Day 1: Machine Learning Foundations

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

December 20, 2025

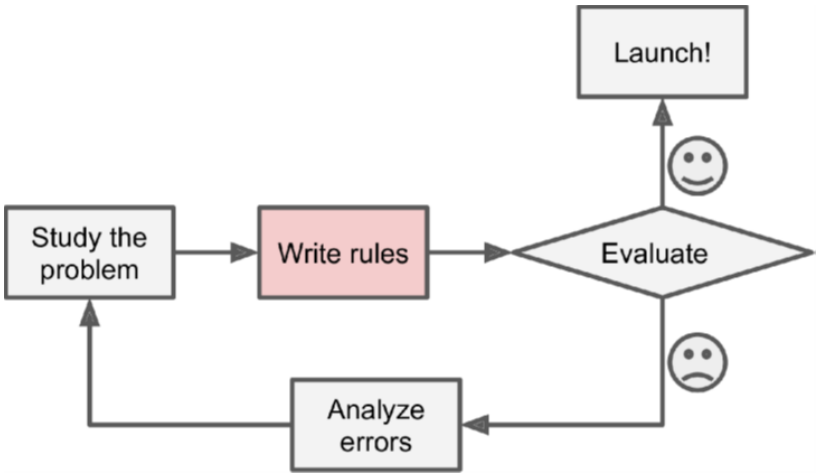
Learning Outcomes

- Explain the core components of the ML framework and how they differ from traditional programming.
- Identify data problems and model failures using loss curves.
- Select appropriate data splitting strategies (Stratified K-Fold) to simulate unseen data.
- Select and justify the appropriate metrics based on data characteristics.
- Understand essential preprocessing techniques including cleaning, scaling, and feature selection to optimize model performance.

Table of Contents

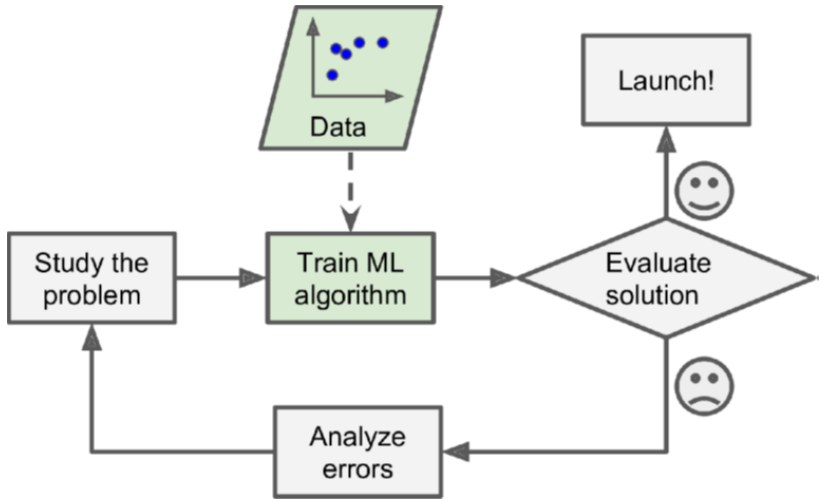
What is Machine Learning?

Machine Learning is the field of study that gives computers the ability to learn (using data) without being explicitly programmed.



The traditional approach

Rules + Data = Answers



The Machine Learning approach

Data + Answers = Rules

Data Types

1. Structured Data

Organized in rows and columns.

- **Tabular Data**
  - Spreadsheets, SQL Databases
  - *Columns = Features, Rows = Samples*
- **Time-Series Data**
  - Data indexed by time
  - Stock prices, Weather, IoT sensors

2. Unstructured Data

Complex formats.

- **Text Data (NLP)**
  - Emails, Tweets, Documents
- **Images & Video (Computer Vision)**
  - Medical imaging, Surveillance, Face ID
- **Audio Data**
  - Voice commands, Music processing

Note: *Machine Learning excels at Structured Data, while Deep Learning dominates Unstructured Data.*

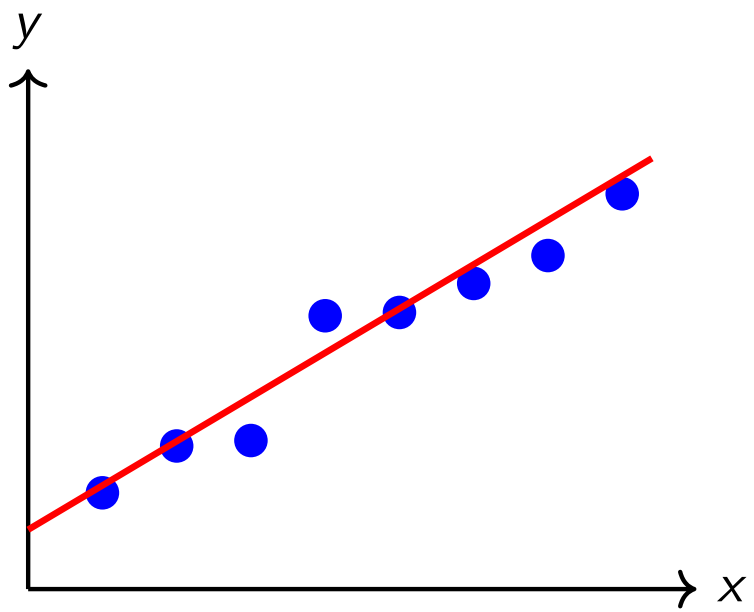
Types of Machine Learning

1. Supervised Learning

We have input data X and corresponding target labels y. The goal is to learn a mapping $f(x) \approx y$.

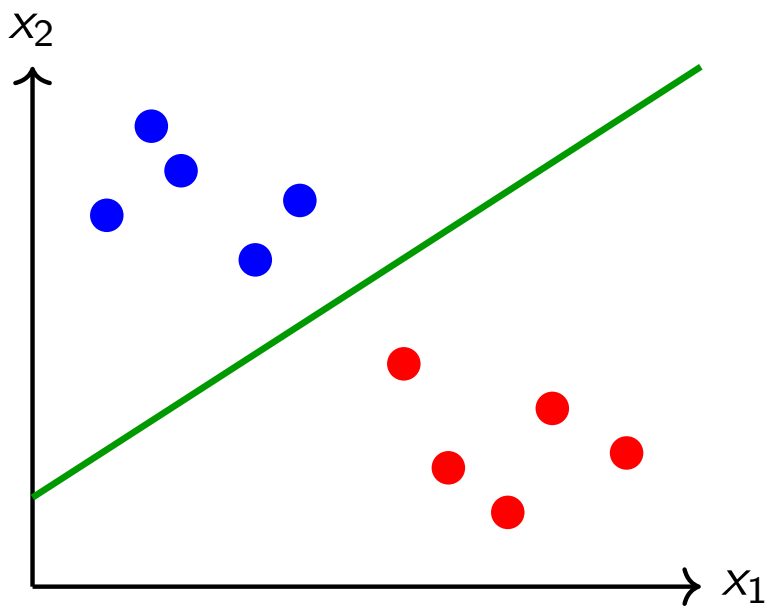
A. Regression

Predicting continuous values (e.g., Price)



B. Classification

Predicting discrete categories (e.g., Spam Classification)

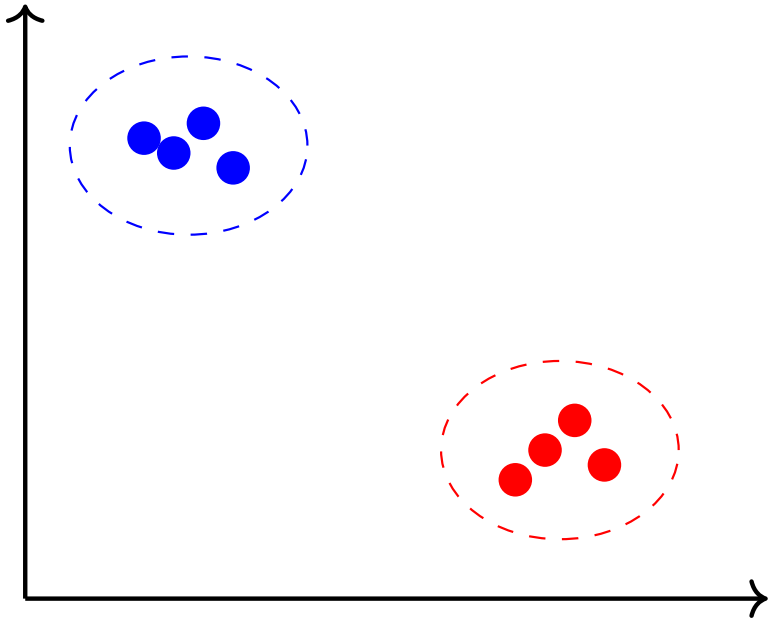


2. Unsupervised Learning

We only have input data X (No labels). The goal is to discover hidden structures or patterns.

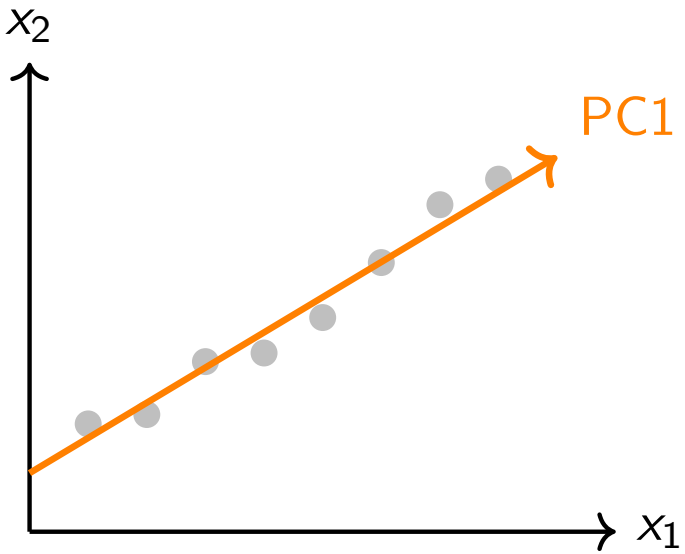
A. Clustering (Grouping)

Grouping similar examples (e.g., customers segmentation)



