multivariable calculus. We take the partial derivative of E with respect to b and set it to zero, and similarly for m. This is because, at the minimum point of a function, its slope

(derivative) is zero.

Derivation of b :

$\frac{\partial E}{\partial b} = \frac{\partial}{\partial b}\sum_{i=1}^{n}(y_i - mx_i - b)^2 = 0$

$$\sum_{i=1}^{n}\frac{\partial}{\partial b}(y_i - mx_i - b)^2 = 0$$

$$\sum_{i=1}^{n} -2\,(y_i - mx_i - b) = 0$$

$$\sum_{i=1}^{n}(y_i - mx_i - b) = 0$$

$$\frac{\sum_{i=1}^{n}(y_i)}{n} - \frac{\sum_{i=1}^{n}(mx_i)}{n} - \frac{\sum_{i=1}^{n}(b)}{n} = 0$$

$$\bar{y} - m\bar{x} - b = 0$$

b = $\bar{y} - m\bar{x}$

Derivation of m :

$E = \sum_{i=1}^{n}(y_i - mx_i - \bar{y} + m\bar{x})^2 = 0$

$\frac{\partial E}{\partial m} = \sum_{i=1}^{n}\frac{\partial}{\partial m}(y_i - mx_i - \bar{y} + m\bar{x})^2 = 0$

$$\sum_{i=1}^{n} 2\,(\,y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) = 0$$

$$\sum_{i=1}^{n} -2\,(\,y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) = 0$$

$$\sum_{i=1}^{n}(\,y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) = 0$$

$$\sum_{i=1}^{n}[(y_i - \bar{y}) - m(x_i - \bar{x})](x_i - \bar{x}) = 0$$

$$\sum_{i=1}^{n}[(y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2] = 0$$

$$\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}) = m\sum_{i=1}^{n}(x_i - \bar{x})^2$$

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Multiple Linear Regression

Multiple Linear Regression is an extension of simple linear regression. Unlike simple linear regression, which involves one input feature (independent variable) and one output (dependent variable), multiple linear regression involves more than one input feature and still predicts a single output.
We use it when there are multiple predictors (input variables) influencing the target.

Example Dataset

Suppose we have data for 1000 students :

CGPA	IQ	Placement
8	80	8
9	90	9
5	120	15
...	...	...

In this case, we have two input features: CGPA and IQ, and one target variable: Placement.
Since we have multiple input columns, we use Multiple Linear Regression.
The model takes CGPA and IQ as inputs and predicts Placement.

Mathematical Equation

The equation of the regression line becomes :

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

where :

- x_1 -> CGPA
- x_2 -> IQ
- β_0 -> Intercept (bias term)
- β_1 , β_2 -> slopes (weights)

If we have **n** input features, the equation generalizes to :
$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$

Geometric Interpretation

- In simple linear regression, the model fits a line in 2D space.
- In multiple linear regression with 2 features, it fits a plane in 3D space.
- In higher dimensions (n > 2), the model fits a hyperplane in n-dimensional space.

Objective of the Model

The goal of multiple linear regression is to find the optimal values of:
$\beta_0 , \beta_1 , \beta_2 , \ldots , \beta_n$
These are the coefficients (weights) of the model.
The algorithm tries to fit a hyperplane that minimizes the error between the predicted and actual output. In other words, it tries to find a surface that is as close as possible to all data points in high-dimensional space.

Mathematical Formulation of Multiple Linear Regression

Let's consider data from three students of a college. Our goal is to apply Multiple Linear Regression to predict placement based on CGPA and IQ.

$cgpa(x_1)$	$iq(x_2)$	$placement$
$8\ x_{11}$	$80\ x_{12}$	8
$7\ x_{21}$	$70\ x_{22}$	7
$5\ x_{31}$	$120\ x_{32}$	15

Assume we know the coefficient values $\beta_0 , \beta_1 , \beta_2$.
We want to predict:
$\hat{y}_1 = ?$ $\hat{y}_2 = ?$ $\hat{y}_3 = ?$

To calculate the predicted outputs ($\hat{y}_i$):
$\hat{y}_1 = \beta_0 + \beta_1.8 + \beta_2.80$
$\hat{y}_2 = \beta_0 + \beta_1.7 + \beta_2.70$
$\hat{y}_3 = \beta_0 + \beta_1.5 + \beta_2.120$

