Fondamenti di Analisi dei Dati

from data analysis to predictive techniques

Prof. Antonino Furnari (antonino.furnari@unict.it)
Corso di Studi in Informatica
Dip. di Matematica e Informatica
Università di Catania

Università di Catania

Introduction to Predictive Analysis

Predictive analysis represents a fundamental shift in how we approach data.

Rather than passively observing patterns, we actively build models to forecast outcomes and understand complex relationships.

This journey bridges traditional statistics with modern machine learning, equipping you with essential tools for both interpretation and prediction.

From Description to Prediction

Traditional Analysis

So far, we've focused on **understanding data** through:

- Summarizing with mean, median, variance
- Studying relationships via covariance and correlation
- Statistical inference to generalize findings

This approach is **passive**—we observe, compute scores, and perform tests.

Predictive Modeling

Now we shift to an **active approach** with two main goals:

- **Inference:** Understanding relationships between variables
- **Prediction:** Building models for unseen data

This field overlaps heavily with **machine learning** and requires creating formal predictive models.

Case Study: The Pima Indians Diabetes Dataset

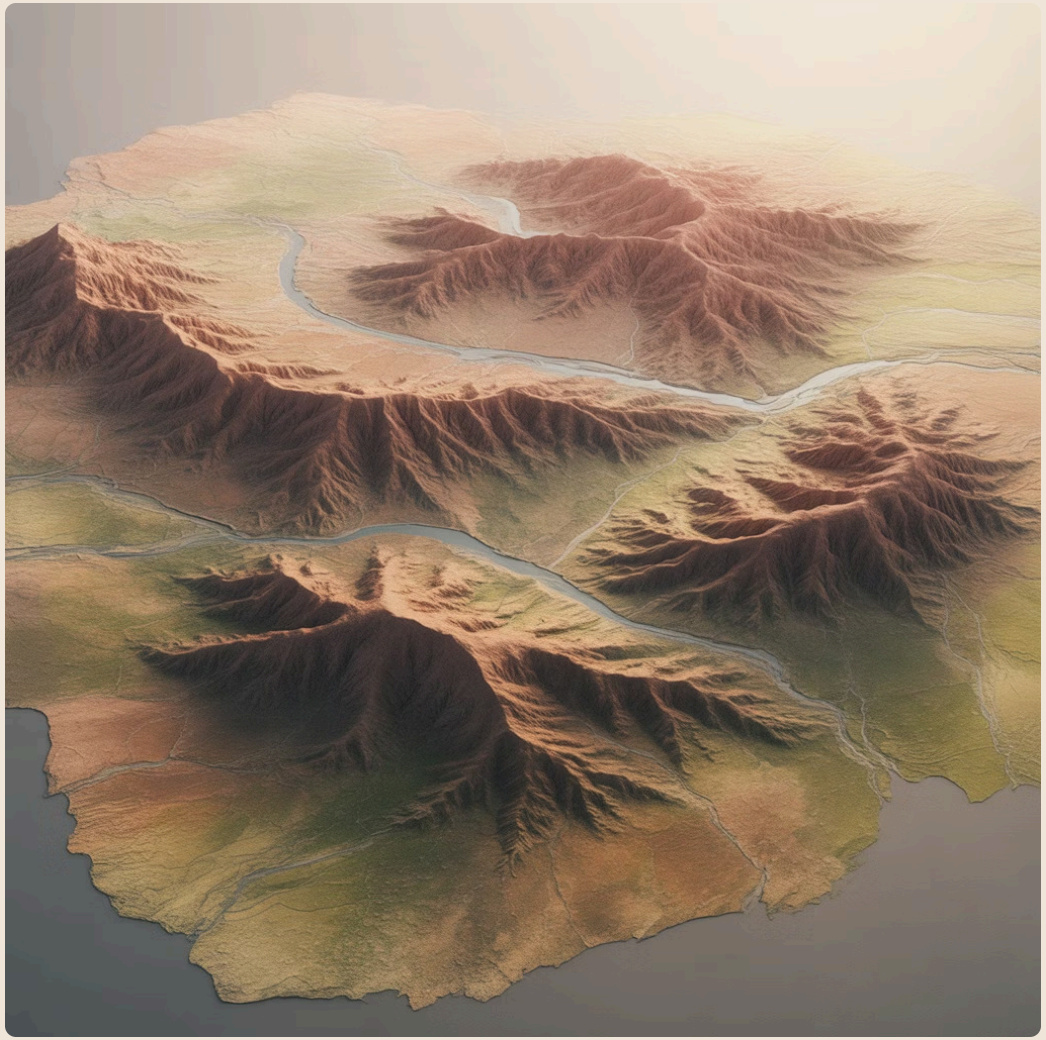
To ground our exploration, we'll examine the Pima Indians Diabetes Dataset. This dataset records clinical measurements from female patients of Pima Indian heritage, aged 21 and older, who underwent diabetes testing. It contains eight predictor variables and one outcome variable indicating diabetes status.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

