Introduction to Supervised Machine Learning

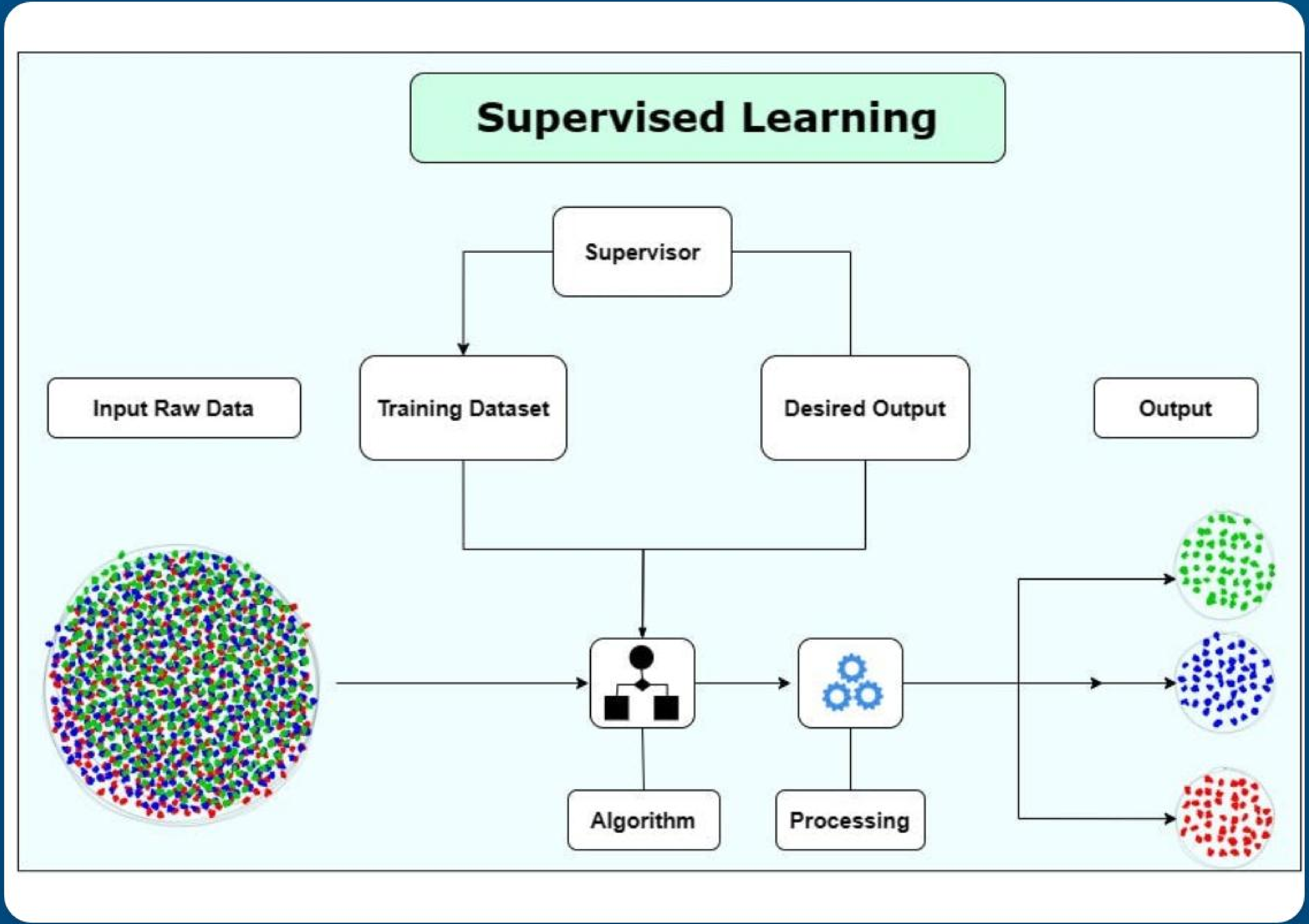
From Linear Regression to KNN: A Mathematical Deep Dive

What is Supervised Learning?

The Core Concept

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.

- **Input (X):** Features or variables (e.g., email content, house size).
- **Output (Y):** Target or label (e.g., "Spam", Price).
- **Goal:** Approximate the mapping function so well that when you have new input data (x), you can predict the output variables (Y) for that data.



Linear Regression: The Intuition

Fitting a Line to Data

Imagine we want to predict housing prices based on the size of the house.

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.

Visually, we try to draw the "best fit" straight line through our scatter plot of data points.

Scatter plot of housing prices vs size with a regression line

Mathematical Formulation

The Hypothesis

The function we are trying to learn is called the Hypothesis, denoted by *h*.

$$h_\theta(x) = \theta_0 + \theta_1 x$$

This is simply the equation of a straight line.

The Parameters

The θ values are called parameters or weights.

- θ_0 : The bias unit (y-intercept).
- θ_1 : The feature weight (slope).

Goal: Choose θ_0 and θ_1 so that $h_\theta(x)$ is close to y for our training examples.