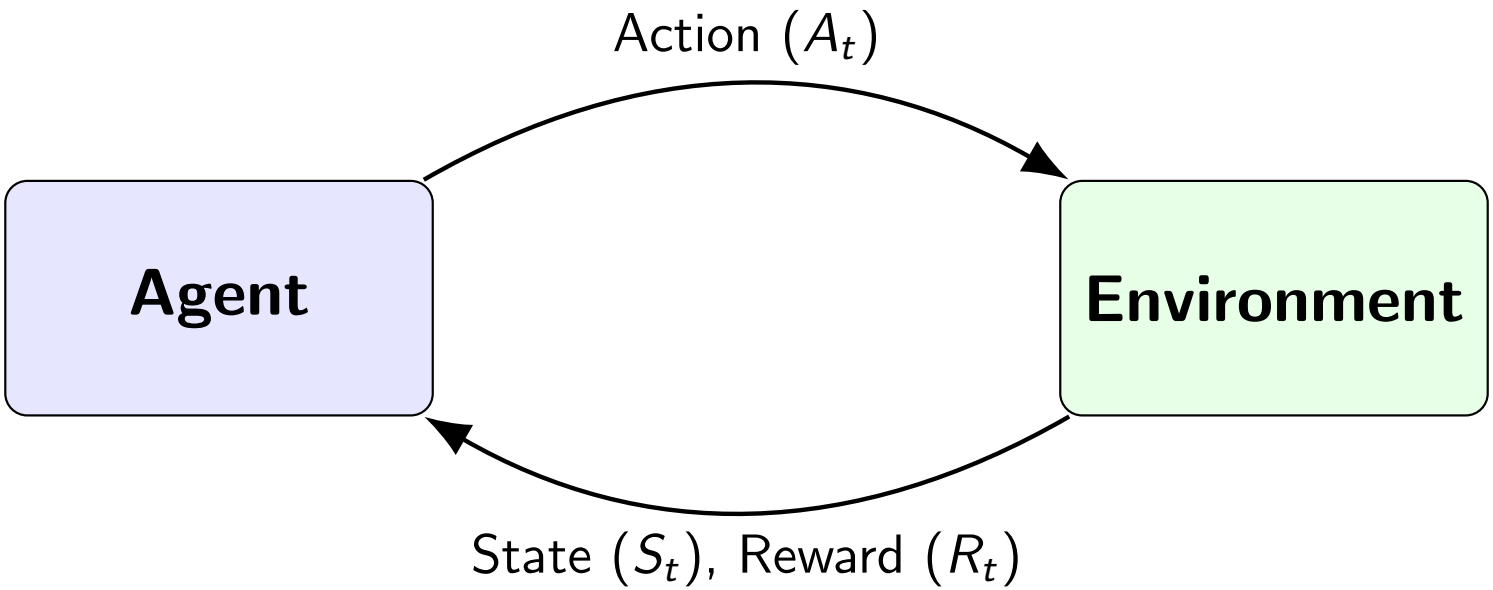
B. Dimensionality Reduction

Reduce the number of features (e.g., $100D \rightarrow 2D$ for visualization)



3. Reinforcement Learning

Learning through trial and error. An **Agent** takes actions in an **Environment** to maximize cumulative **Reward**.



- **Examples**: Chess engines (AlphaZero), Robot navigation, Stock trading bots, LLMs Preference Finetuning (RLHF).

The Three Pillars of Learning

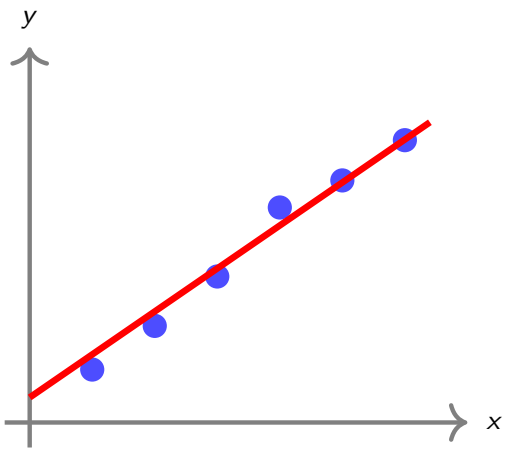
Every supervised learning algorithm consists of three specific components

1. **The Hypothesis (Model)**
2. **The Loss Function**
3. **The Optimizer**

1. The Hypothesis (The Model)

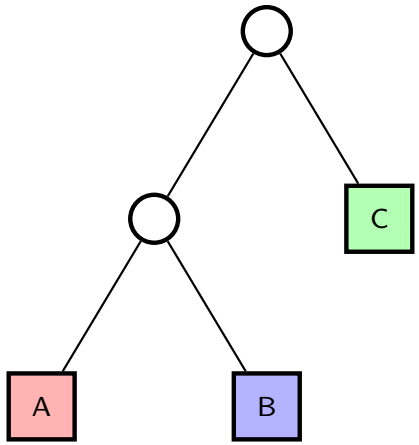
- The mathematical function that maps inputs to predictions.
- **Notation**: $\hat{y} = f(x)$.
- It contains **parameters** (weights) that determine its behavior.
- A model is **"trained"** when these parameters are updated to fit data.
- It can be simple (Linear) or complex (Non-linear).

Linear Regression



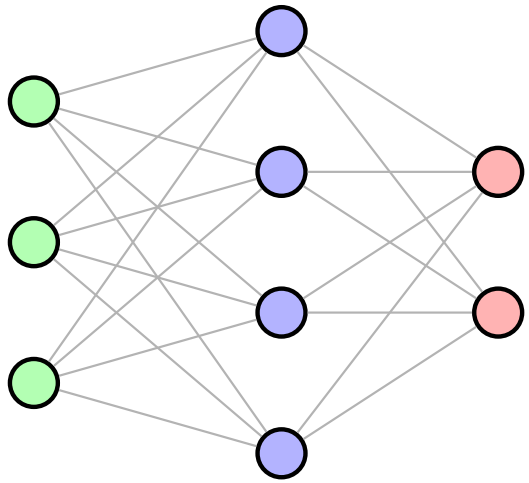
$\hat{y} = wx + b$

Decision Tree



$\hat{y} = \text{tree}(x)$

Neural Network



$\hat{y} = \sigma(W_2\sigma(W_1x))$

2. The Loss Function (The Critic)

Definition: A differentiable function $J(\theta)$ that quantifies the *cost* of the model's errors.

Goal: Find model parameters θ that minimize $J(\theta)$.

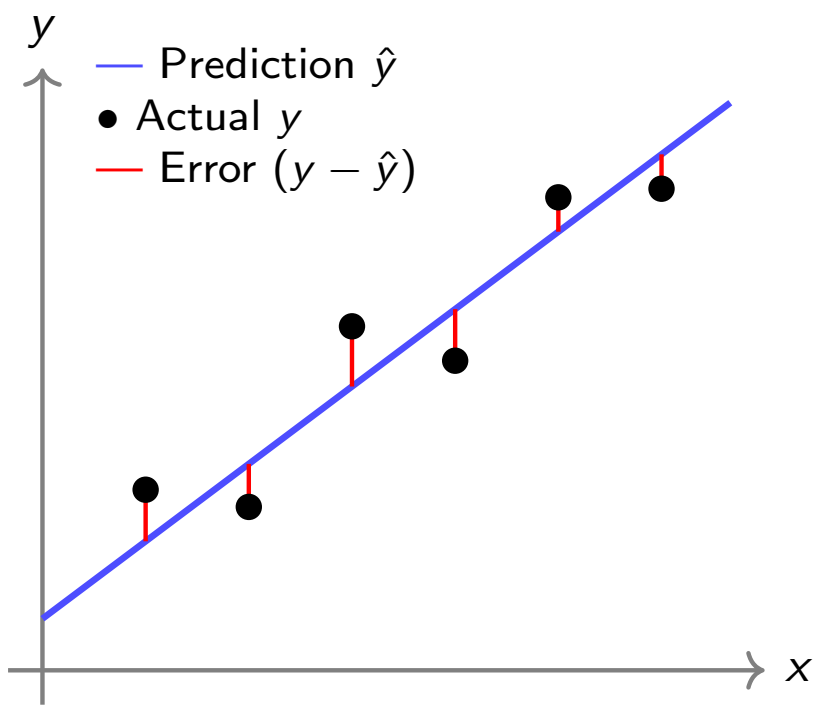
Common Loss Functions:

- **Mean Squared Error (MSE)** (for regression tasks)
- **Cross-Entropy Loss (Logloss)** (for classification tasks)

Lower loss = Better model

- But how can we minimize the loss?

Visualizing Regression Loss

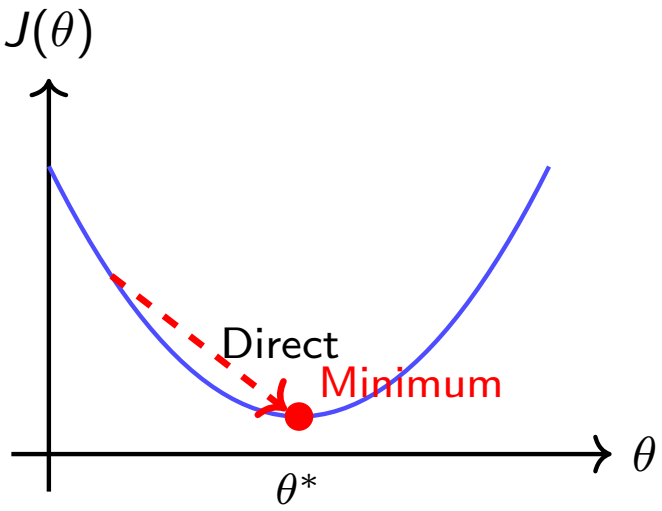


The loss function sums up these red distances (absolute/square).

3. The Optimizer (The Teacher)

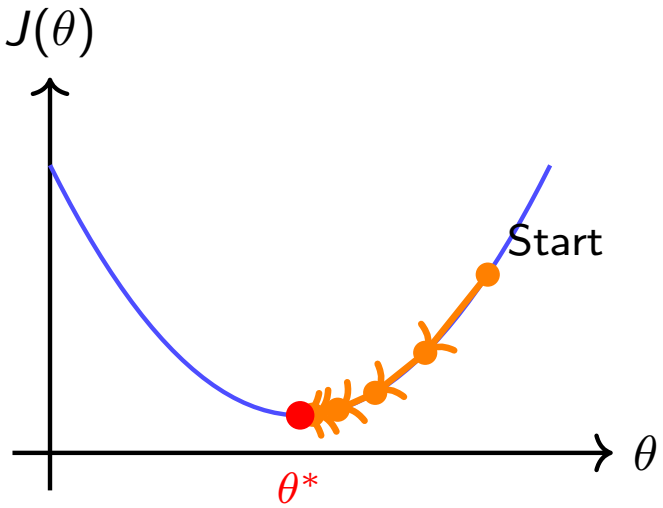
Definition: The algorithm that updates parameters θ to minimize loss $J(\theta)$.

