

The Two Pillars of Supervised Learning: An Intuitive Primer on Linear and Logistic Regression

Aksel Aghajanyan

Technical Report — Knowledge Base Archive

Abstract

Machine learning textbooks often introduce Linear and Logistic Regression using intimidating matrix calculus, vectors, and proofs. In reality, the core ideas behind both algorithms are simple, elegant, and grounded in common sense. This guide explains both models using 95% plain English, real-world analogies, and only 5% essential math. We explore how Linear Regression predicts continuous numbers (like house prices or pizza delivery times) by finding the best-fitting line, why straight lines fail when answering simple Yes/No questions, and how Logistic Regression uses an S-shaped curve to turn numbers into honest probabilities (0% to 100%). By the end of this paper, any beginner will understand how these models think, learn from mistakes, and make decisions.

I. INTRODUCTION: THE TWO BIG QUESTIONS IN MACHINE LEARNING

Every machine learning problem you encounter in the real world usually asks one of two simple questions:

1) “How much?” or “How many?”

Examples: How many dollars will this house sell for? How many minutes until the food arrives? What will the temperature be tomorrow?

→ This is called **Regression**. Our answer is an ordinary continuous number.

2) “Which one?” or “Yes or No?”

Examples: Is this incoming email Spam or Not Spam? Will this customer cancel their subscription (Yes/No)? Is this medical test positive or negative?

→ This is called **Classification**. Our answer is a category or a probability.

To solve these two questions, data scientists turn to two trusted tools:

- **Linear Regression** is the master tool for answering “How much?”
- **Logistic Regression** is the master tool for answering “Yes or No?”

Let us look at each one step by step, keeping formulas to an absolute minimum.

II. PART I: LINEAR REGRESSION (PREDICTING A NUMBER)

A. The Everyday Analogy: Estimating Taxi Fares

Imagine you get into a taxi cab. Before the car even rolls an inch, the meter starts with a fixed base fare of \$3.00. Then, for every kilometer traveled, the driver adds \$2.00 to the bill.

If you travel 5 kilometers, your brain instantly calculates:

$$\text{Cost} = \$3.00 + (\$2.00 \times 5) = \$13.00$$

Without knowing it, you just performed **Linear Regression**!

B. The Only Formula You Need (The 5% Math)

In data science, we write this exact same taxi calculation using three letters:

$$\hat{y} = w \cdot x + b \quad (1)$$

Here is the entire formula translated into plain English:

- x (**The Feature**): What you already know. In our taxi ride, this is the *distance* (5 km). In real estate, this might be the size of an apartment.
- w (**The Weight or Slope**): The *multiplier knob*. It tells the computer how sensitive the outcome is to changes in x . For our taxi, $w = 2$ dollars per kilometer.
- b (**The Bias or Starting Line**): The *baseline anchor*. What the prediction defaults to when $x = 0$. For our taxi, $b = 3$ dollars base fare.
- $\hat{y}$ (**y-hat / The Prediction**): The computer's final estimate. The little hat on top of y is data science code for: "This is a guess, not guaranteed truth."

If you have multiple inputs (e.g., apartment size, number of bedrooms, and distance to city center), the computer simply gives each input its own weight knob:

$$\text{Price} = (w_1 \times \text{Size}) + (w_2 \times \text{Bedrooms}) + b$$

C. How Does the Computer Know if It Made a Mistake?

When you train a machine learning model, the computer starts with random guesses for w and b . Naturally, its first guesses will be terrible.

To help the computer get better, we need a scorecard that measures how wrong it is. We call this the **Cost Function** (specifically, **Mean Squared Error**):

$$\text{Error} = (\text{Real Answer} - \text{Model Guess})^2 \quad (2)$$

Why do we square the error? For two very practical reasons:

- 1) **No self-canceling errors**: If the model under-predicts by \$10 on house A (-10) and over-predicts by \$10 on house B ($+10$), adding them directly gives ($-10 + 10 = 0$), foolishly suggesting the model is perfect! Squaring makes both numbers positive ($100 + 100 = 200$).
- 2) **Harsh punishment for huge blunders**: A small mistake of 2 becomes an error score of 4. But an embarrassing mistake of 20 explodes into an error score of 400. The model quickly learns to avoid reckless wild guesses.

D. How the Computer Learns: The Blindfolded Hiker

How does the computer adjust its knobs (w and b) to reduce the error score? It uses a technique called **Gradient Descent**.

The Analogy of the Blindfolded Hiker

Imagine you are blindfolded and placed on the side of a hilly valley in thick fog. Your goal is to reach the lowest cabin at the very bottom of the valley.

- You cannot see the cabin.
- But with your feet, you can feel which direction slopes downward.
- You take a careful step in the downward direction.
- You pause, feel the slope again, and take another step downward.
- You repeat this process over and over until the ground feels completely flat beneath your shoes. You have arrived at the lowest point!

