# . Bias–Variance Tradeoff

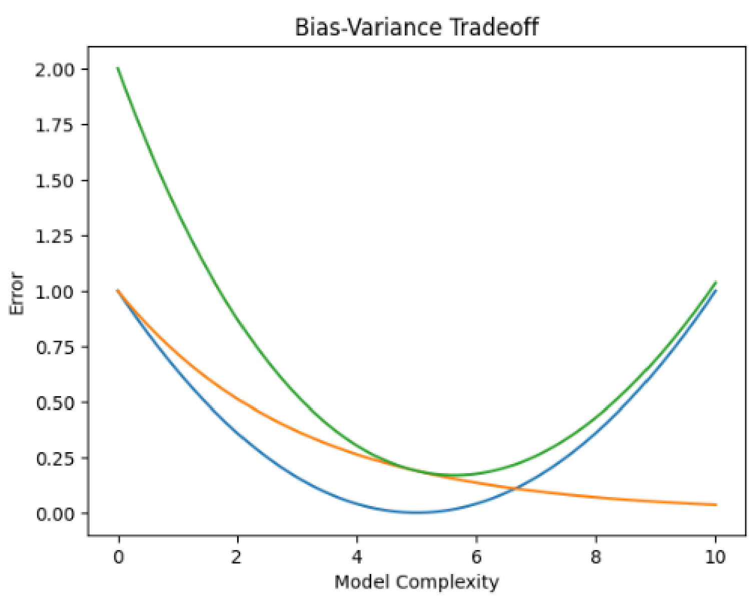
**Definition:**
The bias–variance tradeoff describes the balance between:

- **Bias** → error due to a model being too simple (underfitting)
- **Variance** → error due to a model being too complex (overfitting)

**Graph explanation:**

- X-axis: Model complexity
- Y-axis: Error
- Bias decreases as complexity increases
- Variance increases as complexity increases
- **Total error is minimum at an optimal complexity**

🎯 Goal: find the sweet spot where total error is lowest.



## 2. Loss Function vs Cost Function

**Loss Function:**

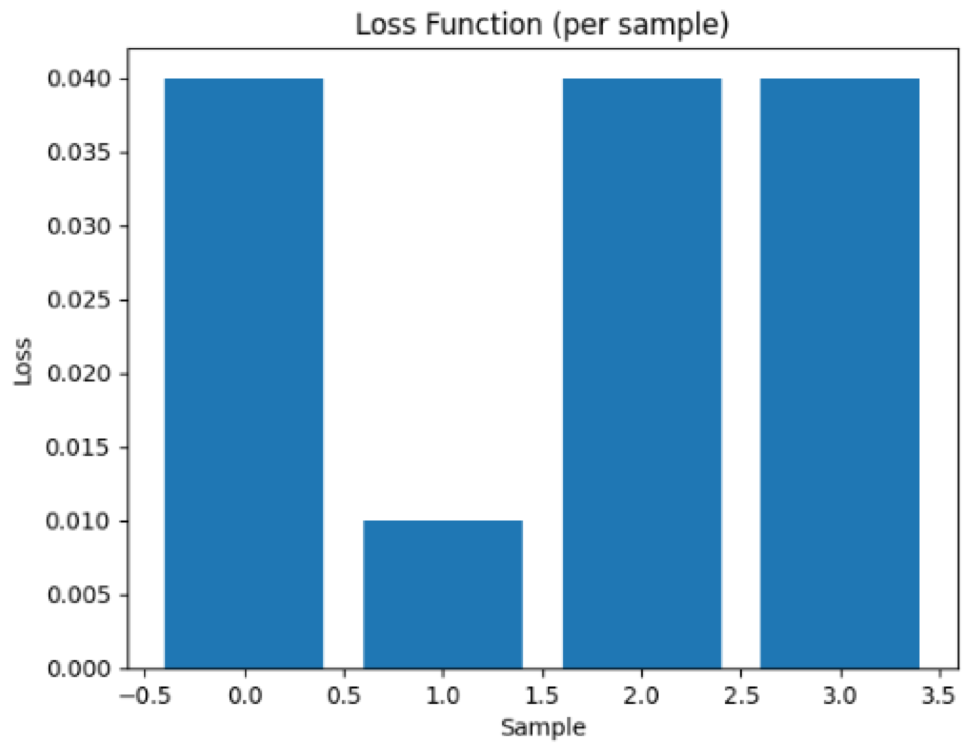
- Measures error for **one training example**
- Example: Mean Squared Error for a single data point