Closed-Form Solution



+ Finds exact solution instantly
– Only works for simple problems
– Impossible for neural networks

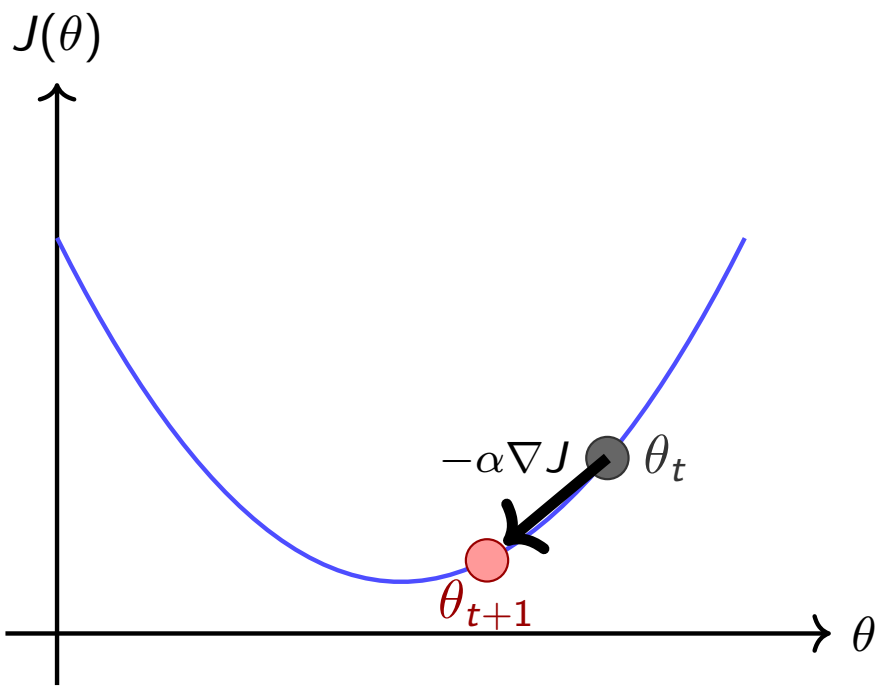
Gradient Descent (Iterative)



+ Works for any model
+ Universal for ML/DL
– Requires many iterations

How Gradient Descent Works

Core Idea: Move parameters in the direction that reduces loss.



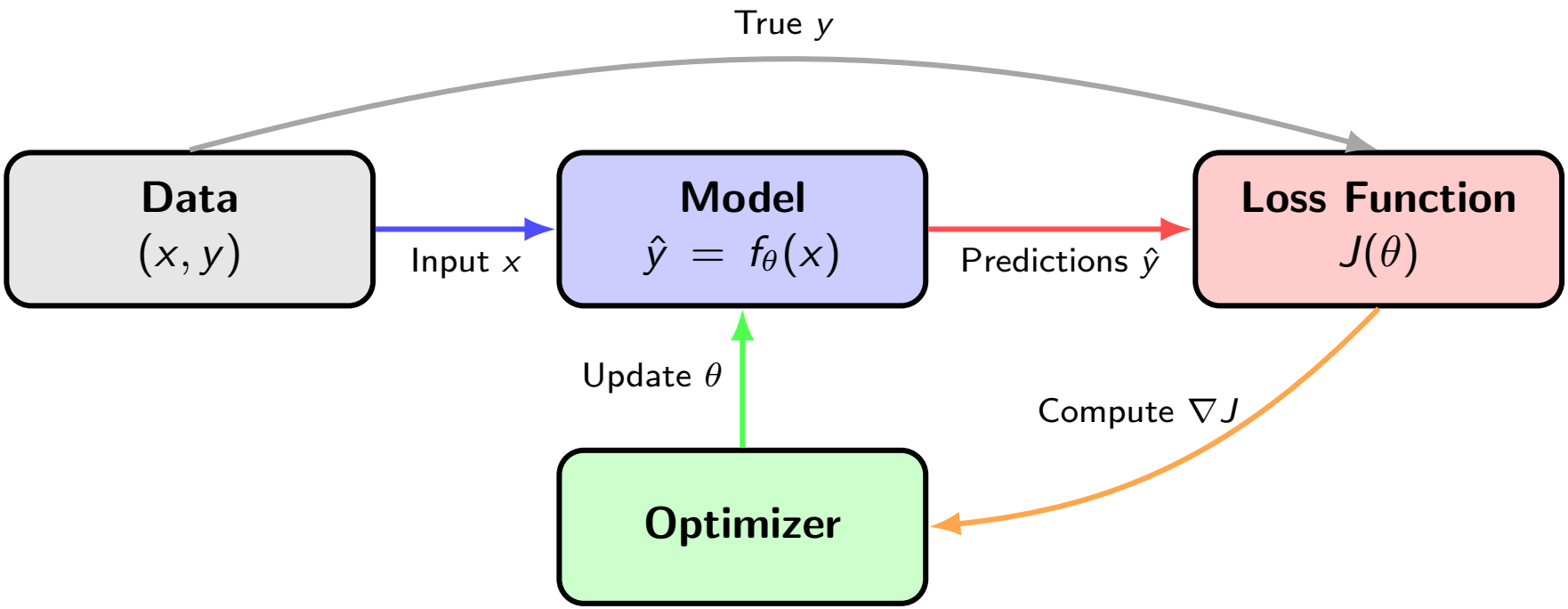
Step-by-Step Process:

1. **Compute Loss**
   Evaluate $J(\theta_t)$ at current parameters
2. **Calculate Gradient**
   Find slope: $\nabla J(\theta_t) = \frac{\partial J}{\partial \theta}$
3. **Update Parameters**
   Move opposite to gradient with a step size (learning rate) α:

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)$$

4. **Repeat**
   Until convergence (loss stops decreasing)

Putting It All Together: The Learning Loop



Forward Pass:

- Data $\rightarrow$ Model $\rightarrow$ Predictions
- Compute Loss

Backward Pass:

- Calculate gradients
- Update parameters

Repeat until loss converges (model learns)!

What Can Go Wrong?

Since our task is to select a learning algorithm and train it on data, there are two main sources of failure:

1. Bad Data

- Not enough data
- Non-representative data
- Poor quality (Noise)
- Irrelevant features

+

2. Bad Algorithm

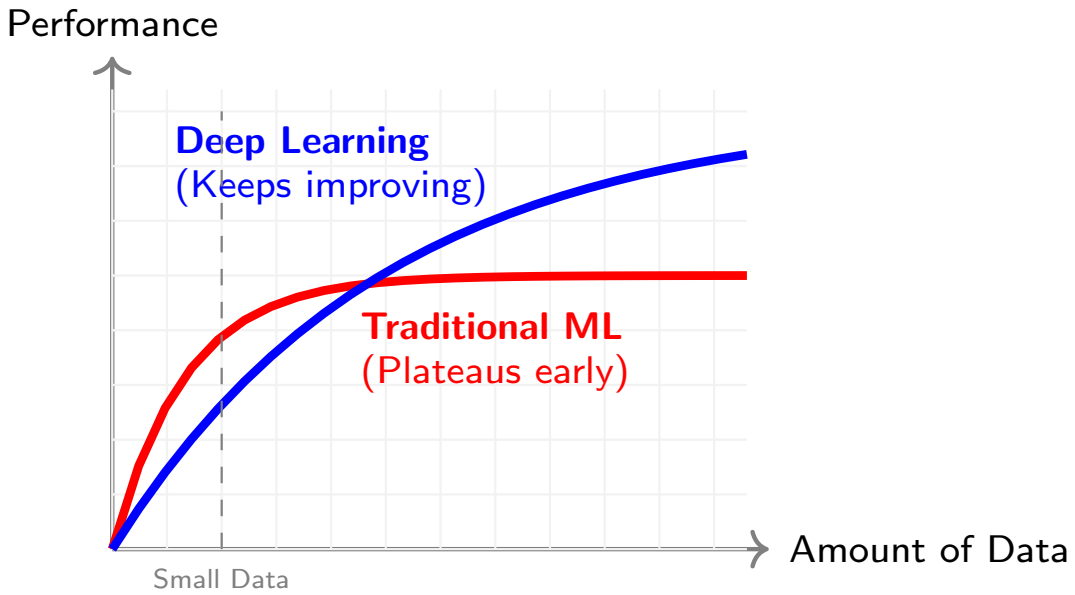
- Overfitting
- Underfitting

Problem 1: Insufficient Data

The Reality Gap

- **Humans:** Can generalize from 1 example (Zero-shot).
- **Machines:** Need thousands to find the signal in the noise.

Without enough data, complex models just memorize the noise.



Problem 2: Non-Representative Data (Bias)

أكاديمية كاوست
KAUST ACADEMY

Definition: Sampling Bias

When your training data does not accurately reflect the real world you want to predict.

Example: The "Happiness" Model

Scenario:

- You want to predict **Happiness** based on **Wealth**.
- But you only collect data from the **Middle Class**.

The Result:

- The model learns: *"More Money = More Happy."*
- **Fail:** It will fail catastrophically for **Billionaires** (where money stops mattering) or **Poverty** (where struggles are different).

Problems 3 & 4: Poor Quality & Irrelevant Features

Poor Quality Data

- **Noise:** Random errors in the data.
- **Outliers:** Extreme values that distort the pattern.
- **Missing Values:** Empty cells in your spreadsheet.

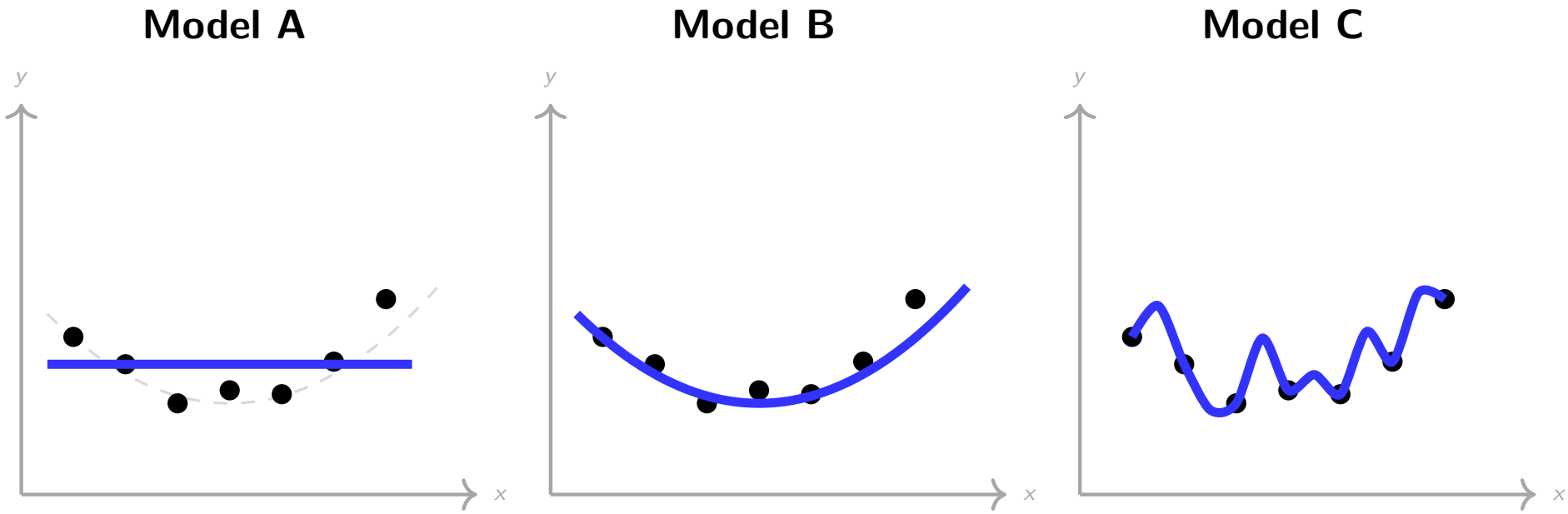
Solution: Data Cleaning (takes 80% of a Data Scientist's time).

