Classification
A classification task is when you want to predict a category (not a number).

What Makes Classification Hard?
Sometimes the relationship between the predictors and the class is:

- Clear and separable — easy to classify.
- Overlapping — multiple possible boundaries.
- Nonexistent — no useful relationship (maybe wrong or missing features).

Key idea: "All models are wrong, but some are useful."
So even imperfect classifiers can still be useful if they capture most trends.

What Makes a "Good" Predictor?
A predictor (or feature) is good if:

- It is related to the target (helps separate classes).
- It is not redundant with other features.

Example:
If both "height in inches" and "height in centimeters" are used — they're redundant.
If "height" and "shoe size" are used to predict gender — those are correlated but not identical, which can help.

Measuring Relationships: Correlation
Correlation tells us how linearly related two variables are. For continuous X and Y:
$r = cov(x, y) / \sigma x \sigma y$

r = 1: perfect + relationship
r = -1: perfect - relationship
r = 0: no linear relationship

But — note! No correlation doesn't mean no relationship. A curved or nonlinear pattern can still have r = 0.

When Y Is a Class (Not Continuous)
If Y is categorical, we can't use the normal correlation formula. Instead, we handle it differently depending on the type of class:

Y type	Example	How to measure relation with X
Nominal (no order)	{Yes, No}, {Red, Blue, Green}	Compare **means of X** across categories
Ordinal (ordered)	{Terrible, Bad, OK, Good, Great}	Compare **ranks** of X — use **Spearman correlation**

Spearman Correlation Example
We replace each variable by its rank (1st, 2nd, 3rd, …). Then we measure how similar the ranks are between X and Y. (rank here is the order of the magnitude …. so ranks of x = [10, 0, 20, 30, 40] would be [2, 1, 3, 4, 5].

X	Y
10	1
20	0
30	2
40	3
50	4

We replace x with ranks of x and reorder the rows by rank of x.

R(X)	R(Y)
1	2
2	1
3	3
4	4
5	5

Correlation is then found

Correlation vs. Causation
Even if two things move together (high correlation), that doesn't mean one causes the other.

Temperature ↔ Ice Cream Sales
- They're positively correlated.
- But it's not that ice cream causes heat — rather, a third factor (season) explains both.

Sleeping with shoes on ↔ Waking up with a headache
- Correlated, but the real cause is going to bed drunk.
- That's the confounding variable.

Correlation shows association, not causation. To prove causation, you'd need experiments or control groups (e.g., A/B tests, clinical trials).

Simpson's Paradox
Sometimes the direction of a relationship reverses when you group or ungroup data.

Example: Across universities, women might appear less likely to be accepted — but within each department, they might actually have higher acceptance rates.

So …………… always look at data both overall and within groups — aggregation can hide or flip trends.

Instance-Based Classifiers
Instead of learning explicit rules or parameters, some models store examples and use them directly to make predictions. That's what KNN does — it's called an instance-based or lazy learner.

How It Works

1. Training phase:
   - There's basically no training.
   - You just store all your labeled examples (the dataset is the model).

2. Prediction phase: to classify a new point x_{new} :
   - Compute its distance to every training point.
   - Find the k nearest neighbors (the k smallest distances).
   - Vote among their labels to decide the class.

Common distance measures: euclidean distance, manhattan distance.

<u>All features should be on similar scales</u> — otherwise one large-ranged variable (like "income") dominates the distance. Common scalings: normalization (0-1 scaling) and standardization (z-score).

Aggregation Rules: once the k neighbors are chosen:

- Majority rule: each neighbor votes equally.
- Weighted majority: closer neighbors get higher weight, e.g.

Choosing k:

- Small k → very flexible, can overfit (sensitive to noise).
- Large k → smoother, more general, may blur class boundaries.

Understanding the Effect of k

If performance worsens when k increases …. it usually means the neighborhood became too large, mixing points from different classes. The model is becoming too simple — it's averaging too much → underfitting.

Conversely, very small k (like 1) can overfit, because the prediction is based on just one neighbor (possibly a noisy or outlier point).

The Bias–Variance Tradeoff in KNN

k	Model behavior	Problem type
Small (e.g. 1–3)	High variance (fits noise)	Overfitting
Large (e.g. 20+)	High bias (too smooth)	Underfitting

You pick k where performance on validation data is best — usually found via cross-validation.