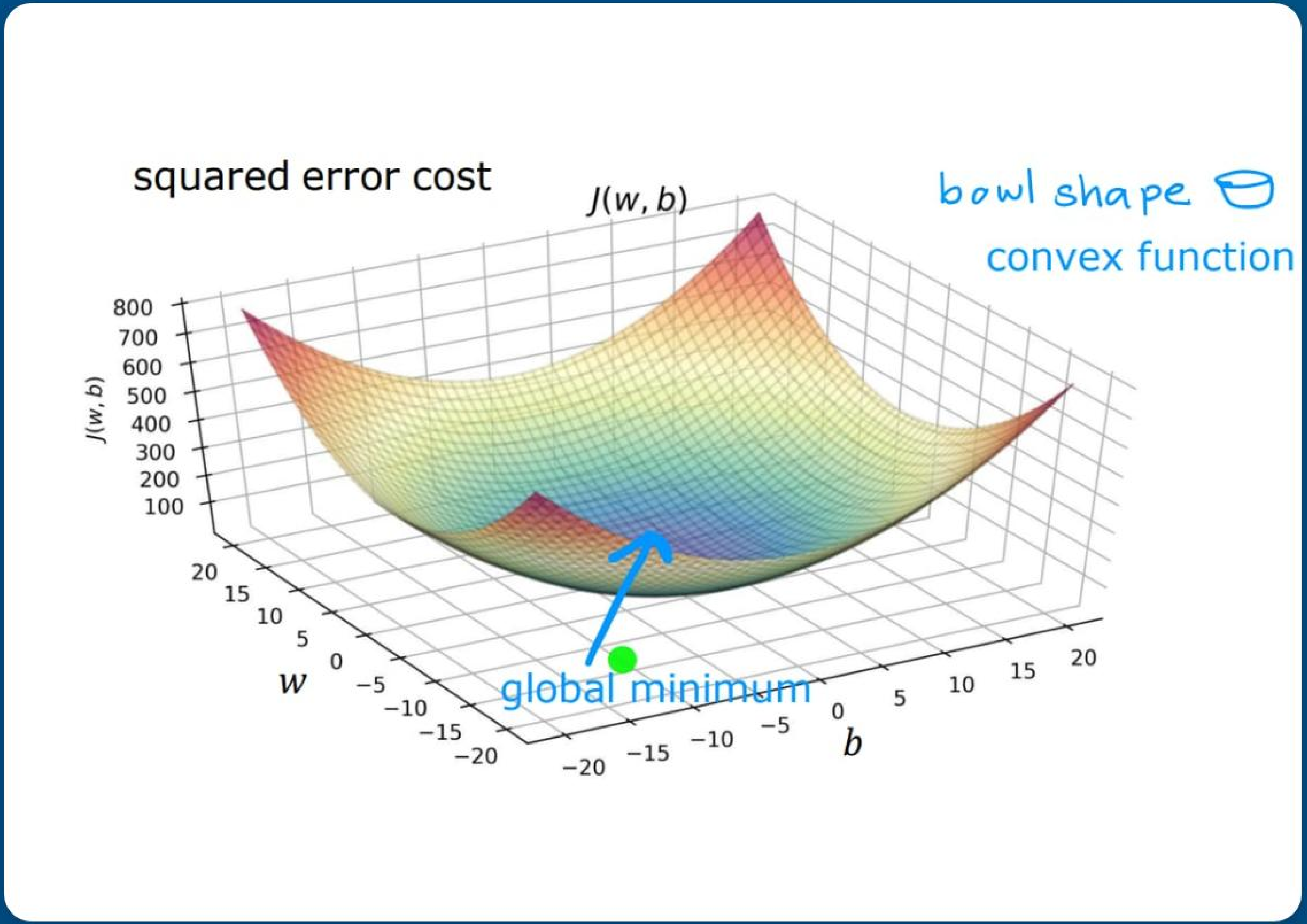
The Cost Function (MSE)