**Cost Function:**

- Average (or sum) of loss over **all training examples**

**Graph explanation:**

- Each bar shows loss for one sample
- Cost = average height of all bars

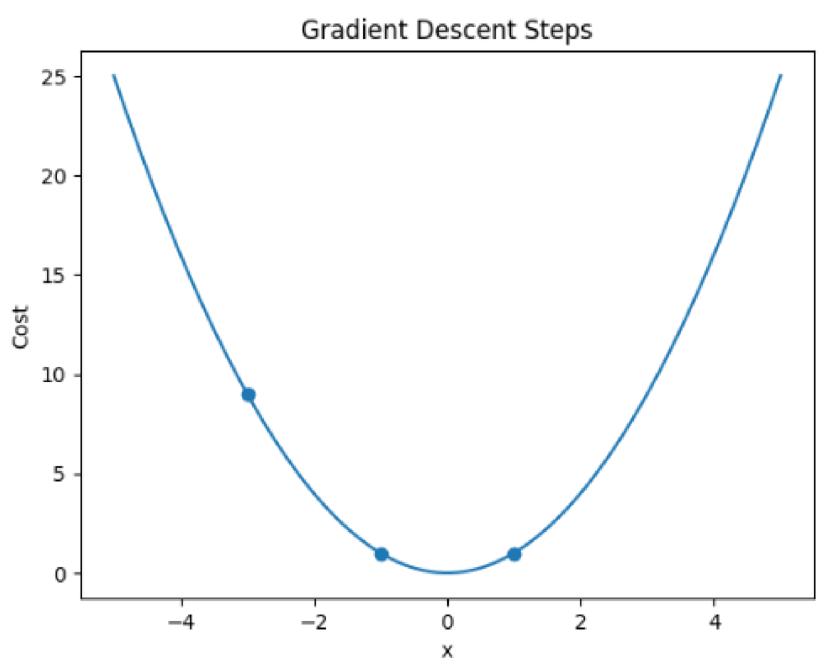


# 3. Gradient Descent

**Definition:**
Gradient descent is an optimization algorithm used to **minimize the cost function** by updating parameters in the direction of the **negative gradient**.

**Graph explanation:**

- Curve = cost function
- Dots = parameter updates
- Steps move downhill until reaching the minimum



