Linear Regression

What is Linear Regression?

Linear Regression is a **supervised learning algorithm** used for **predicting a continuous (numeric) value** based on one or more input features.

In simple terms:

It finds the **best-fitting straight line** (or hyperplane) through your data to model the relationship between **input (X)** and **output (Y)**.

📈 Example

Let's say you want to predict someone's **salary** based on their **years of experience**.

Experience (Years)	Salary ($)
1	25,000
2	30,000
3	35,000
4	40,000

Linear regression tries to fit a line like this:

Salary=m×Experience+bSalary = m \times Experience + bSalary=m×Experience+b

where:

- m = slope (how much salary increases per year)
- b = intercept (base salary when experience = 0)

So, if you find
m = 5000 and b = 20000,
then:

Salary=5000×Experience+20000Salary = 5000 \times Experience + 20000Salary=5000×Experience+20000

⚙️ Formula (for one feature)

y=w1x+by = w_1x + by=w1x+b

- x → input variable (feature)
- y → predicted output
- w_1 → weight (slope of the line)
- b → bias (intercept)

For multiple features (called **Multiple Linear Regression**):

y=w1x1+w2x2+...+wnxn+by = w_1x_1 + w_2x_2 + ... + w_nx_n + by=w1x1+w2x2+...+wnxn +b

---

🎯 Goal

The goal is to **find the best values of** w_1, w_2, ..., b that **minimize the error** between predicted and actual values.

We use a **loss function** called **Mean Squared Error (MSE)**:

MSE=1n∑i=1n(yi−y^i)2MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2MSE=n1i=1∑n(yi −y^i)2

where

- y_i → actual value
- $\hat{y}_i$ → predicted value

The algorithm tries to **minimize this error** (usually using Gradient Descent).

💻 In Python (using sklearn)

```
from sklearn.linear_model import LinearRegression


# Example data
X = [[1], [2], [3], [4]]  # experience
y = [25000, 30000, 35000, 40000]  # salary


# Create model
model = LinearRegression()
```

```
# Train model
model.fit(X, y)


# Predict salary for 5 years of experience
print(model.predict([[5]]))  # Output ≈ [45000]
```

🔍 Key Points

Concept	Description
Type	Supervised Learning
Output	Continuous values (e.g. price, temperature)
Equation	Straight line (y = mx + b)
Loss Function	Mean Squared Error (MSE)
Optimization	Gradient Descent or Normal Equation
Example Use Cases	Predicting prices, sales, growth, trends

🧩 Real-world Examples

- Predicting **house prices** based on area, bedrooms, location.
- Estimating **sales** from advertising spend.
- Forecasting **temperature** or **stock prices** (basic models).