Visual Intuition: Decision Boundaries

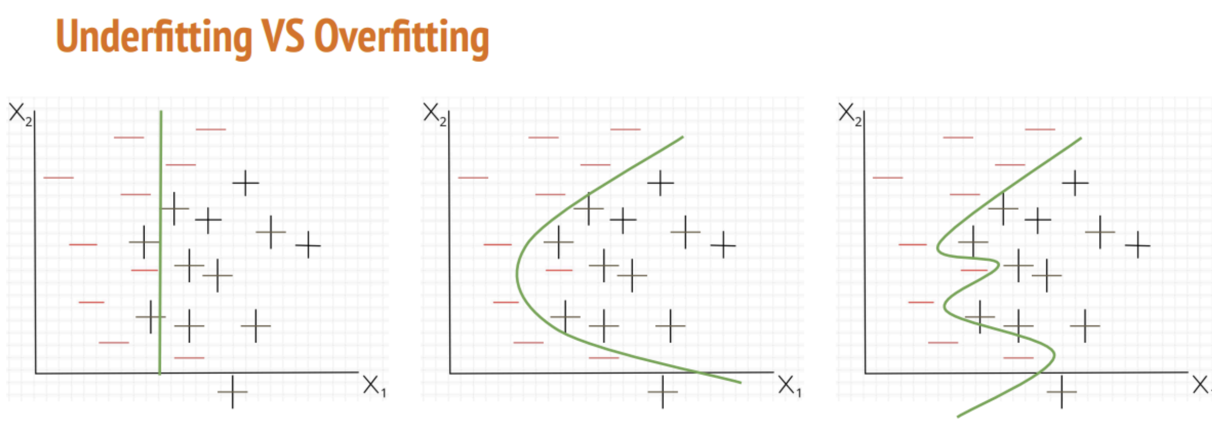
K = 1 → jagged boundary; reacts to every local point (very detailed, but noisy).
K = large → smoother boundary that may miss small structures.

This idea generalizes to any dataset — more neighbors = smoother predictions.

How Do We Know We've Done Well at Classification?
When we build a classifier, we don't just want it to memorize the data —
we want it to learn patterns that generalize to new data.

1. Test/train split
2. Watch for overfitting/underfitting

Type	Description	Effect
Overfitting	Model is too complex; memorizes training data	High training accuracy, poor test accuracy
Underfitting	Model too simple; misses real patterns	Low accuracy on both train & test

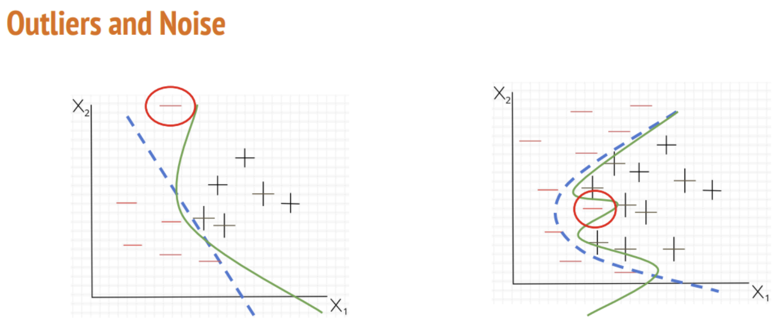


3. Evaluate with metrics
   - Accuracy
   - Precision
   - Recall
   - F1-score
   - (and, for probabilistic models, ROC/AUC)

4. Consider the Type of Mistakes
   - Sometimes what the model gets wrong matters more than how many.
   - Example: Predicting "no disease" for a sick patient is worse than the opposite. So evaluation should also match the cost of errors for your task.

5. Visual idea: Model complexity vs. error
   - When you plot error vs. model complexity:
     - Training error decreases steadily.
     - Testing error decreases at first, then rises again when overfitting begins.
   - You want to stop training around that minimum in test error.

6. Outliers and Noise



KNN Pros and Cons

Pros: Simple to understand why a given unseen record was given a particular class
Cons: Expensive to classify new points. KNN can be problematic in high dimensions (curse of dimensionality).