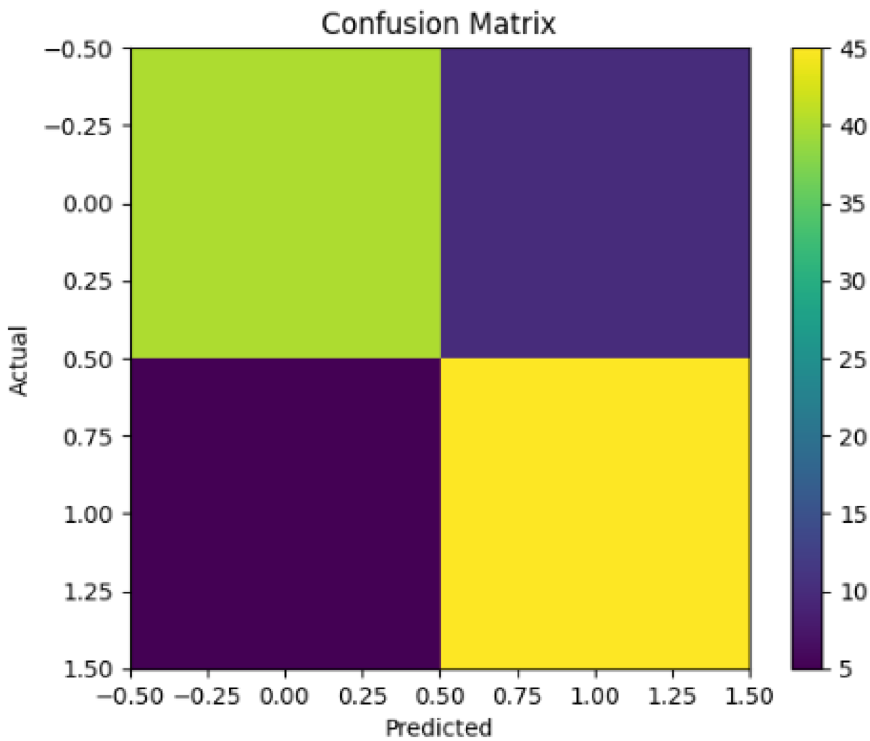

# 4. Confusion Matrix

**Definition:**
A confusion matrix evaluates classification performance by comparing **actual vs predicted** labels

| Actual / Predicted | Positive | Negative |
|---|---|---|
| Positive | TP | FN |
| Negative | FP | TN |

**Graph explanation:**

- Heatmap shows counts
- Diagonal values = correct predictions
- Off-diagonal = errors



## 5 L1 (Lasso) and L2 (Ridge) Regularization

# Why Regularization is needed

When a model is **too complex**, it fits noise → **overfitting**.
Regularization **adds a penalty** to the loss function to keep weights small and the model simple.

### General formula

Loss=Error+λ×Penalty\text{Loss} = \text{Error} + \lambda \times \text{Penalty}Loss=Error+λ×Penalty

- λ (lambda) controls **how strong** the penalty is

---

# ◆ L1 Regularization (Lasso)

### Formula

Loss=MSE+λ∑|wi|\text{Loss} = \text{MSE} + \lambda \sum |w_i|Loss=MSE+λ∑|wi|

### Key ideas

- Penalizes **absolute value** of weights
- Can make some weights **exactly zero**
- Performs **feature selection automatically**

### Intuition

- Forces the model to **ignore less important features**
- Produces a **sparse model**

### When to use L1

- When you have **many features**
- When you want **feature selection**
- When interpretability matters

### Interview line

*"Lasso regularization can shrink coefficients to zero, effectively performing feature selection."*

---

# ◆ L2 Regularization (Ridge)

## Formula

Loss=MSE+λ∑wi2\text{Loss} = \text{MSE} + \lambda \sum w_i^2Loss=MSE+λ∑wi2

## Key ideas

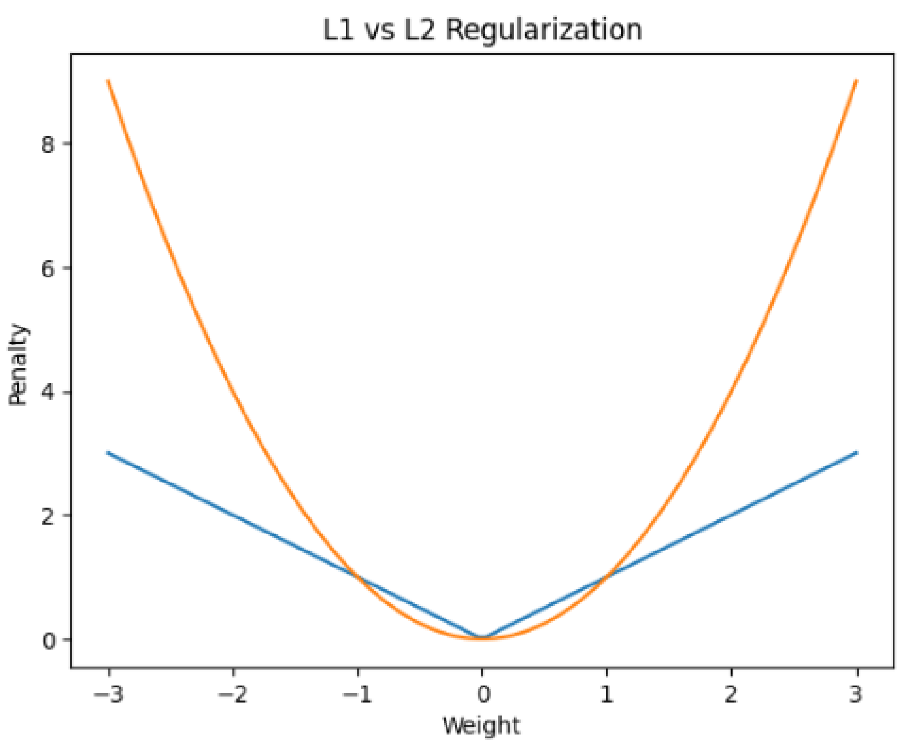
- Penalizes **square of weights**
- Shrinks weights **close to zero but not exactly zero**
- Keeps all features