Or more generally, using variable notation:
$\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12}$
$\hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22}$
$\hat{y}_3 = \beta_0 + \beta_1 x_{31} + \beta_2 x_{32}$

Now, suppose we have m input features and n students (i.e., n training samples).
Each prediction becomes:
$\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots + \beta_m x_{im}$

So for all students:
$\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_m x_{1m}$
$\hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_m x_{2m}$
$\hat{y}_3 = \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + \cdots + \beta_m x_{3m}$
$\vdots$
$\hat{y}_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_m x_{nm}$

Matrix Form (Compact Representation)

Let's express all predictions in matrix form:

Prediction vector $\hat{Y}$:

$$\hat{y}_{n*1} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ . \\ \hat{y}_n \end{bmatrix}_{n*1} = \begin{bmatrix} \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_m x_{1m} \\ \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_m x_{2m} \\ \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + \cdots + \beta_m x_{3m} \\ . \\ \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_m x_{nm} \end{bmatrix}_{n*1}$$

Matrix Representation :
$\hat{Y} = X\beta$
Where:

$$X_{n*(m+1)} = \begin{bmatrix} 1 + x_{11} + x_{12} + \cdots + x_{1m} \\ 1 + x_{21} + x_{22} + \cdots + x_{2m} \\ 1 + x_{31} + x_{32} + \cdots + x_{3m} \\ . \\ 1 + x_{n1} + x_{n2} + \cdots + x_{nm} \end{bmatrix} \quad \beta_{(m+1)*1} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ . \\ \beta_m \end{bmatrix}$$

So,
$\hat{Y} = X\beta$

Resulting in:
$\hat{y}_{n*1} = X_{n*(m+1)} \cdot \beta_{(m+1)*1}$

Loss Function in Multiple Linear Regression

In regression, our aim is to find a best-fit line or hyperplane that minimizes the distance between the actual values and predicted values.
We want to minimize the sum of squared distances (errors):
$d_1^2 + d_2^2 + d_3^2 + \cdots + d_n^2$

Objective

In Multiple Linear Regression, we minimize the Sum of Squared Errors (SSE) — the total squared distance between actual and predicted values.
This is known as the Loss Function:

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

Vector & Matrix Representation

$$y_i = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ y_n \end{bmatrix}_{n*1} \qquad \hat{y}_i = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ . \\ \hat{y}_n \end{bmatrix}_{n*1}$$

$$e = y_i - \hat{y}_i = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ y_3 - \hat{y}_3 \\ . \quad . \\ y_n - \hat{y}_n \end{bmatrix}$$

$$e^T e = [y_1 - \hat{y}_1 \;\; y_2 - \hat{y}_2 \ldots\ldots y_n - \hat{y}_n]_{1*n} \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ y_3 - \hat{y}_3 \\ . \quad . \\ y_n - \hat{y}_n \end{bmatrix}_{n*1}$$

$e^T \mathrm{e} = (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \ldots\ldots + (y_n - \hat{y}_n)^2$

$$e^T e = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$E = e^T \mathrm{e}$

Expanded Matrix Form

Let's now expand $E = e^T e$ using matrix algebra.
$e = y - \hat{y} \Rightarrow E = (y - \hat{y})^T (y - \hat{y})$
Now expand using distributive property:
$E = y^T y - y^T \hat{y} - \hat{y}^T y + \hat{y}^T \hat{y}$
Since $y^T \hat{y}$ and $\hat{y}^T y$ are scalars and equal:
$E = y^T y - 2y^T \hat{y} + \hat{y}^T \hat{y}$

Proving $y^T \hat{y} = \hat{y}^T y$

Now we will prove that the two expressions below are equal:
$y^T \hat{y} = \hat{y}^T y$

Let's assume:

- $y = A$
- $\hat{y} = B$

So we want to prove:
$A^T B = (A^T B)^T$ OR $C = C^T$

This means we want to prove $A^T B$ is a symmetric matrix, i.e., its transpose is equal to itself.