Irrelevant Features

- *"Garbage In, Garbage Out"*
- If you try to predict "Price of a Car" using "Color of the seats", the model will fail.

Solution: Feature Engineering (Selection & Extraction).

Algorithm Problem: Generalization

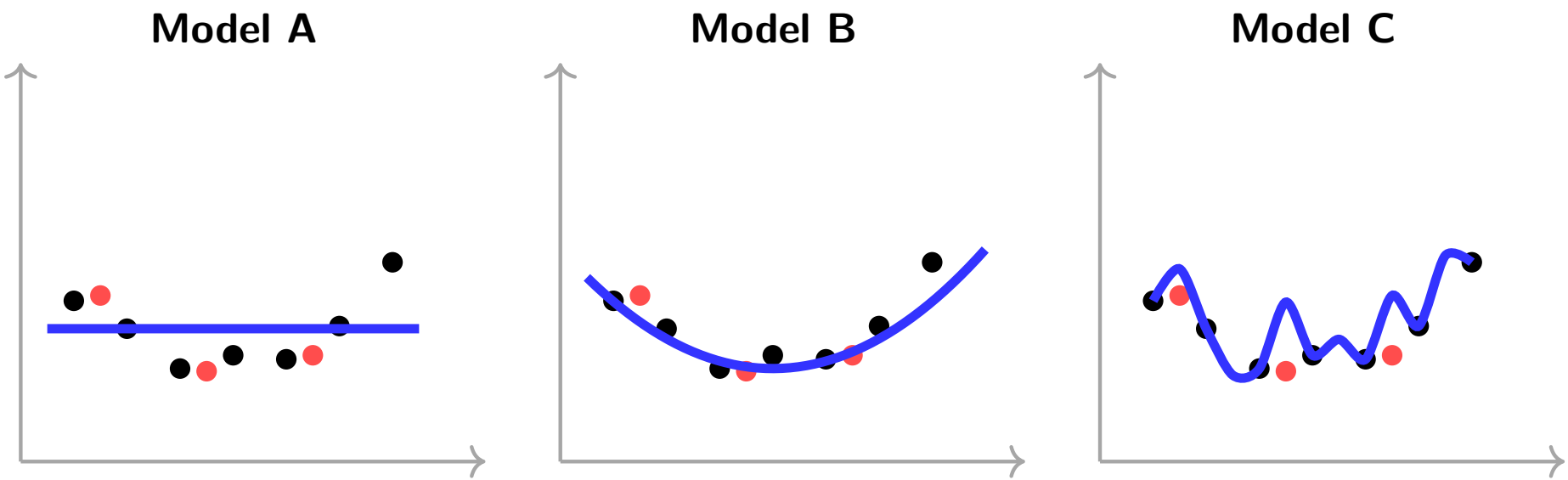


Train MSE ≈ 0.20 | Train MSE ≈ 0.06 | Train MSE ≈ 0.00

All three models see the same training points but fit them in very different ways. Which one should we pick?

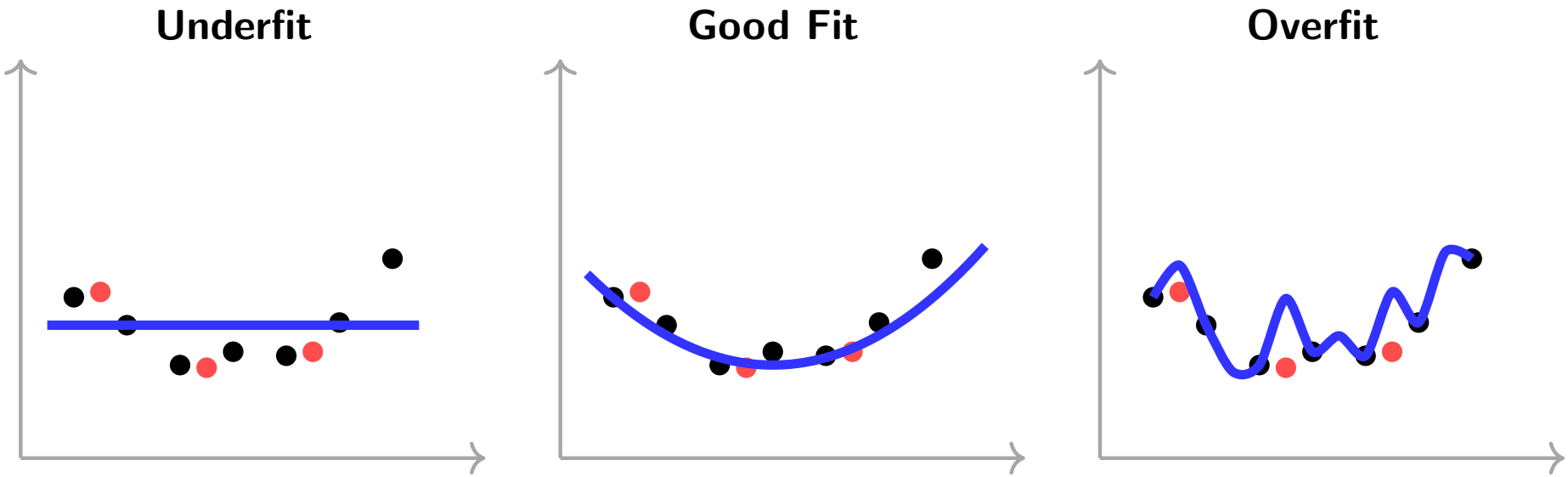
What Do We Really Want?

- We care about how the model behaves on **new unseen data**, not only on the points it has already seen
- Imagine we now collect a few fresh samples from the same process



On these new red points, Model B stays close, while Model A and Model C miss more.

Underfit vs Good Fit vs Overfit



Too Simple (High Bias)
Ignores patterns

Captures Structure
Balanced

Too Complex (High Variance)
Memorizes noise

Underfit vs Good Fit vs Overfit

- **Underfitting**
  - Model is too simple for the true pattern
  - It cannot reach low error even with a lot of data
- **Overfitting**
  - Model is too flexible and learns noise
  - It fits training points very well but is unstable on new points
- **Good fit**
  - Model has just enough flexibility to capture useful structure
  - It behaves similarly on seen and unseen data

What is Generalization?

Definition

Generalization is the ability of a model to perform well on data it has **never seen before**.

High Training Accuracy $\neq$ Learning

- **Memorization (Capacity):** Modern models are powerful enough to memorize the entire training set, acting like a "lookup table" rather than a learner.
- **Shortcuts (Laziness):** If a simple noise pattern correlates with the target, the model will learn that instead of the complex true signal.

Examples:

Domain	Intended Pattern	Reality
Vision	Recognizes animal features (ears, snout).	Learns that **snow background** = Wolf.
Tabular	Evaluates financial history (income, debt).	Learns that **Application Font/ID** = High Risk.
NLP	Analyzes sentence semantics and sentiment.	Memorizes specific **names** (e.g., "Spielberg").
RL	Learns strategy to win the game/level.	Finds a **glitch** or loops indefinitely to farm points.

Addressing Overfitting

Since we cannot trust the training error, we need a systematic approach to validate our models.

1. Detect:
Splitting Data
(Holdout, K-Fold)

2. Measure:
Metrics
(Accuracy, MSE, F1, AUC)

3. Mitigate:
Strategies
(Regularization, More Data, Better Features)

Addressing Overfitting

Since we cannot trust the training error, we need a systematic approach to validate our models.

1. Detect:
Splitting Data
(Holdout, K-Fold)

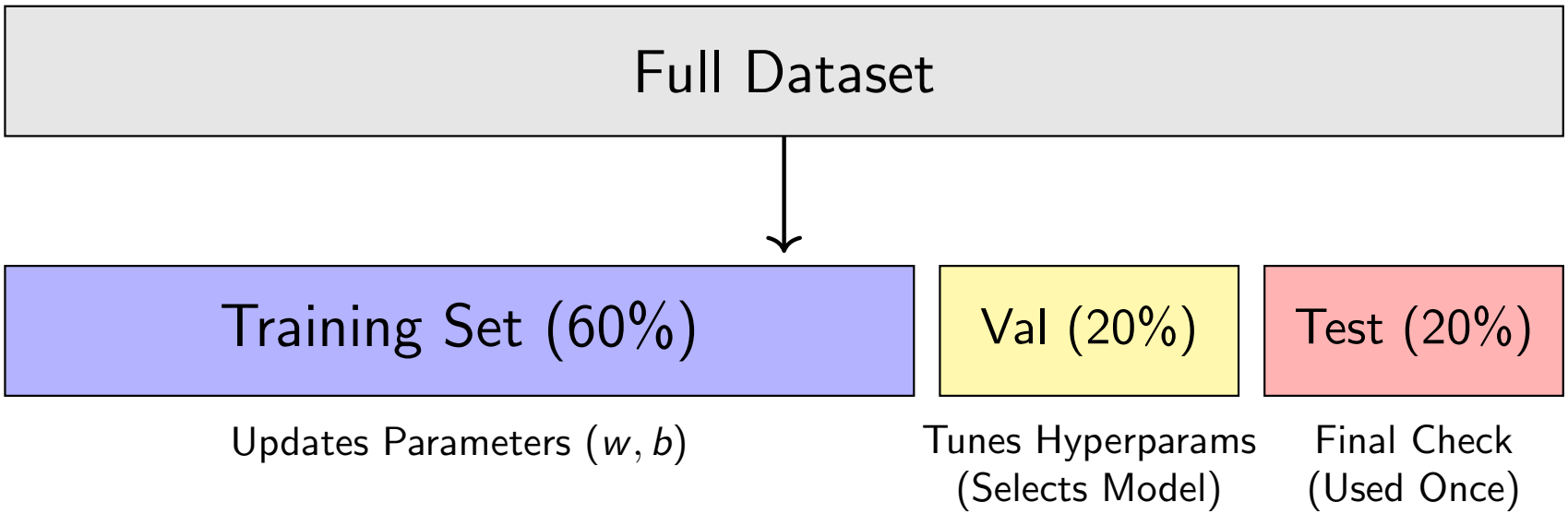
2. Measure:
Metrics
(Accuracy, MSE, F1, AUC)

3. Mitigate:
Strategies
(Regularization, More Data, Better Features)

$\rightarrow$ *Step 1: How to simulate "unseen data" using Splits.*

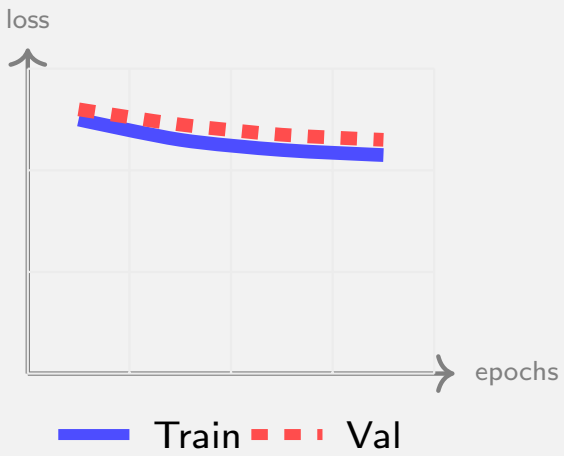
Detecting Overfitting: The Holdout Split

- We cannot trust performance on training data (memorization).
- **Solution:** Hide a piece of data to simulate the "Future."