## Intuition

- Reduces the influence of large coefficients
- Makes the model **stable and smooth**

## When to use L2

- When **all features are useful**
- When features are **correlated**
- To reduce variance without removing features



| Aspect | L1 (Lasso) | L2 (Ridge) |
|---|---|---|
| Penalty | \|w\| | $w^2$ |
| Feature selection | Yes | No |
| Coefficients | Some become 0 | All small |
| Use case | High-dimensional data | Multicollinearity |

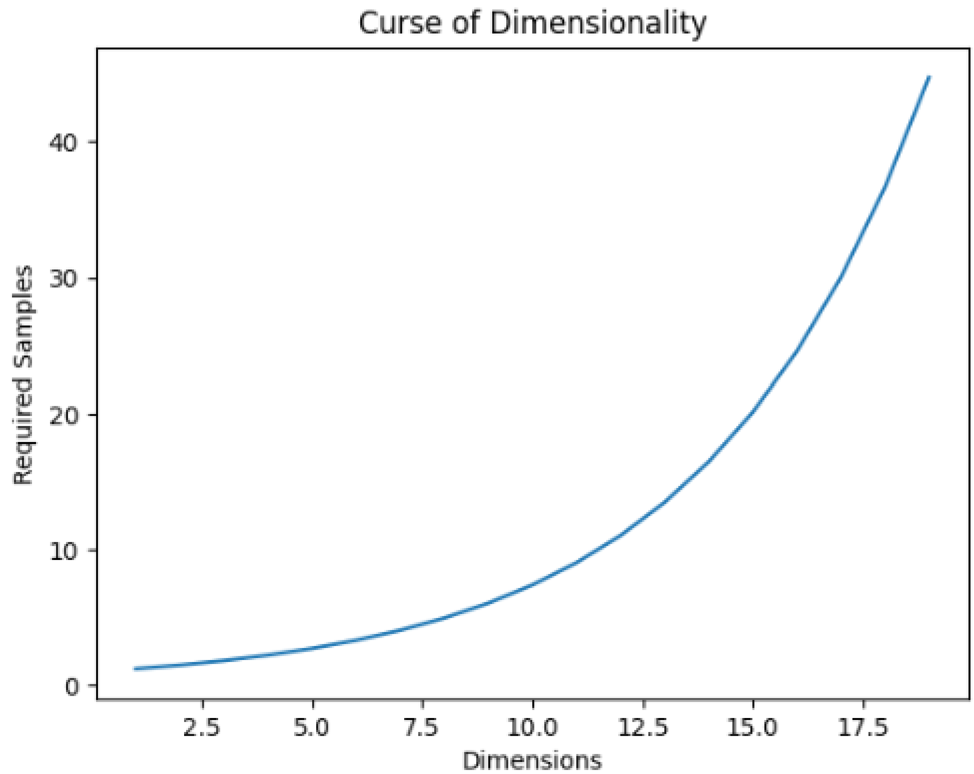
# 6. Curse of Dimensionality

**Definition:**
As the number of features increases, the amount of data required to generalize well grows **exponentially**.

**Graph explanation:**

- X-axis: number of dimensions
- Y-axis: required samples
- Sharp rise shows why high-dimensional data is hard

⚠ Leads to sparsity and poor model performance.