Measuring the Error

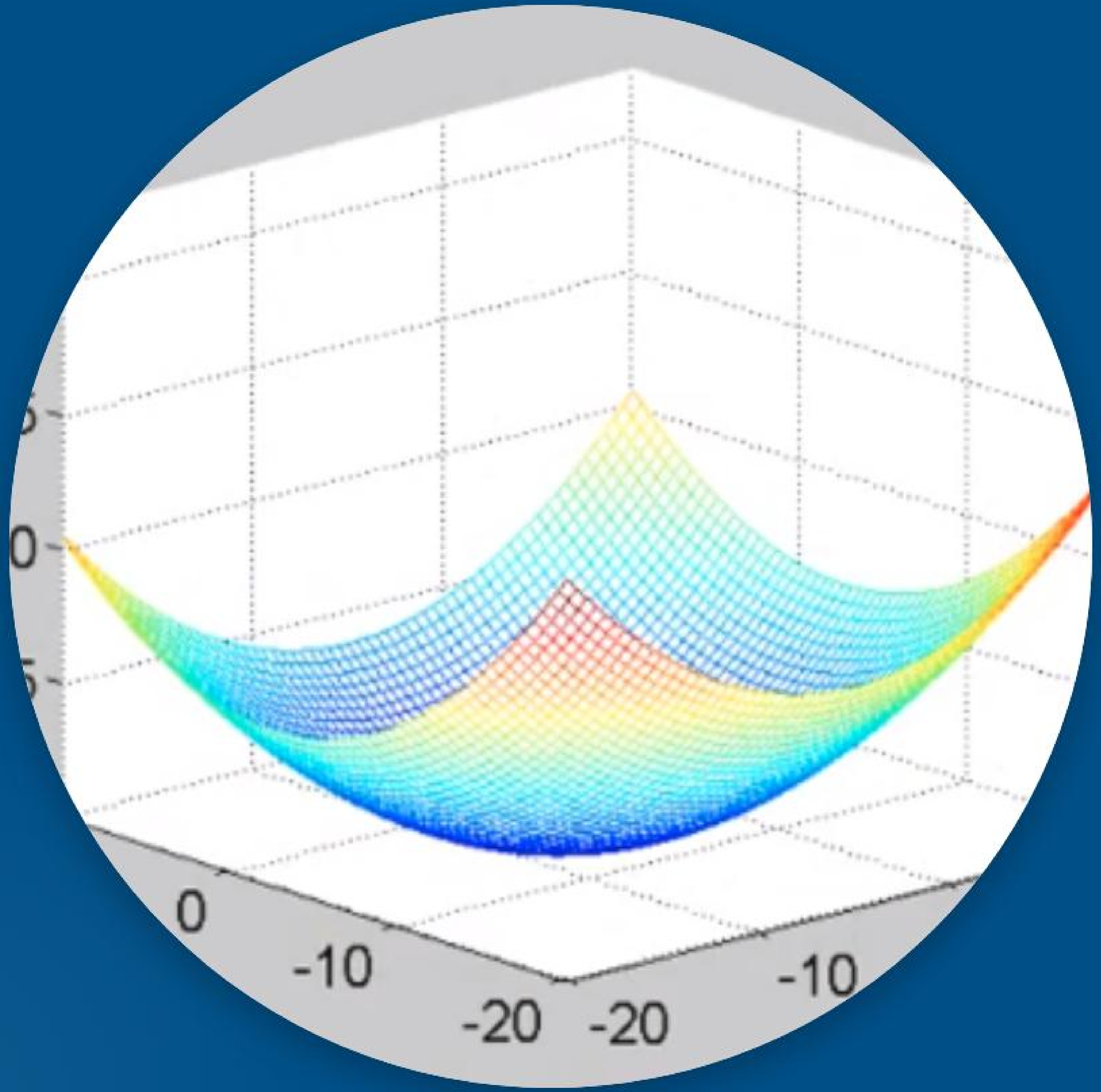
We need a metric to measure how well our line fits the data. We use the **Mean Squared Error (MSE)** cost function, denoted as $J(\theta)$.

$$J\theta_0\,\theta_1 = \frac{1}{2m}\sum_{i=1}^{m} h_\theta\left(x^{(i)}\right) - y^{(i)\,2}$$

Here, *m* is the number of training examples. We square the error to penalize large deviations. The 1/2 is for mathematical convenience during differentiation.



Gradient Descent: Intuition



Walking Down the Hill

Imagine you are standing on top of a hill (the cost function surface) and you want to get to the bottom (minimum cost) as quickly as possible.

You look around 360 degrees and ask, "If I take a baby step, in which direction will I go down the steepest?"

You take a step in that direction, and repeat the process. Eventually, you will reach the valley floor (global minimum).

Gradient Descent Algorithm

The Process

Repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

Learning Rate

α (alpha) is the Learning Rate.

It controls how big of a step we take downhill. If too small, convergence is slow. If too large, we might overshoot the minimum.

Simultaneous Update

Crucially, we must update both θ_0 and θ_1 simultaneously at each step of the iteration.

Deriving the Update Rules

Derivative for Bias (θ_0)

By taking the partial derivative of the MSE cost function with respect to θ_0:

$$\frac{\partial}{\partial\theta_0}J(\theta) = \frac{1}{m}\sum_{i=1}^{m} h_\theta(x^{(i)}) - y^{(i)}$$

Derivative for Weight (θ_1)

Similarly, for θ_1, applying the chain rule gives us an extra x term:

$$\frac{\partial}{\partial\theta_1}J(\theta) = \frac{1}{m}\sum_{i=1}^{m} h_\theta(x^{(i)}) - y^{(i)} \cdot x^{(i)}$$