Diagnosis: Loss Curves

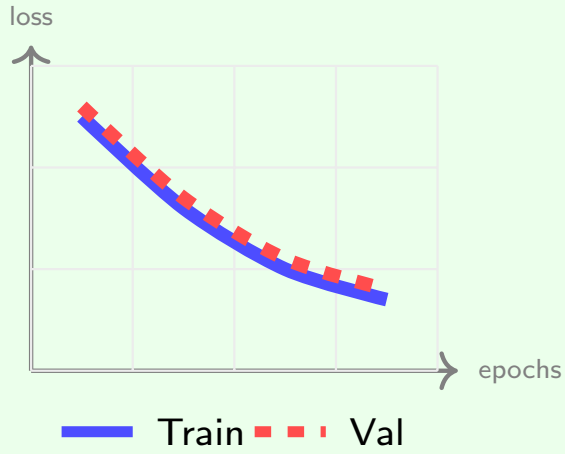
Underfitting



Both losses remain **high**

Model too simple

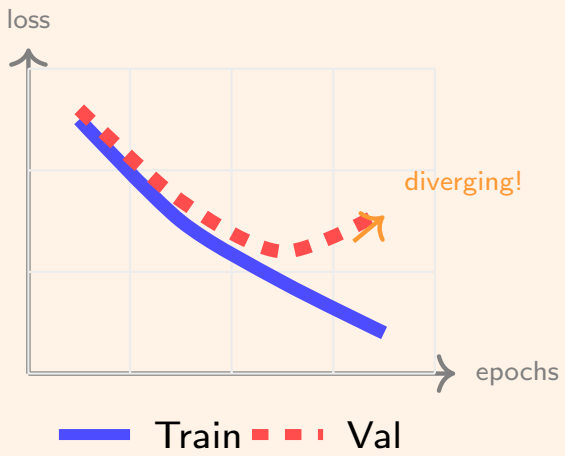
Good Fit



Both losses are **low**

Curves track down closely

Overfitting

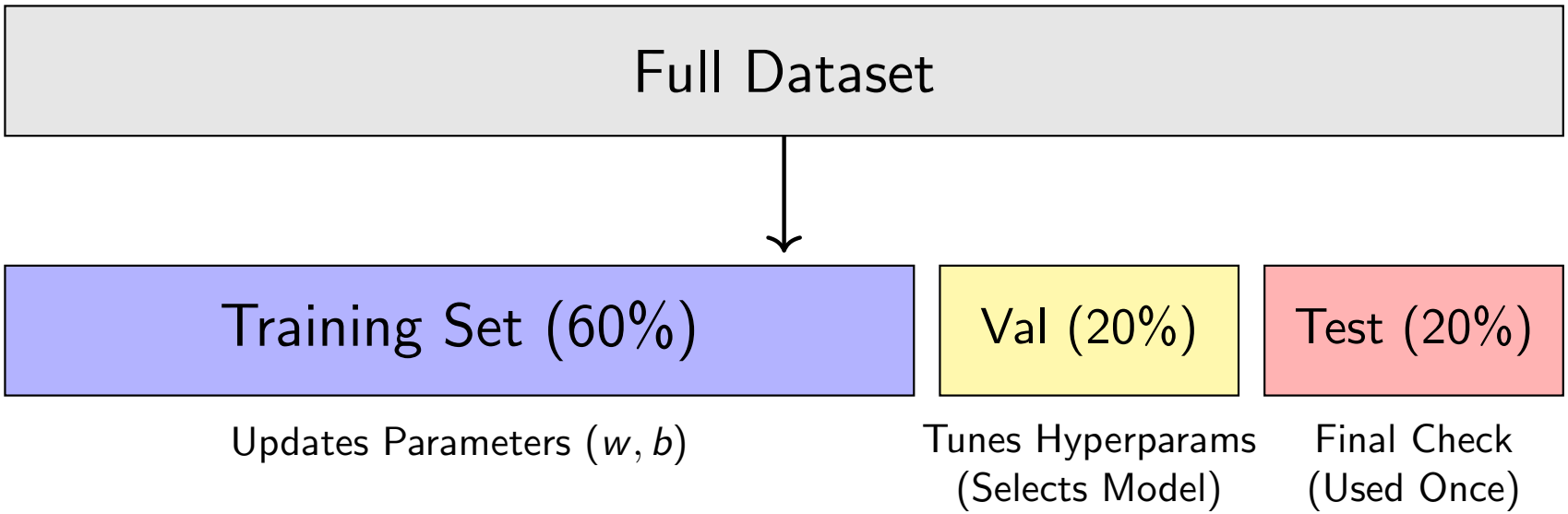


Train loss ↓, Val loss ↑

Model started to overfit!

The gap between training and validation loss reveals overfitting!

Detecting Overfitting: The Holdout Split



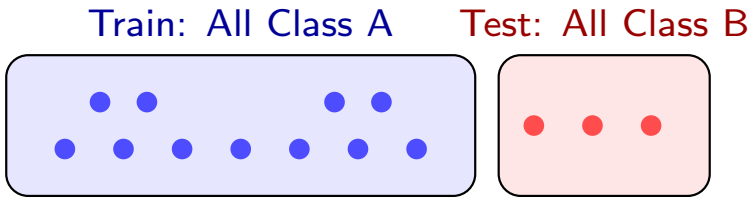
- While this split works conceptually, we may encounter several problems in practice...

The Problem: Ordered Data

Data is Often Ordered

- Real-world data is often collected sequentially (by date, time, or class).
- If we split data directly, the model trains on one pattern but gets tested on a completely different one.

1. Ordered Split (Biased)



Model never saw Class B!

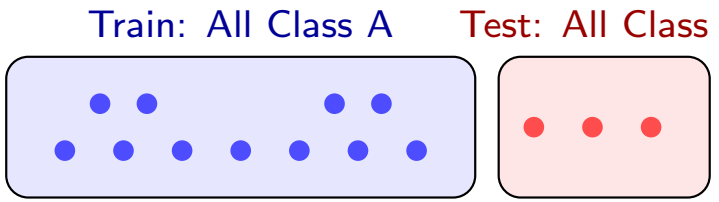
The Problem: Ordered Data

Data is Often Ordered

- Real-world data is often collected sequentially (by date, time, or class).
- If we split data directly, the model trains on one pattern but gets tested on a completely different one.

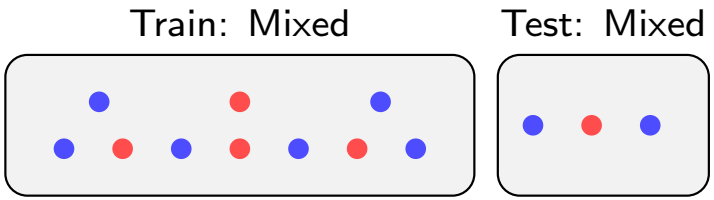
Solution:
Shuffle the data before splitting!

1. Ordered Split (Biased)



Model never saw Class B!

2. Shuffled Split (Correct)



Representative Split ✓

The Problem: Ordered Data

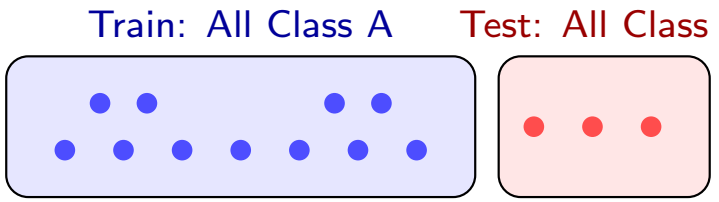
Data is Often Ordered

- Real-world data is often collected sequentially (by date, time, or class).
- If we split data directly, the model trains on one pattern but gets tested on a completely different one.

Solution:
Shuffle the data before splitting!

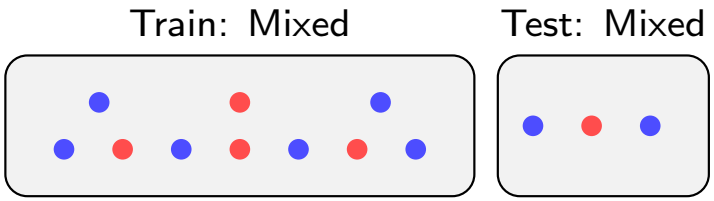
This helps, but what if data is small?

1. Ordered Split (Biased)



Model never saw Class B!

2. Shuffled Split (Correct)



Representative Split ✓

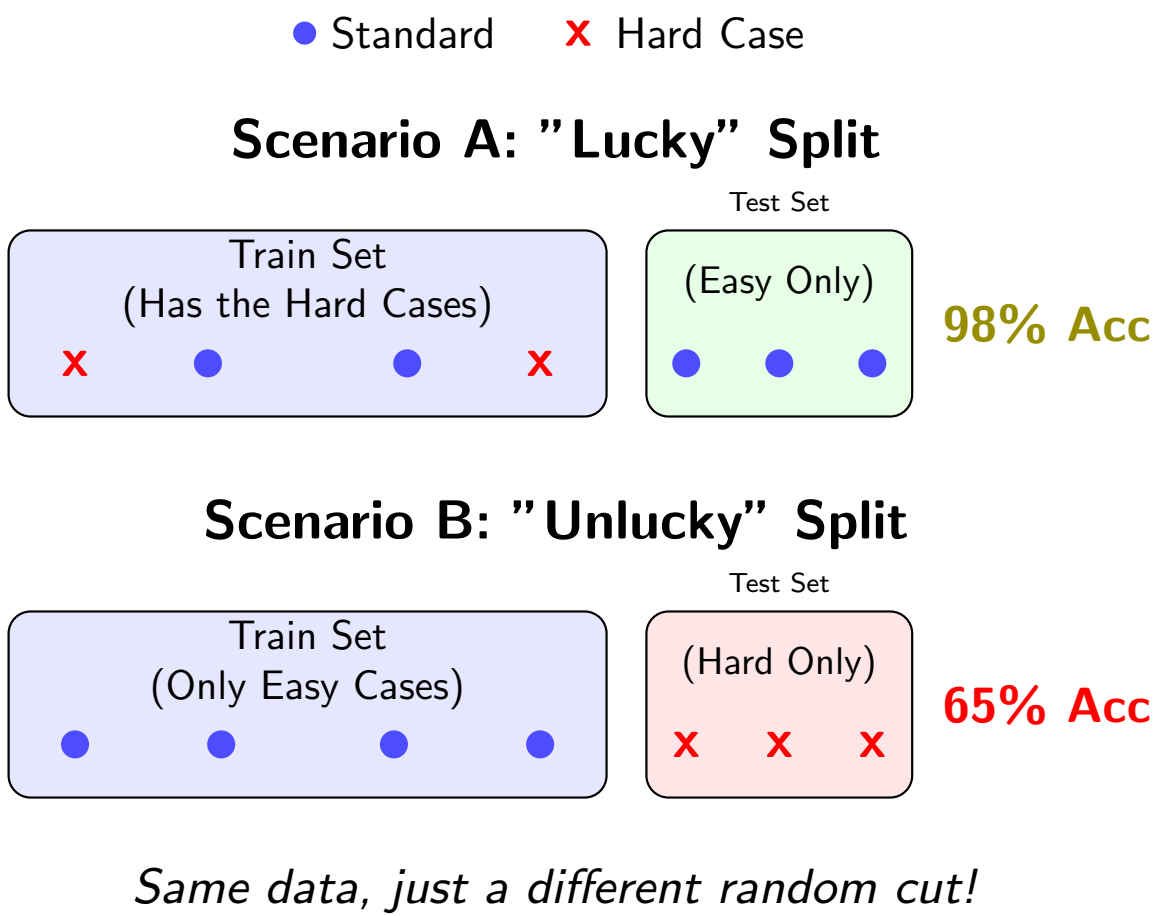
What If Data is Small?