The dataset records data about female patients of Pima Indian heritage, aged 21 and older, who were tested for diabetes. It includes the following variables:

- **Pregnancies**: Number of times the patient has been pregnant
- **Glucose**: Plasma glucose concentration after a 2-hour oral glucose tolerance test
- **BloodPressure**: Diastolic blood pressure (mm Hg)
- **SkinThickness**: Triceps skin fold thickness (mm)
- **Insulin**: 2-hour serum insulin (mu U/ml)
- **BMI**: Body mass index (weight in kg/(height in m)^2)
- **DiabetesPedigreeFunction**: A function that scores the likelihood of diabetes based on family history
- **Age**: Age of the patient (in years)
- **Outcome**: Binary indicator of diabetes status (0 = No diabetes, 1 = Diabetes)

What Is a Model?



In data analysis, a **model is a simplified representation of reality**. It's not reality itself, but a formal abstraction designed to serve a specific purpose.

Consider a geographical map: it's flat whilst the Earth is spherical, and it omits countless details like trees, rocks, and buildings. **Yet this simplification makes it useful for navigation.**

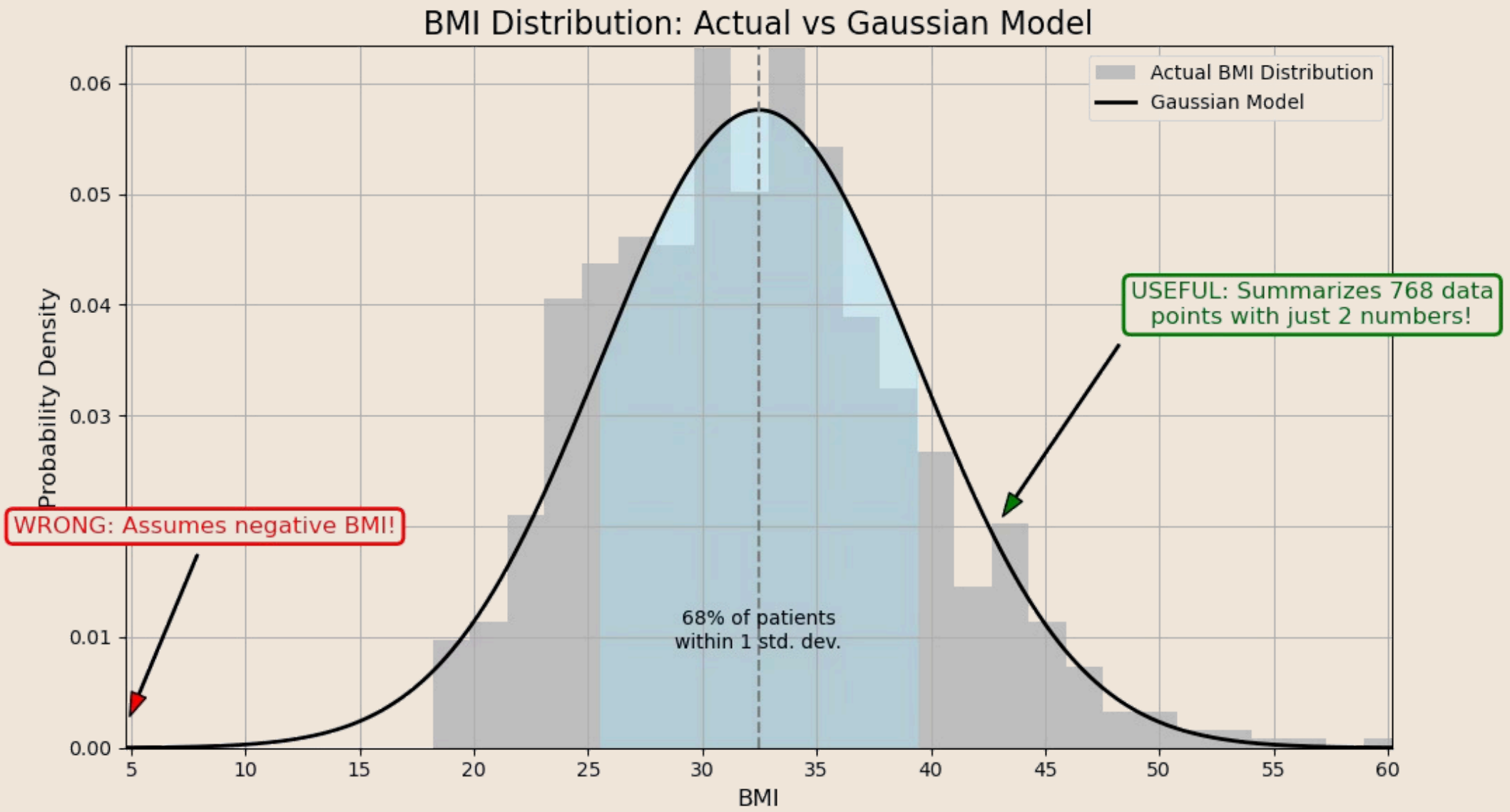
> *"All models are wrong, but some are useful."* — George Box