# 7. Logistic Regression vs Linear Regression

**Linear Regression:**

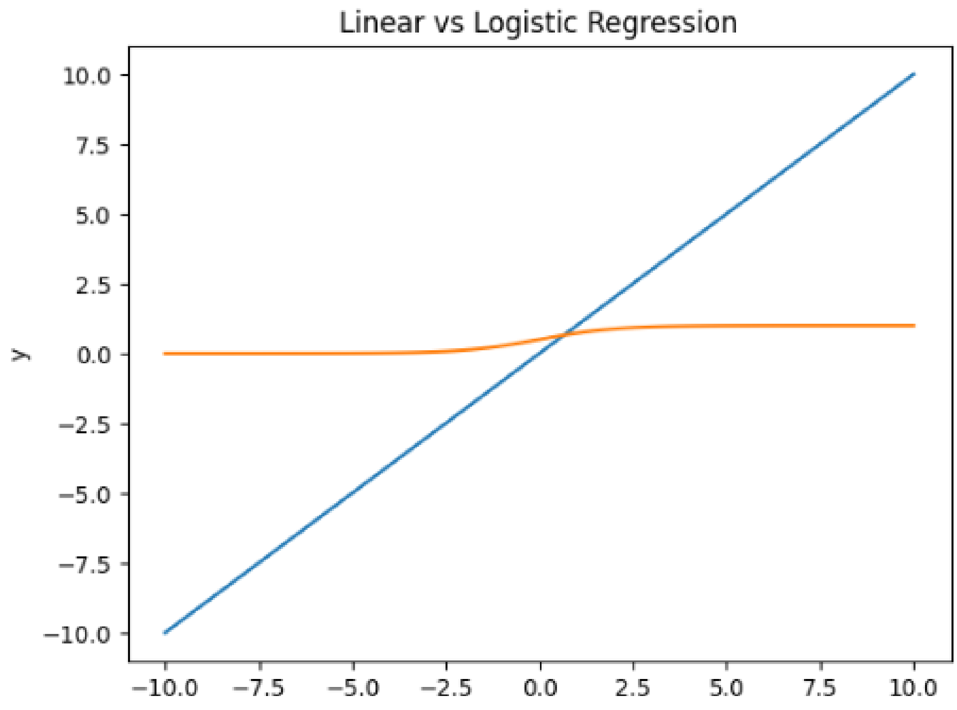
- Predicts continuous values
- Output range: $(-\infty, +\infty)$

**Logistic Regression:**

- Used for classification
- Outputs probability between **0 and 1** using sigmoid

**Graph explanation:**

- Straight line → linear regression
- S-shaped curve → logistic regression




## 8 Parameters vs Hyperparameters

# ◆ Parameters

### What they are

- Learned **from the data**
- Updated during training

### Examples

- Linear regression: weights (w), bias (b)
- Logistic regression: coefficients
- Neural networks: weights & biases

### Role

- Define the **model itself**
- Decide the final prediction function

### Interview line

*"Parameters are learned from training data during model training."*

---

## ◆ Hyperparameters

### What they are

- Set **before training**
- Control **how learning happens**

### Examples

- Learning rate
- Regularization strength (λ)
- Number of trees in Random Forest
- k in KNN
- Number of epochs

### Role

- Control **model complexity and training behavior**
- Tuned using cross-validation or grid search

| Aspect | Parameters | Hyperparameters |
|---|---|---|
| Learned from data | ✅ | ❌ |
| Set before training | ❌ | ✅ |
| Example | weights, bias | learning rate, λ |
| Updated during training | Yes | No |