The Remaining Problem

Shuffling fixes the *order* bias, but with small datasets, a single random split can still be **unlucky**.

The Law of Small Numbers:

- 20% of a small dataset is a tiny sample.
- It is statistically likely that this small sample will not match the population perfectly.
- **Result:** We get "Lucky" or "Unlucky" splits.

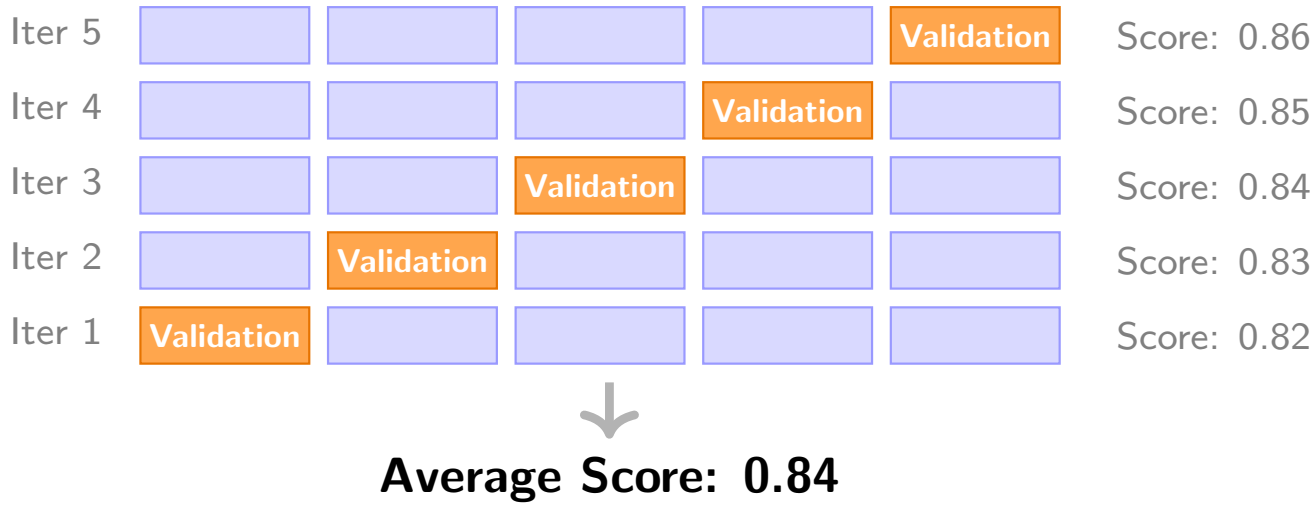


Same data, just a different random cut!

The Solution: K-Fold Cross Validation

Use ALL Data for Testing (Across Multiple Rounds)

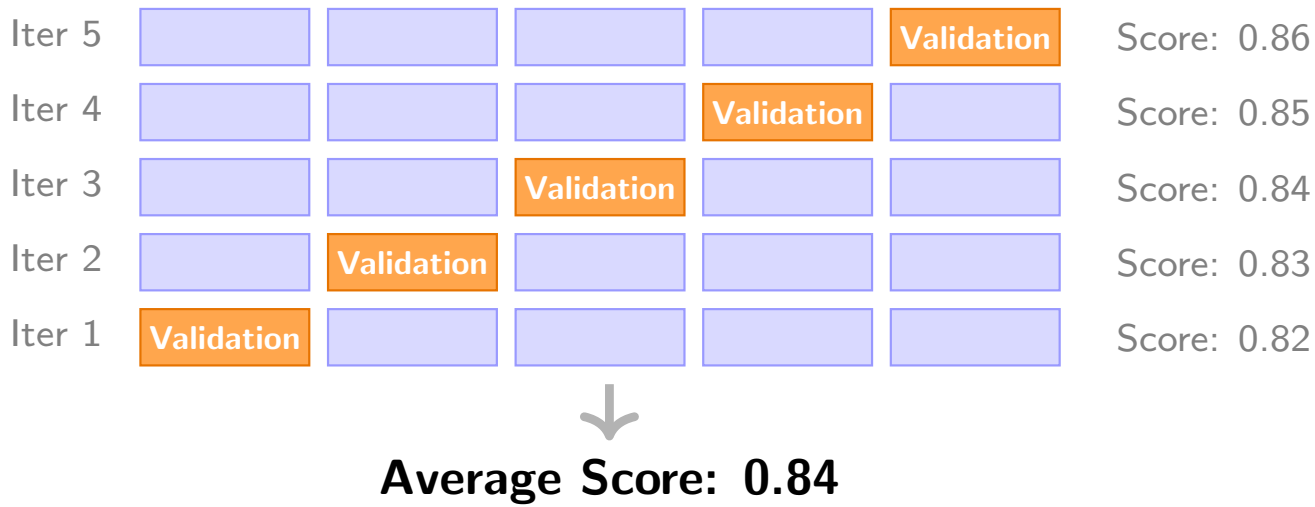
1. Split shuffled data into K equal folds (e.g., $K = 5$).
2. Train on $K - 1$, test on the remaining 1.
3. **Rotate** until every fold has been the test set exactly once.
4. **Average** the scores.



The Solution: K-Fold Cross Validation

Why K-Fold Works

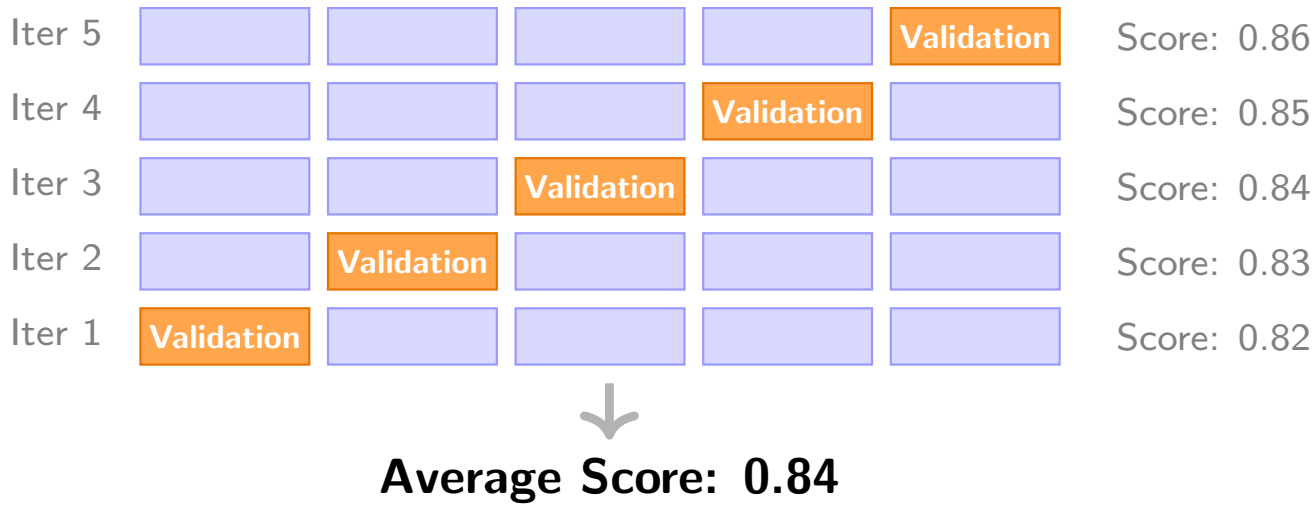
Every data point serves as a test sample exactly once. We evaluate on **100%** of the data instead of just 20%, removing the luck factor.



The Solution: K-Fold Cross Validation

Why K-Fold Works

Every data point serves as a test sample exactly once. We evaluate on **100%** of the data instead of just 20%, removing the luck factor.



- Only one problem left...

The Hidden Trap: Imbalanced Data

What is Imbalance?

- When one class dominates the target (e.g., 99% Healthy, 1% Sick).

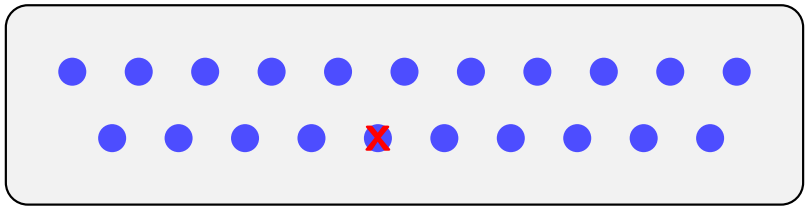
The Accuracy Paradox

If you have 99 blue dots and 1 red dot, a model that **always guesses Blue** has 99% Accuracy!

But it failed to find the one thing we cared about!