Symmetric Matrix Example:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)^T$$

Now let's prove that $y^T \hat{y}$ is symmetric :
We know :
$\hat{y} = X\beta \Rightarrow y^T \hat{y} = y^T X\beta$

Let's break down the dimensions:

- X is an $n \times (m+1)$ matrix (with intercept column)

- β is a (m+1)×1 column vector
- y^T is a 1*n row vector

so $y^T X\beta$ results in a scalar.
And since the transpose of a scalar is the same scalar, we get :
$y^T\hat{y} = \hat{y}^T y$ proven

Loss Function in Terms of β

We had derived the error function as:
$E = (y - \hat{y})^T(y - \hat{y})$

Now using the identity $\hat{y} = X\beta$, we substitute :
$E = y^T y - 2y^T X\beta + \beta^T X^T X\beta$

Interpretation

- E(β) is a function of β
- As we change the values of β , the error E also changes
- Input X and output y are fixed, only β is variable

Goal : We want to find the values of β that minimize the error function E(β).

Relation of Loss Function with Coefficients

To minimize the loss function, we differentiate it with respect to β and set it to zero:

$$\frac{\partial E}{\partial \boldsymbol{\beta}} = 0$$

We want to find the value of β where the loss is minimized (i.e., the slope is zero at the minimum point).

Differentiating the Loss Function

From earlier:
$E(\beta) = y^T y - 2y^T X\beta + \beta^T X^T X\beta$

Now differentiating w.r.t. β :

$$\frac{\partial E}{\partial \beta} = 0 - 2y^T X + 2\beta^T X^T X$$

Matrix Differentiation Rule Used:

$$\frac{\partial}{\partial \boldsymbol{\beta}} \left(\boldsymbol{\beta}^T A \boldsymbol{\beta} \right) = 2A\boldsymbol{\beta}, \quad \text{if } A \text{ is symmetric}$$

in our case:

- $A = X^TX$, which is symmetric because :
  $(X^TX)^T = X^T(X^T)^T = X^TX$

Solving for β

Set the derivative to zero:
$0 = -2y^TX + 2\beta^TX^TX$

Divide both sides by 2:
$\beta^TX^TX = y^TX$

Multiply both sides by $(X^TX)^{-1}$:
$\beta^T = y^TX(X^TX)^{-1}$

Now transpose both sides:
$(\beta^T)^T = [y^TX(X^TX)^{-1}]^T$

Apply transpose rule:
$\beta = ((X^TX)^{-1})^T X^TY$

Since the transpose of an inverse symmetric matrix is the same:
$\beta = (X^TX)^{-1} X^TY$

Shape Analysis of Matrices in OLS

- shape of β = (m+1)*1

Understanding Shape of Each Term:

- X^T is of shape (m+1)*n
- X is of shape n*(m+1)
- so , X^TX is of shape (m+1)*(m+1)
- Therefore $(X^TX)^{-1}$ is also of shape (m+1)*(m+1)

Now :

- X^TY is of shape (m+1)*1

- so , $(X^TX)^{-1}\,X^TY$ is of shape (m+1)*1

Thus,
$\beta = (X^TX)^{-1}\,X^TY$
remains consistent in shape, confirming it's valid.

Proof: $(X^TX)^{-1}$ is Symmetric

Let:
$A = X^TX$ (a square matrix)

we need to prove :
$(A^{-1})^T = A^{-1}$

This would imply :
$[(X^TX)^{-1}]^T = (X^TX)^{-1}$

Proof:

given:
$AA^{-1} = I \Rightarrow (AA^{-1})^T = I^T = I$
$(A^{-1})^TA^T = I$

since $A = A^T$ (because X^TX is symmetric), this becomes :
$(A^{-1})^TA = I$

Now pre-multiply both sides with A^{-1} :
$A^{-1}(A^{-1})^TA = A^{-1} \Rightarrow (A^{-1})^T = A^{-1}$

Hence proved , $(X^TX)^{-1}$ is symmetric .

Final Normal Equation (OLS Formula)

$\beta = (X^TX)^{-1}\,X^TY$

- This is the Ordinary Least Squares (OLS) solution for multiple linear regression.
- It minimizes the sum of squared errors between actual and predicted values.
- β contains the intercept and coefficients.