Multivariate Linear Regression

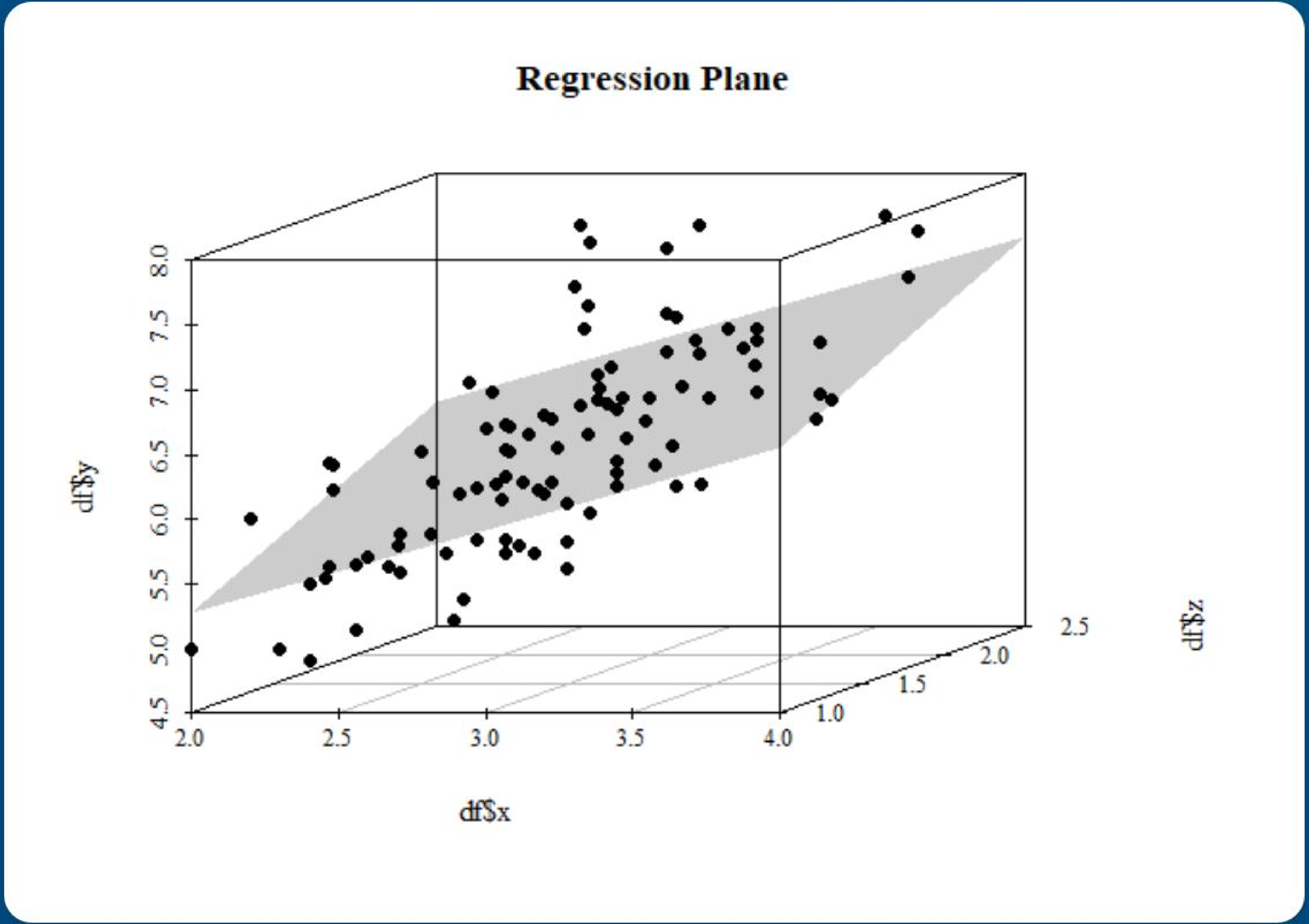
Multiple Features

When we have more than one feature (e.g., size, number of bedrooms, age of house), we denote the input as a vector x.

The hypothesis becomes:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

In this space, we are fitting a hyperplane rather than a simple line.



Vectorization & Normal Equation

Vectorized Hypothesis

To compute efficiently, we define $x_0 = 1$ and use linear algebra:

$$h_\theta(x) = \theta^T x$$

This allows us to compute predictions for the entire dataset in one matrix multiplication.

The Normal Equation

Instead of iterating with Gradient Descent, we can solve for θ analytically:

$$\theta = (X^T X)^{-1} X^T y$$

This gives the exact minimum but is computationally expensive for very large datasets ($O(n^3)$).