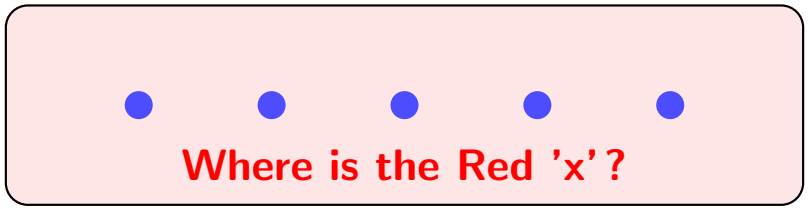
1. Target Distribution

(95% Blue, 5% Red)



2. Random Split

Test Set Target Distribution

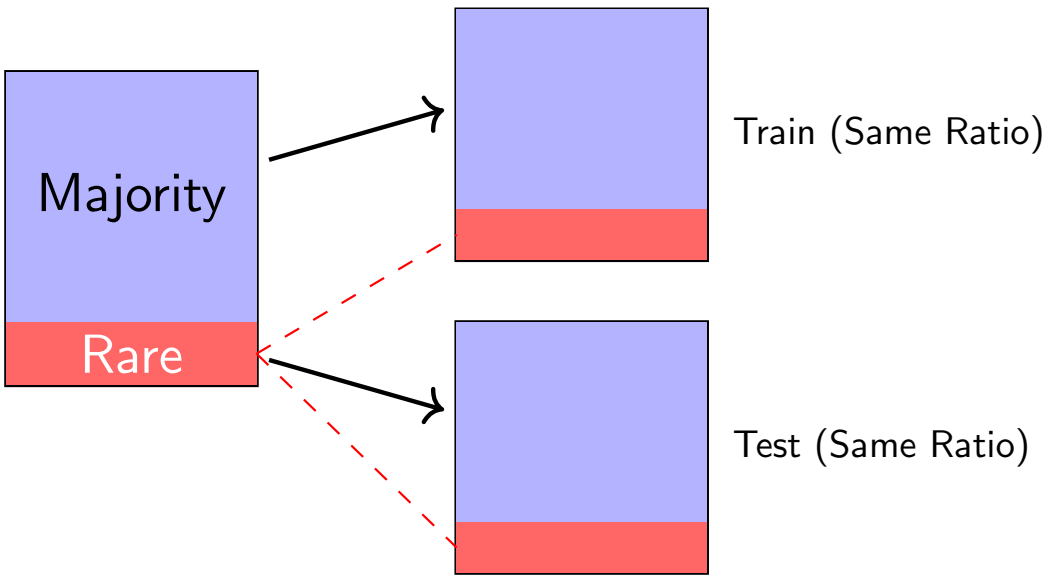


The Fix: Stratified Splitting

أكاديمية كاوست
KAUST ACADEMY

Definition

Stratification forces the Train and Test splits to preserve the **same class ratios** as the original dataset.



Why it works:

- It guarantees the Test set includes the "Red 'x'".
- It allows us to calculate metrics like Recall and F1-Score properly.

Tip:
Use **Stratified K-Fold** for the best of both worlds!

Addressing Overfitting

Since we cannot trust the training error, we need a systematic approach to validate our models.

1. Detect:
Splitting Data
(Holdout, K-Fold)

2. Measure:
Metrics
(Accuracy, MSE, F1, AUC)

3. Mitigate:
Strategies
(Regularization, More Data, Better Features)

$\rightarrow$ *Step 1: How to simulate "unseen data" using Splits.*
$\rightarrow$ *Step 2: How to measure performance using Metrics.*

Measuring Performance: Metrics

Definition

A **Metric** is a quantifiable measure used to evaluate how well a model is performing on a specific task.

Loss Function vs. Metric

Loss (For the Machine)

- Used during **Training**.
- Must be **Differentiable** (smooth) so we can calculate gradients.
- *Example: Cross-Entropy, MSE.*

Metric (For the Human)

- Used during **Evaluation** (Validation/Test).
- Must be **Interpretable** by humans (business value).
- *Example: Accuracy, F1-Score, Dollar Amount.*

Measuring Performance: Metrics

أكاديمية كاوست
KAUST ACADEMY

Definition

A **Metric** is a quantifiable measure used to evaluate how well a model is performing on a specific task.

Loss Function vs. Metric

Loss (For the Machine)

- Used during **Training**.
- Must be **Differentiable** (smooth) so we can calculate gradients.
- *Example: Cross-Entropy, MSE.*

Metric (For the Human)

- Used during **Evaluation** (Validation/Test).
- Must be **Interpretable** by humans (business value).
- *Example: Accuracy, F1-Score, Dollar Amount.*

Metrics differ based on the task: Regression (Numbers) vs. Classification (Classes).

Metrics for Regression: MAE vs. MSE

Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n}\sum |y - \hat{y}|$$

- Measures average absolute distance.
- **Robust:** Treats all errors linearly.

Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum (y - \hat{y})^2$$

- Squares the difference.
- **Sensitive to outliers:** Large errors are punished heavily ($10^2 = 100$).
- Easy to differentiate.

Interpreting the Score

Scenario: You are training a model to predict **House Prices**.

$MAE = 10{,}000$

What it means:

- "On average, my prediction is off by **$10,000**."

Easy to explain.

$MSE = 100{,}000{,}000$

What it means:

- The unit is **"Dollars Squared"** ($\$^2$).
- Hard to interpret directly!
- **Why use it?** To force the model to focus on fixing the massive mistakes first.

Common Fix: Take the square root ($\sqrt{MSE}$) to get **RMSE**, which brings the units back to dollars.

Metrics for Classification: Accuracy

أكاديمية كاوست
KAUST ACADEMY

Definition

The percentage of predictions that our model got correct.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Metrics for Classification: Accuracy

أكاديمية كاوست
KAUST ACADEMY

Definition

The percentage of predictions that our model got correct.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Q: Can we use it as a loss?

Metrics for Classification: Accuracy

أكاديمية كاوست
KAUST ACADEMY

Definition

The percentage of predictions that our model got correct.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Q: Can we use it as a loss? **No, non-differentiable.**

Metrics for Classification: Accuracy

أكاديمية كاوست
KAUST ACADEMY

Definition

The percentage of predictions that our model got correct.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Accuracy works well until you work with an imbalance target. Can you figure out why?

When Accuracy Fails: Imbalanced Data

Imagine a dataset where **99%** of samples are Class A and only **1%** are Class B.

Scenario:

- We build a "Lazy Model" that ignores the data and always predicts **"Class A"**.

Result:

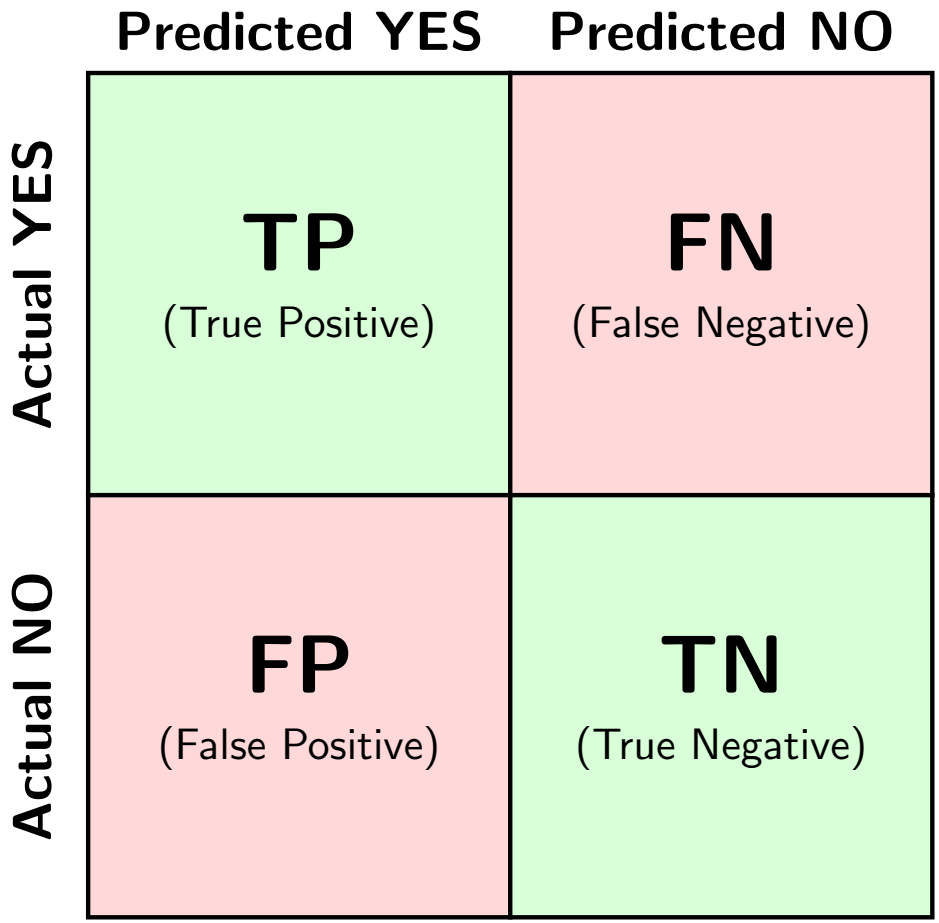
99%

Accuracy

The model is useless (it never finds Class B), but Accuracy says it's amazing. We need better metrics to evaluate its performance against the minority class.

The Foundation: Confusion Matrix

To understand the errors, we break them down into a table.



Metric: Precision

أكاديمية كاوست
KAUST ACADEMY

Question

"Of all the times the model predicted **YES**, *how often was it right?"*

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \textbf{False Positives}}$$

When to use it?

- When **False Positives** are expensive/annoying.
- **Example: Spam Filter.**
- We do **not** want to mark a real email (work, family) as Spam. Better to let a few spam emails through than delete a real one.

Metric: Recall

أكاديمية كاوست
KAUST ACADEMY

Question

"Of all the **Real YES** *cases in the world, how many did we find?"*

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \textbf{False Negatives}}$$

When to use it?

- When **False Negatives** are dangerous/fatal.
- **Example: Cancer/Fraud Detection.**
- We must find **every** sick patient. It is okay to raise a false alarm (do more tests), but we cannot miss a sick person.

Metric: F1-Score

- Often we need a balance between Precision and Recall.
- If you optimize only one, the other usually suffers (Precision-Recall Tradeoff).

F1-Score (Harmonic Mean of Precision and Recall)

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Used for imbalanced data.

Why Harmonic Mean not Arithmetic Mean?

- It forces the model to be good at **both**. If Precision is 100% but Recall is 0%, the Average is 50%, but the **F1-Score is 0**.

How to know if a specific score is good?

Question: My F1-Score is 40%. Is that good?

Baselines

Question: My F1-Score is 40%. Is that good?

Answer: Compare it to a "Dummy" Baseline!

Classification Baseline

- Always predict the **Majority Class**.
- If your model can't beat this, it hasn't learned anything.

Regression Baseline

- MSE baseline: Use the **Mean**.
- MAE baseline: Use the **Median**.
- If your model can't beat this, you are doing worse than just guessing the average/median!

Addressing Overfitting

Since we cannot trust the training error, we need a systematic approach to validate our models.

1. Detect:
Splitting Data
(Holdout, K-Fold)

2. Measure:
Metrics
(Accuracy, MSE, F1, AUC)

3. Mitigate:
Strategies
(Regularization, More Data, Better Features)

→ *Step 1: How to simulate "unseen data" using Splits.*
→ *Step 2: How to measure performance using Metrics.*
→ *Step 3: How to mitigate overfitting.*

Hyperparameters: Capacity & Optimization

Hyperparameters are settings chosen **before** training.

- **Structural:** Control model complexity/capacity (e.g., Tree Depth, Layers).
- **Optimization:** Control the training process (e.g., Learning Rate).

A. Structural Params (Controlling Overfitting)

Rigid (Risk: Underfitting)

- Poly Degree $= 1$ (Line)
- Tree Depth $= 1$ (Stump)
- NN Layers $= 1$ (Shallow)

Flexible (Risk: Overfitting)

- Poly Degree $= 20$ (Wiggly)
- Tree Depth $= 100$ (Deep)
- NN Layers $= 20$ (Deep)

Hyperparameters are settings chosen **before** training.