This famous aphorism captures the essence of modelling. A data model —whether statistical or machine learning—cannot capture every nuance of a complex real-world process.

But it becomes valuable if it helps us achieve our goal, whether understanding relationships or predicting outcomes.

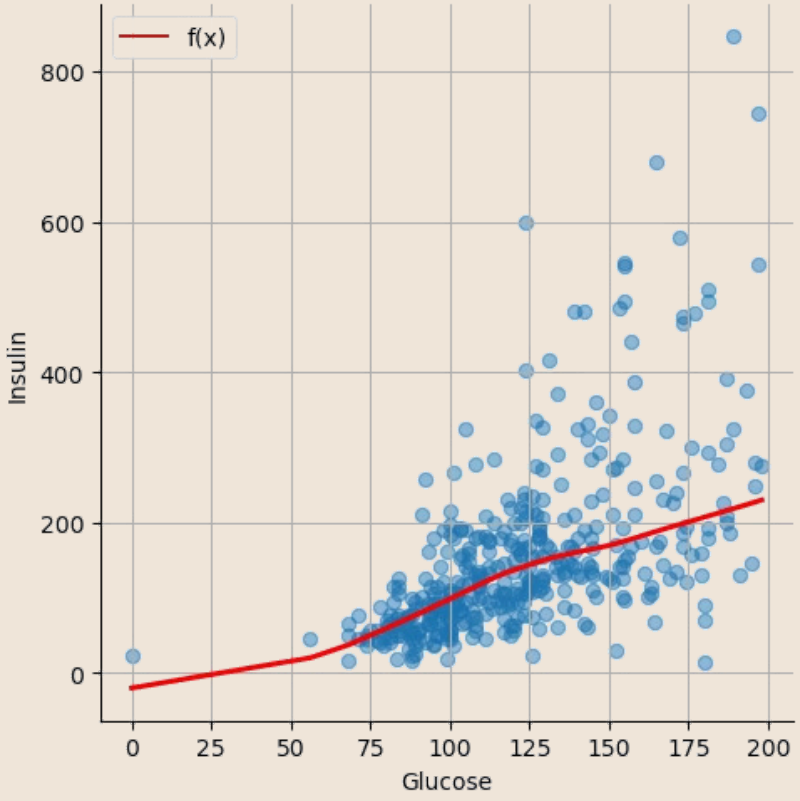
Example: Modelling BMI Distribution

Consider modelling the distribution of Body Mass Index (BMI) in the Pima dataset using a Gaussian (Normal) distribution. This distribution is defined by just two parameters: mean (μ) and standard deviation (σ). Whilst this model is technically "wrong"—it assumes negative BMI values are possible and extends to infinity—it proves remarkably useful.



Reality check from the data:

Mean BMI: 32.46
Standard Deviation: 6.92
Patients within 1 std dev: 513 out of 757
Percentage within 1 std dev: 67.77%



The General Predictive Model Framework

When performing predictive analysis, we work with a dependent variable Y (what we want to predict) and a set of predictor variables $X = (X_1, X_2, \ldots, X_n)$.

