



# DAY 19 — Logistic Regression (Classification)

Goal : Predict probabilities and make binary decisions.

## 1 Why We Need Logistic Regression

Linear Regression predicts **continuous values**:

- Salary
- Price
- Temperature

But many real ML problems are **yes/no**:

- Spam or not spam?
- Fraud or not fraud?
- Pass or fail?

We need a model that outputs:

Probability between 0 and 1

That's where **Logistic Regression** comes in.

## 2 Logistic Regression is NOT “Regression”

Despite the name:

| Logistic Regression is a **classification algorithm**

Why the name?

- It uses a regression-like equation
- But applies a **non-linear function**

## 3 The Core Idea

### Step 1 — Linear combination

$$z = w_1x_1 + w_2x_2 + \dots + b$$

### Step 2 — Sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

This converts:

$$(-\infty, +\infty) \rightarrow (0, 1)$$

### Interpretation

- Output = probability of class 1
- Example: 0.82 → 82% chance of "YES"

## 4 Decision Boundary

Default rule:

Probability  $\geq 0.5 \rightarrow$  Class 1

Probability  $< 0.5 \rightarrow$  Class 0

But **0.5 is NOT special.**

- You can change it depending on the problem
- This controls precision vs recall

## 5 Why we can't use MSE

Linear regression uses **MSE**.

Logistic regression uses **Log Loss (Cross-Entropy)**

### Log Loss Intuition:

- Confident & wrong → heavy penalty
- Uncertain → smaller penalty

This makes training **stable and meaningful**.

## 6 Cost Function

for one data point:

```
If y = 1 → -log(p)  
If y = 0 → -log(1-p)
```

This forces:

- High probability for correct class
- Low probability for wrong class

## 7 Logistic Regression Learns Using Gradient Descent

Same idea as linear regression:

1. Start with random weights
2. Compute loss
3. Upgrade weights
4. Repeat

Difference:

- Uses **Log loss**
- Uses **sigmoid**

## 8 Regularization in Logistic Regression

Logistic Regression **supports regularization by default**.

Type	Effect
L2 (Ridge)	Default, stable
L1 (Lasso)	Feature selection

```
LogisticRegression(penalty="l1", solver="liblinear")
```

Regularization prevents overfitting in classification too.

## 9 Implement Logistic Regression (Sklearn)

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression()  
model.fit(X_train, y_train)  
  
y_prob = model.predict_proba(X_test)  
y_pred = model.predict(X_test)
```

## 10 Interpreting Coefficients

```
model.coef_
```

Meaning:

- Positive weight → increases probability of class 1
- Negative weight → decrease probability

This makes logistic regression **interpretable**.

## 11 Logistic Regression vs Linear Regression

Aspect	Linear	Logistic
Output	Any number	Probability
Task	Regression	Classification
Loss	MSE	Log Loss
Boundary	Line	Sigmoid curve