- **Structural:** Control model complexity/capacity (e.g., Tree Depth, Layers).
- **Optimization:** Control the training process (e.g., Learning Rate).

B. Optimization Params (The "Learning" Process)

Learning Rate (LR): The "Step Size" of the update.

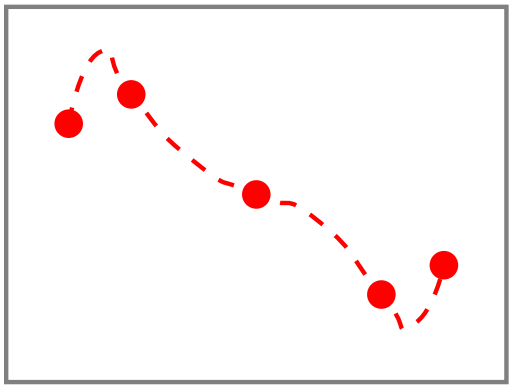
Too Low	**Just Right**	**Too High**
→ Slow/Stuck	→ Converges	→ Unstable/Diverges

Increase Data Size

The Logic

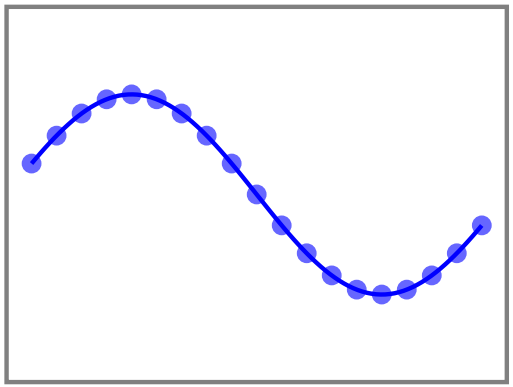
Overfitting happens when the model memorizes "noise" in the gaps between data points. **Solution:** Fill the gaps with more data.

1. Small Data (Sparse)



Easy to memorize path

2. Large Data (Dense)



Forced to see the curve

With more data, the "wiggly" memorization path becomes impossible. The model **must** learn the true underlying curve (Signal).

Reduce Features

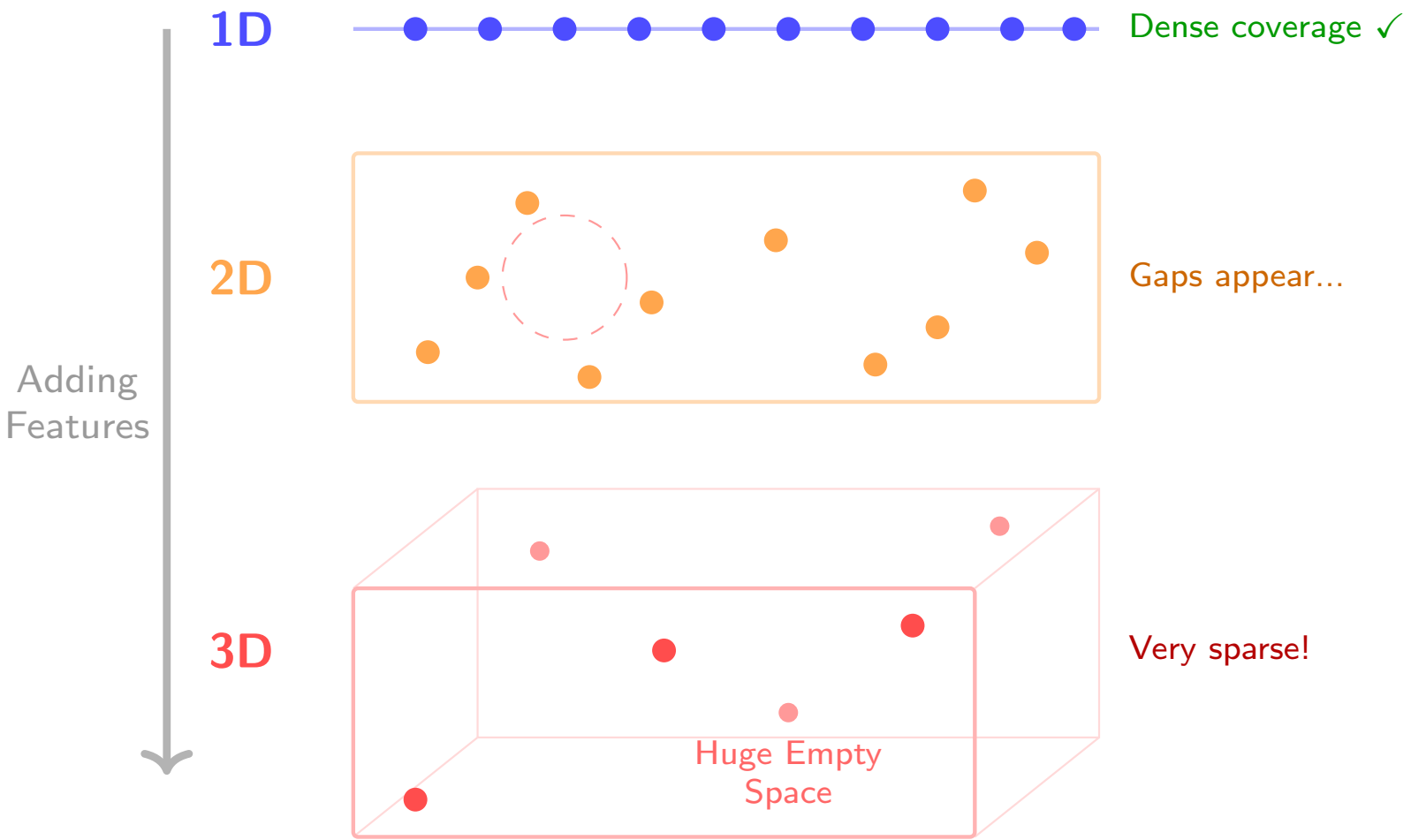
أكاديمية كاوست
KAUST ACADEMY

The Curse of Dimensionality

As you add more features (dimensions), the space grows exponentially. Your data becomes sparse, creating empty regions where the model **hallucinates patterns**.

The Curse of Dimensionality

Visualizing the problem: Same 10 samples, increasing dimensions.



The Curse of Dimensionality

أكاديمية كاوست
KAUST ACADEMY

What Happens in High Dimensions?

- → **1D:** 10 points densely cover the line.
- → **2D:** Same 10 points spread thin → gaps appear.
- → **3D:** Mostly empty space → the model starts to find fake patterns in the noise just to bridge the gaps.

The Solution: Feature Selection

أكاديمية كاوست
KAUST ACADEMY

What Happens in High Dimensions?

- → **1D:** 10 points densely cover the line.
- → **2D:** Same 10 points spread thin → gaps appear.
- → **3D:** Mostly empty space → the model starts to find fake patterns in the noise just to bridge the gaps.

The Fix: Feature Selection

- Remove irrelevant features (noise) and keep only meaningful signals.
- This should push the data back down to lower dimensions where density is high.

Summary: Diagnosing & Fixing

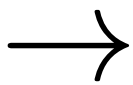
Problem	Diagnosis (Loss)	Mitigation (Fix)
Underfitting	Train High, Val High	↑ Increase Complexity ↑ Add Features
Overfitting	Train Low, Val High	↓ Reduce Complexity ↓ Remove Noise/Features ↑ More Data → Regularization (Next Days)
Good Fit	Train Low, Val Low	✓ All good!

We know that **Good Data** *cures Overfitting. But real-world data is rarely "Good." It is messy, broken, and chaotic.*

Raw Data (Reality)

```
["NaN", "N/A", -500,
"Cat"]
```

Models crash on this.



Clean Data

```
[0.5, 0.2, 0.0, 1.0]
```

Models learn from this.

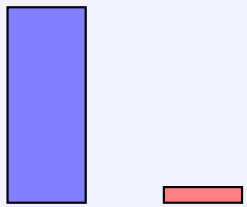
Step 1: EDA (Know Your Data)

أكاديمية كاوست
KAUST ACADEMY

Goal

Before training, become a **Detective**. If you feed garbage in, you get garbage out.

1. Check Target Distribution



- Is it Imbalanced?
- If yes, use Stratified Split & F1-Score.

2. Sanity Checks (Red Flags)

- **Impossible Values:**
- `Age: -5` → Error?
- `Age: 200` → Outlier?
- **Leakage:**
- Does column `X` perfectly predict `y`? (e.g., "TransactionID" contains the label).

Step 2: Cleaning & Scaling

A. Missing Values (NaN) Models cannot do math on "Empty".

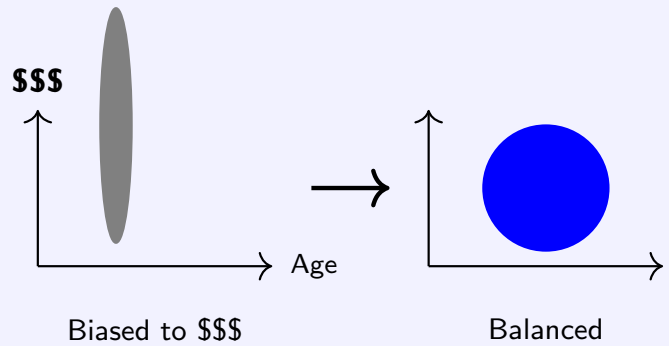
Strategies

1. **Drop:** Delete the row (only if you have lots of data).
2. **Impute:** Fill with Mean/Median.

`[10, ?, 30]` $\rightarrow$ `[10, 20, 30]`

B. Feature Scaling Income (100k) vs. Age (50).

Why Scale?



Models prioritize larger numbers. Scale to $[0, 1]$ (Min-Max Scaler) or $\mu = 0$, $\sigma = 1$ (Standard Scaler).

Step 3: Encoding & Feature Engineering

أكاديمية كاوست
KAUST ACADEMY

C. Encoding (Text → Numbers)
Models speak Math, not English.

1. **Label Encoding** (Ordinal)
   - `["Low", "High"]` → `[0, 1]`
   - Use if order matters.
2. **One-Hot Encoding** (Nominal)
   - `["Cat", "Dog"]` →
   - `Cat: [1, 0]`, `Dog: [0, 1]`
   - Use if no order exists.

D. Feature Engineering Creating *new* signals from existing data.

Example: Date Object

`"2023-12-25"`

↓ *Extract Info* ↓

- `Is_Weekend = 0`
- `Month = 12` (Winter)
- `Days_To_Holiday = 0`

This adds value that the raw date string hid.

Step 4: Fit the Model

We have clean data. We have robust splits. We have metrics.
Now, we need a model.

The Options

- **Regression:** Linear Regression, Random Forest Regressor...
- **Classification:** Logistic Regression, SVM, Neural Networks...

We will explore **Machine Learning** & **Deep Learning** algorithms in the next sessions!

References

- Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*
- Andrew Ng, *Machine Learning Specialization* (Coursera/DeepLearning.AI)

Slides contributed by Mohamed Eltayeb