We express a predictive model in this general form:

$$Y = f(X) + \epsilon$$

Model Components

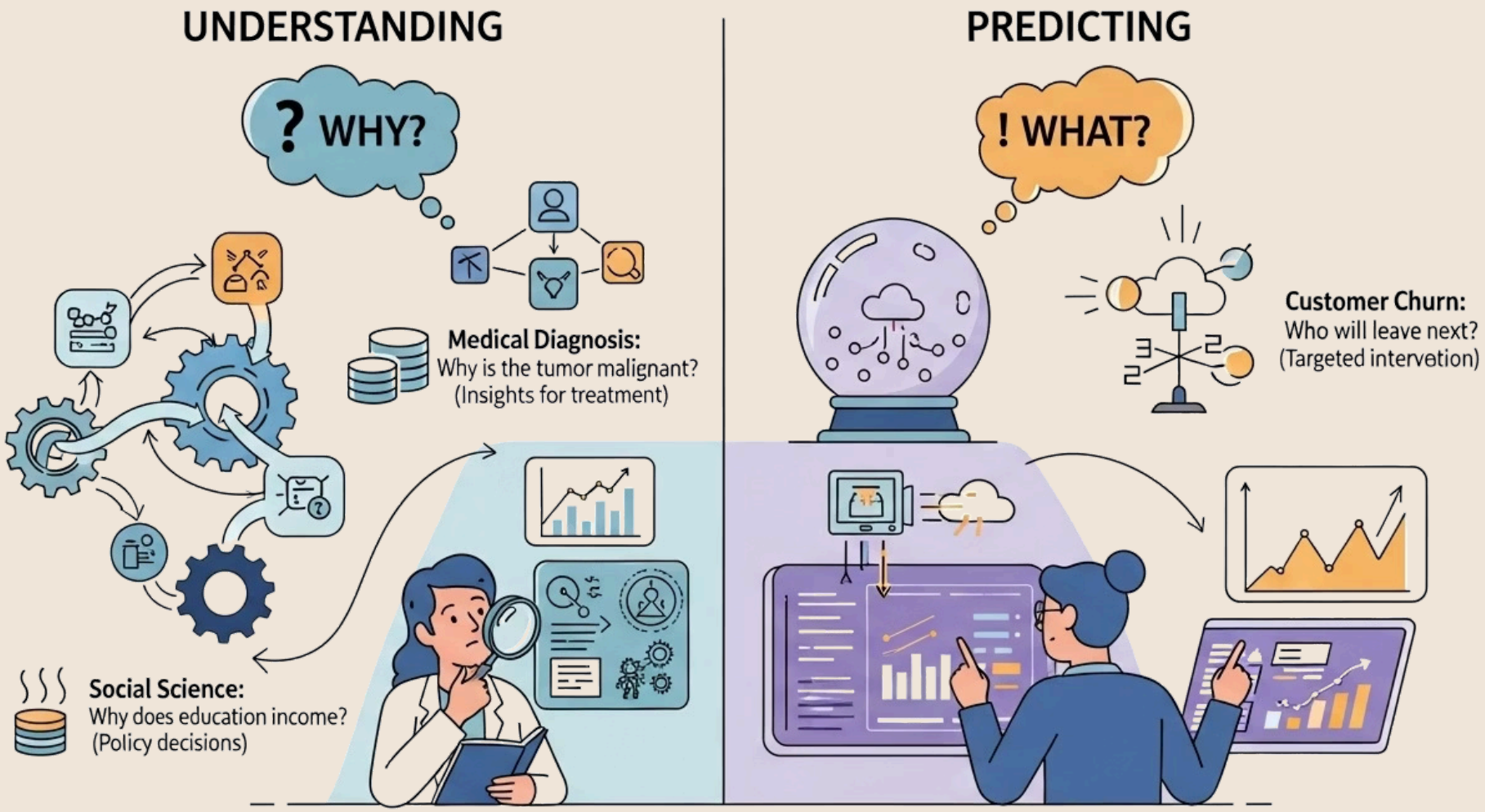
- **f(X)**: The predictive function we aim to discover through learning
- **ε (epsilon)**: The error term capturing prediction mistakes due to randomness or unmeasured factors

Diabetes Example

To **predict insulin levels from glucose concentration, we seek a function f that captures their relationship.** The red curve in visualisations represents our best estimate of this function, fitted to observed data points.