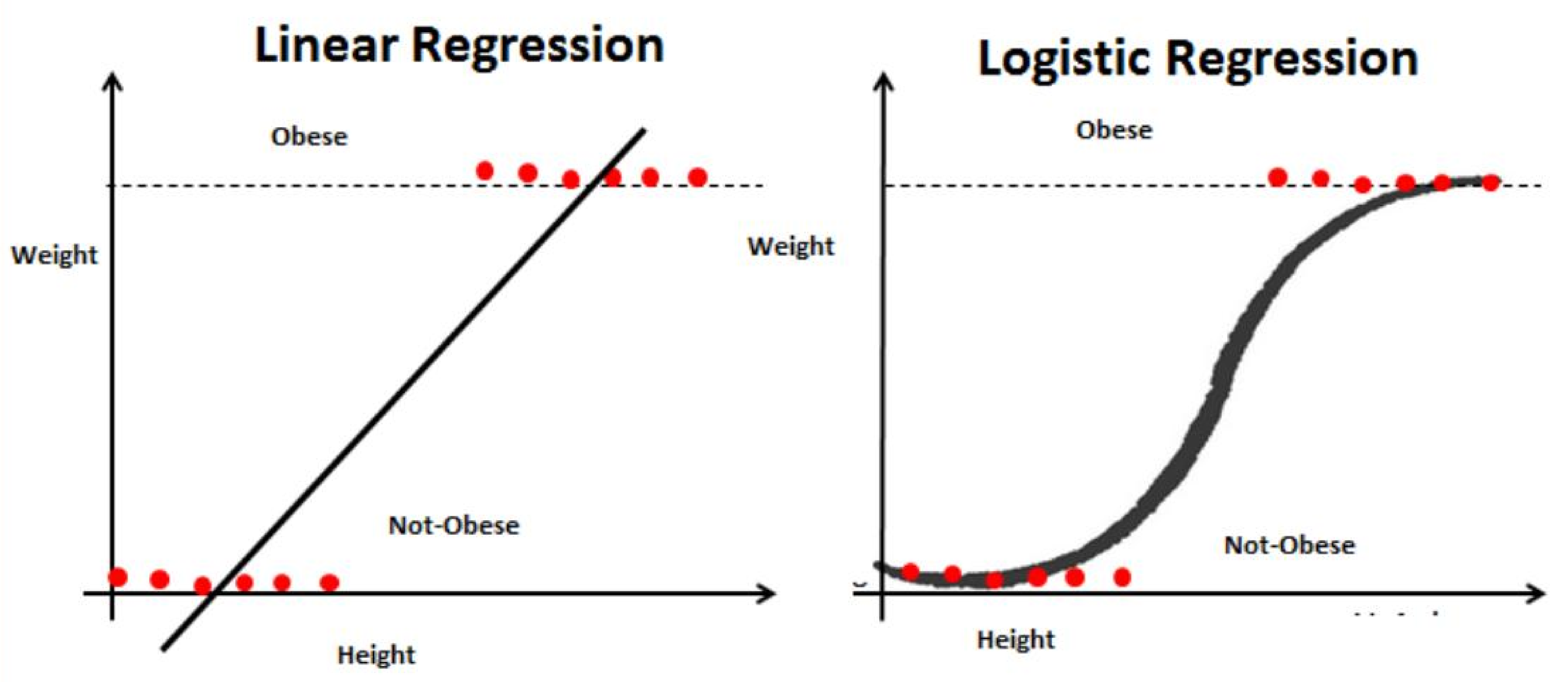
Introduction to Classification

Discrete Outputs

In classification, the variable y is discrete (e.g., 0 or 1, Spam or Not Spam).

Why not Linear Regression?

- Linear regression predicts continuous values, which can be > 1 or < 0.
- It is highly sensitive to outliers, which can shift the decision boundary drastically and incorrectly.