# 9. Feature Engineering vs Feature Selection

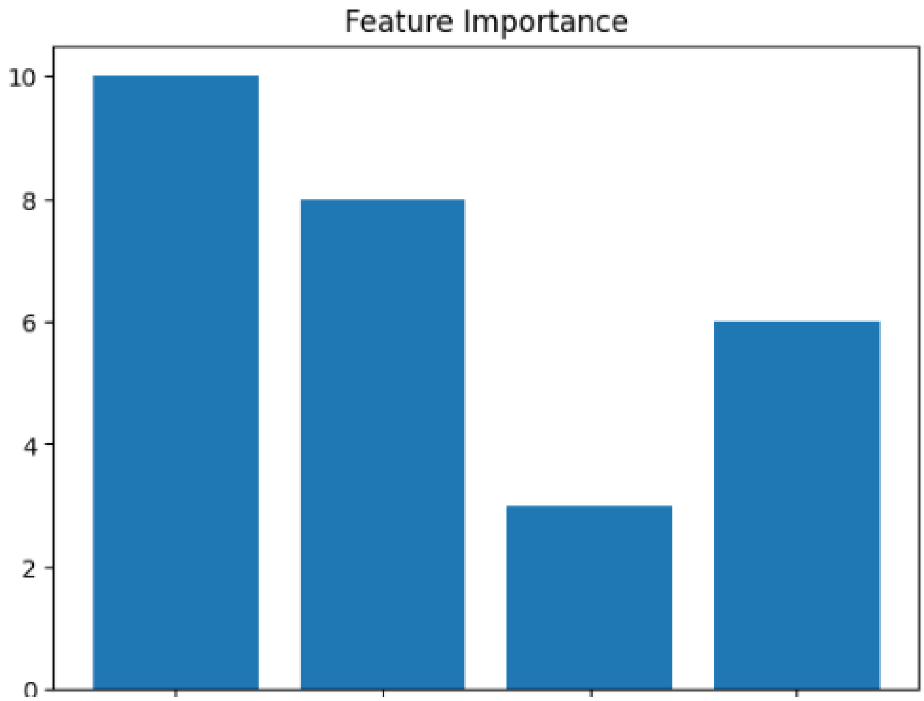
**Feature Engineering:**

- Creating new features from existing data
- Example: extracting day/month from date

**Feature Selection:**

- Choosing the most important features
- Removing irrelevant or redundant ones

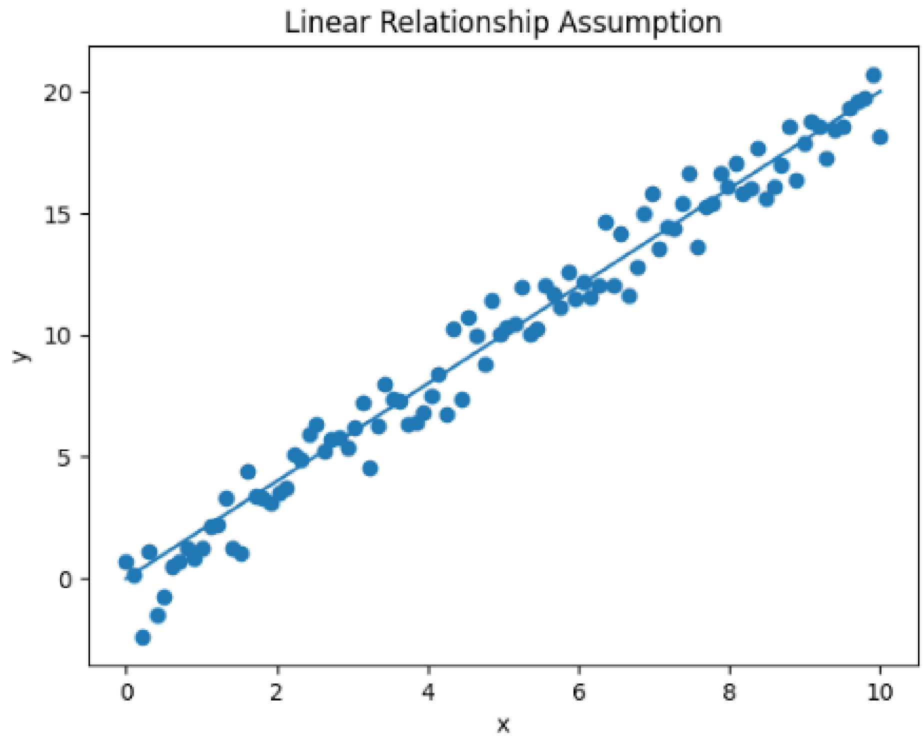
**Graph explanation:**

- Bar heights show feature importance
- Selection keeps high-importance features
- 


# 10. Key Assumptions of Linear Regression

Main assumptions:

1. Linearity
2. Independence of errors
3. Homoscedasticity (constant variance)
4. Normality of errors
5. No multicollinearity




- Points roughly follow a straight line
- Shows linear relationship between X and y