In this analogy:

- The **valley height** is the model's error score.
- The **step direction** is calculated using a little bit of calculus (the gradient).
- The **step size** is called the **Learning Rate** (α). If you take giant basketball jumps, you will overshoot the valley and crash into the other mountain. If your steps are microscopic, it will take you three weeks to reach the bottom. Finding a sensible step size makes learning fast and stable.

III. PART II: WHY LINEAR REGRESSION FAILS FOR YES/NO QUESTIONS

Now let us try something different. Suppose we want to build a spam filter:

- $y = 0$: Legitimate friendly email (Not Spam).
- $y = 1$: Dangerous fraudulent email (Spam).

Can we just draw a straight line through the data like before? Let us see what goes wrong:

Thought Experiment: The Absurd Straight Line

- Suppose an email has 5 suspicious keywords, and our line predicts a spam probability of 0.80 (80% chance of spam). That sounds reasonable.
- But what if an email has 50 suspicious keywords? The straight line keeps climbing upward forever: it outputs a probability of 2.75 (**275% chance**)!
- What if an email comes from your grandmother with zero suspicious keywords? The line plunges downward: it outputs -0.60 (**negative 60% chance**)!

In the real world, **probabilities must live strictly between 0% and 100% (0.0 and 1.0)**. A straight line does not know when to stop; it flies off to positive and negative infinity.

Furthermore, if an email arrives that is ridiculously spammy (say, 500 scam links), a straight line tries so hard to reach that distant point that the whole line tilts, accidentally misclassifying all the normal emails near the middle. We need a mathematical shape that never exceeds 1 and never drops below 0.

IV. PART III: LOGISTIC REGRESSION (PREDICTING YES OR NO)

A. The Magic S-Curve (The Sigmoid Squasher)

Logistic Regression fixes the straight-line problem using an ingenious trick: it takes the straight line and feeds it through an **S-shaped curve** called the **Sigmoid Function**.

$$\text{Probability} = \frac{1}{1 + e^{-z}} \quad \text{where } z = w \cdot x + b \quad (3)$$

Do not worry about the symbol e (it is just Euler's number ≈ 2.718). All you need to know is what this machine does: **It is a universal number squasher!**

- **Input is huge positive (+100):** The sigmoid squashes it down to **0.9999... (99.9%)**
- **Input is huge negative (-100):** The sigmoid squashes it up to **0.0001... (0.01%)**
- **Input is right in the middle (0):** The sigmoid outputs exactly **0.50 (50% coin flip)**

No matter what crazy number $w \cdot x + b$ generates, the sigmoid squashes it neatly into a safe, sensible probability between 0 and 1.

B. How Does Logistic Regression Measure Mistakes? (The Lie Detector)

If someone asks you: "*Will it rain tomorrow?*"

- If you answer "*I think there is a 55% chance of rain,*" and it stays sunny, you were slightly off, but you were cautious. Small penalty.
- But if you shout "*I am 99.99% certain it will pour!*" and the sky is completely clear and dry, you were **confidently wrong**. Big penalty!

Logistic Regression uses an error score called **Binary Cross-Entropy** (or **Log Loss**) that acts just like this lie detector:

- If the truth is **1** and you guess **0.99**: Error ≈ 0 (Reward for being right).
- If the truth is **1** and you guess **0.50**: Error is medium (Hesitant guess).
- If the truth is **1** and you guess **0.01**: Error is near infinity (Huge penalty for stubborn confidence).

This forces the model to be calibrated and honest about its confidence.

C. The Decision Threshold: Drawing the Line

Once the model outputs a probability (e.g., “*There is an 84% chance this email is spam*”), how do we make a final decision?

We set a simple cutoff threshold (usually **50%** or 0.5):

- If Probability $\geq 0.5 \implies$ **Mark as Spam (Class 1)**
- If Probability $< 0.5 \implies$ **Send to Inbox (Class 0)**

You can even adjust this threshold based on real-world stakes:

- For a medical cancer screening, you want to be extra safe: if probability ≥ 0.20 (20%), flag for further tests!
- For a spam filter, you don’t want to accidentally delete an important bank email: you might only delete if probability ≥ 0.90 (90%).

—

V. PART IV: DIRECT SIDE-BY-SIDE COMPARISON

Here is everything you need to remember in one simple table:

TABLE I
LINEAR REGRESSION VS. LOGISTIC REGRESSION IN PLAIN ENGLISH

Feature	Linear Regression	Logistic Regression
The Big Question	“How much?” (Price, Height, Time)	“Yes or No?” (Spam, Churn, Disease)
Output Type	Any real number ($-\infty$ to $+\infty$)	A probability strictly between 0 and 1
The Shape	A straight diagonal line	An S-shaped curved wave (Sigmoid)
Mistake Metric	Squared Distance (Mean Squared Error)	Confidence Penalty (Log Loss)
Key Weakness	Vulnerable to extreme outliers	Cannot easily model twisty zigzag boundaries
Analogy	Estimating a taxi fare with a meter	A doctor guessing chances of recovery