Logistic Regression: The Sigmoid

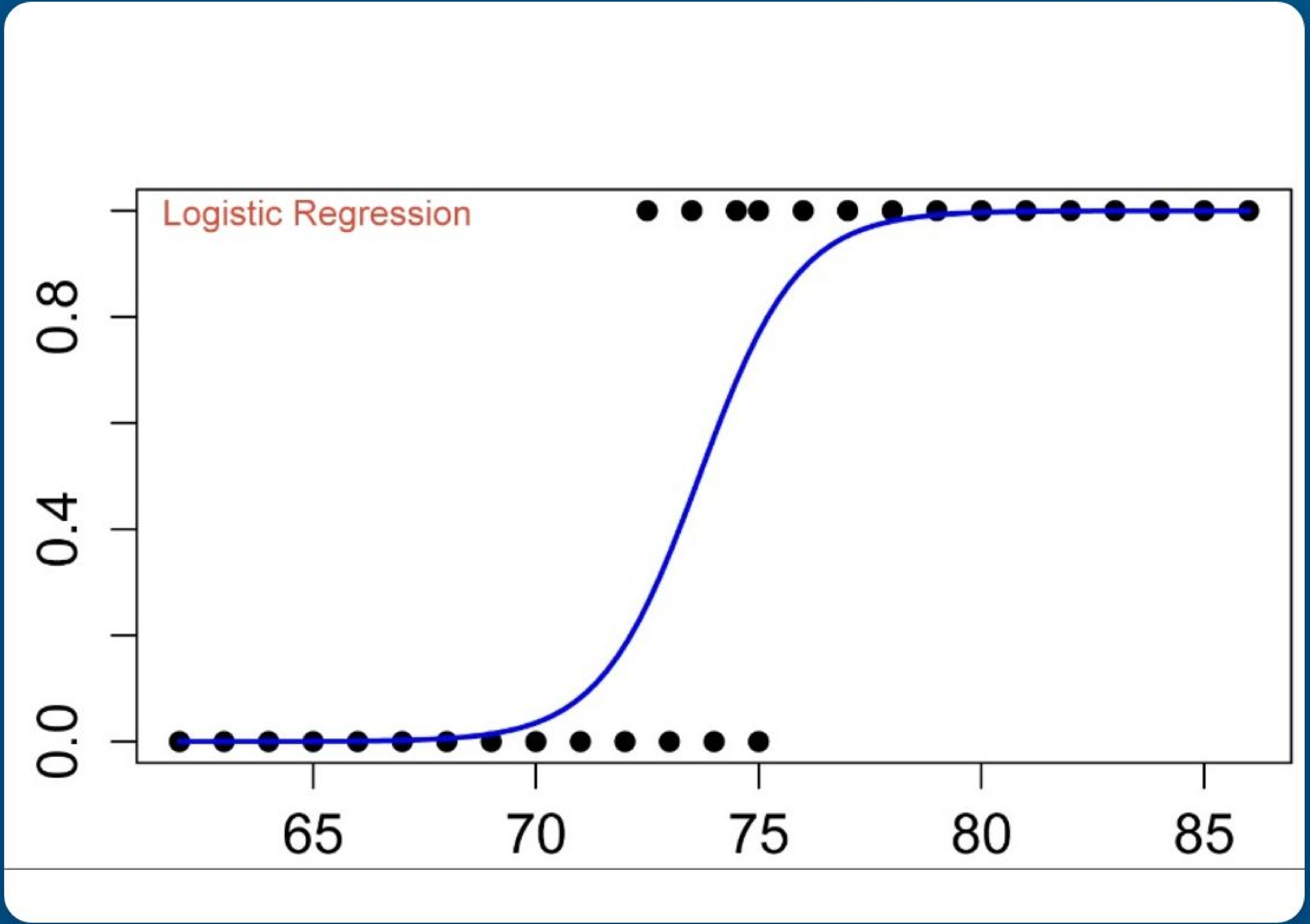
The Sigmoid Function

We want $0 \le h_\theta(x) \le 1$. We map the linear output through the Sigmoid function (Logistic function):

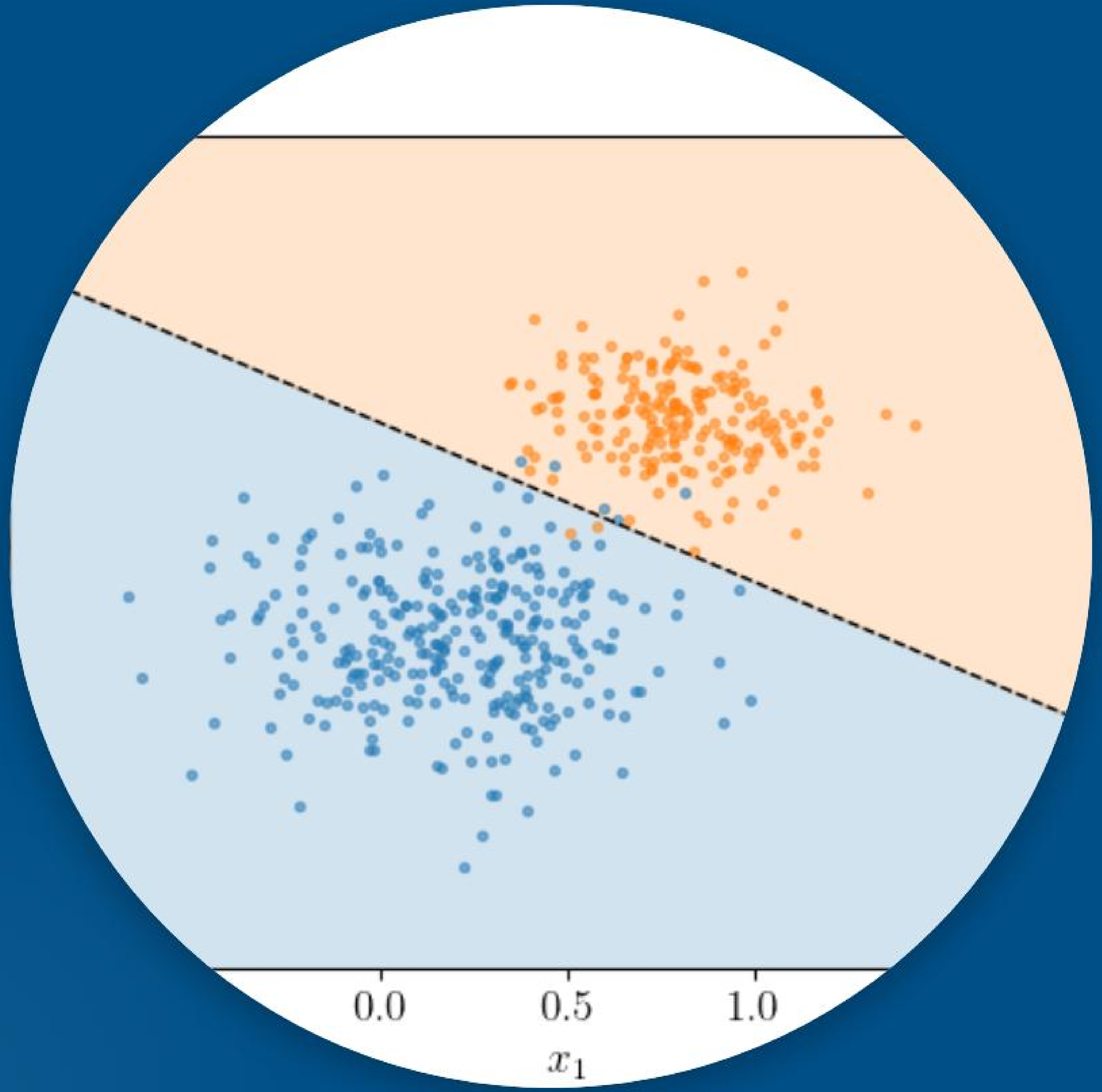
$$g(z) = \frac{1}{1 + e^{-z}}$$

Our hypothesis becomes $h_\theta(x) = g(\theta^T x)$.

This outputs the **probability** that y=1.



The Decision Boundary



Making a Prediction

Even though the output is probabilistic, we can enforce a hard classification.

- Predict $y = 1$ if $h_\theta(x) \geq 0.5$
- Predict $y = 0$ if $h_\theta(x) < 0.5$

This corresponds to $\theta^T x \geq 0$. The line where $\theta^T x = 0$ is the decision boundary.

Logistic Regression Cost Function

Why not MSE?

Using Mean Squared Error with the sigmoid function results in a **non-convex** cost function with many local minima, making gradient descent unreliable.

Log Loss (Cross-Entropy)

We use a convex cost function derived from maximum likelihood estimation:

$$\text{Cost} = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

This heavily penalizes confident wrong predictions (e.g., predicting 1 when actual is 0).

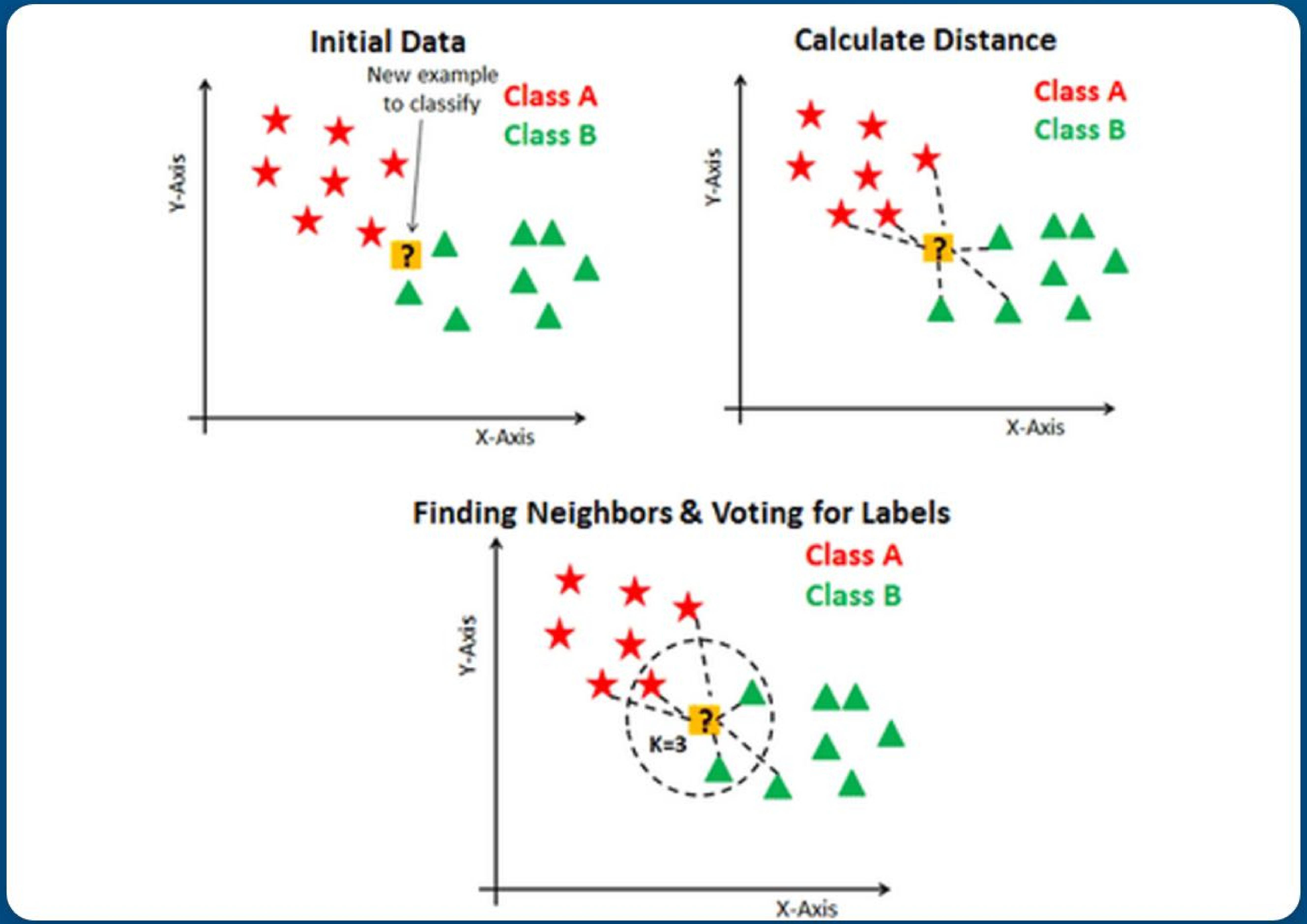
K-Nearest Neighbors (KNN)

Instance-Based Learning

KNN is a simple, non-parametric, lazy learning algorithm. It doesn't learn a "model" or coefficients like regression.

The Logic: "Tell me who your neighbors are, and I'll tell you who you are."

To classify a new data point, we look at the 'K' closest training examples and take a majority vote.



KNN: Distance & Parameters

Euclidean Distance

How do we define "closest"? The most common metric is Euclidean distance:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Choosing 'K'

The choice of K is crucial:

- **Small K (e.g., 1):** Sensitive to noise and outliers (High Variance).
- **Large K:** Smoother decision boundaries, but may miss local patterns (High Bias).

Key Takeaways

Linear Regression

Used for predicting continuous values. Optimizes MSE using Gradient Descent.

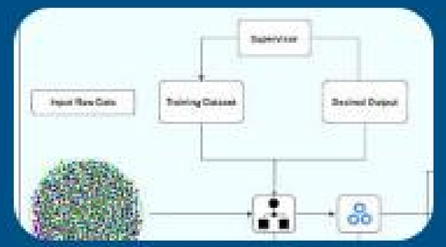
Logistic Regression

Used for binary classification. Outputs probabilities using the Sigmoid function.

K-Nearest Neighbors

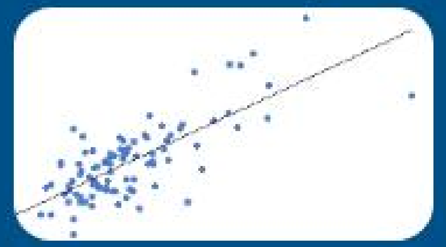
A simple, geometric classifier based on distance to nearby training points.

Image Sources



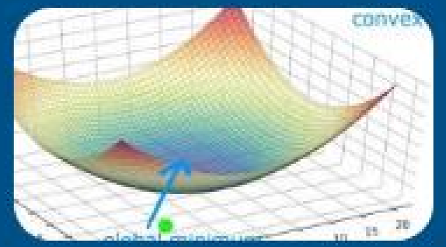
https://images.prismic.io/encord/190bb719-b32f-4da7-85c7-5e3892ff2b38_Supervised+Learning+Flowchart+-+Encord.jpg?auto=compress,format

Source: encord.com



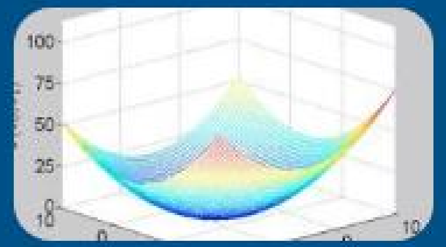
https://prd-api-aggregate.statcrunch.com/api/aggregation/documents/77411ZQKDW?context=results_image&code=&extension=png

Source: www.statcrunch.com



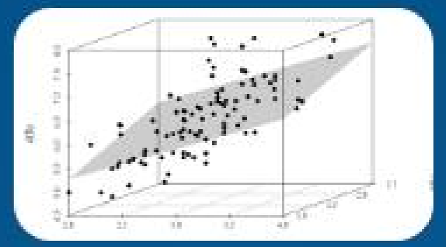
https://global.discourse-cdn.com/dlai/original/3X/1/3/133adb05a8ab320f2d069daacaa20f21bbe63a9d.jpeg

Source: community.deeplearning.ai



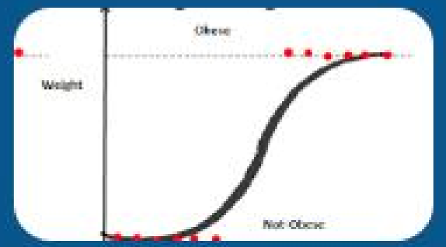
https://i.sstatic.net/Rq40j.png

Source: stackoverflow.com



https://i.sstatic.net/Tc3YO.png

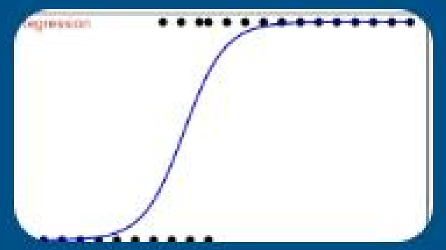
Source: stackoverflow.com



https://cdn.analyticsvidhya.com/wp-content/uploads/2020/12/image-96.png

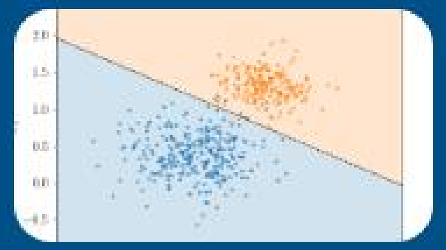
Source: www.analyticsvidhya.com

Image Sources



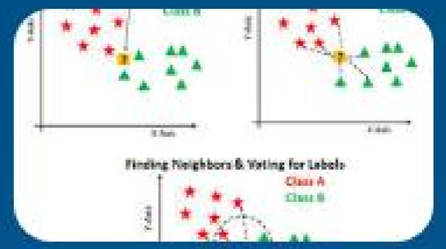
https://miro.medium.com/1*bCCcQhMjHGal89i-7i3xFw.png

Source: medium.com



https://scipython.com/media/old_blog/logistic_regression/decision-boundary.png

Source: scipython.com



https://insightimi.wordpress.com/wp-content/uploads/2020/03/knn-start.png

Source: insightimi.wordpress.com