Two Goals: Understanding vs. Predicting

The usefulness of a model depends entirely on our primary objective. In data analysis, we typically pursue two main—sometimes competing—goals that shape every design choice we make.



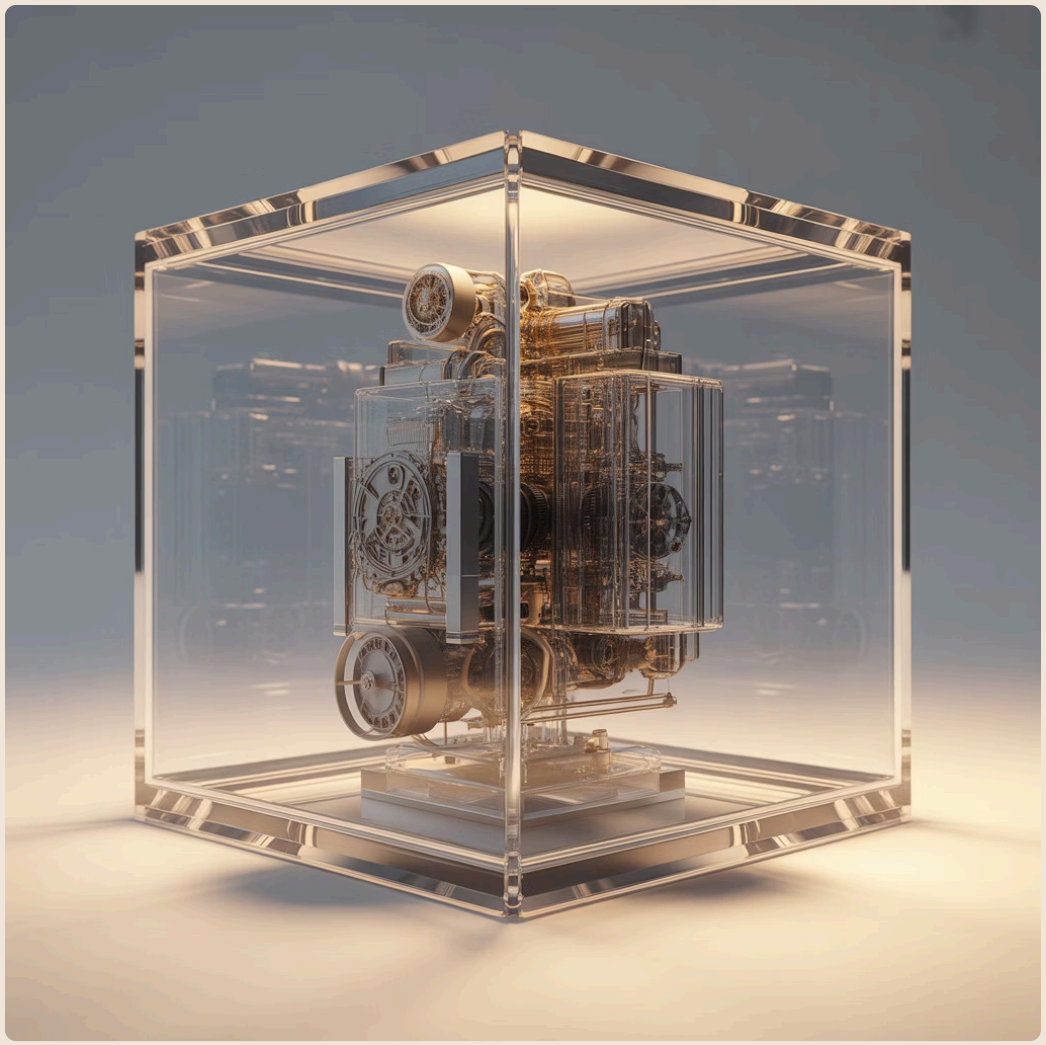
In some cases, the two approaches can overlap in an **integrated approach**.

> **Example**: in diabetes risk management we may seek both to **understand** what are the key drivers of diabetes risk and **predict** which patients are most likely to develop diabetes in the next year.

Statistics vs. Machine Learning

Statistical Approach

The "Glass Box"



Focus: Understanding and Inference

- Relies on assumptions (normality, independence)
- Uses hypothesis testing, p-values, confidence intervals
- Entire dataset used for fitting and testing
- **Interpretability is required**

Key Question: Is this relationship real or random chance?

Machine Learning Approach

The "Black Box"

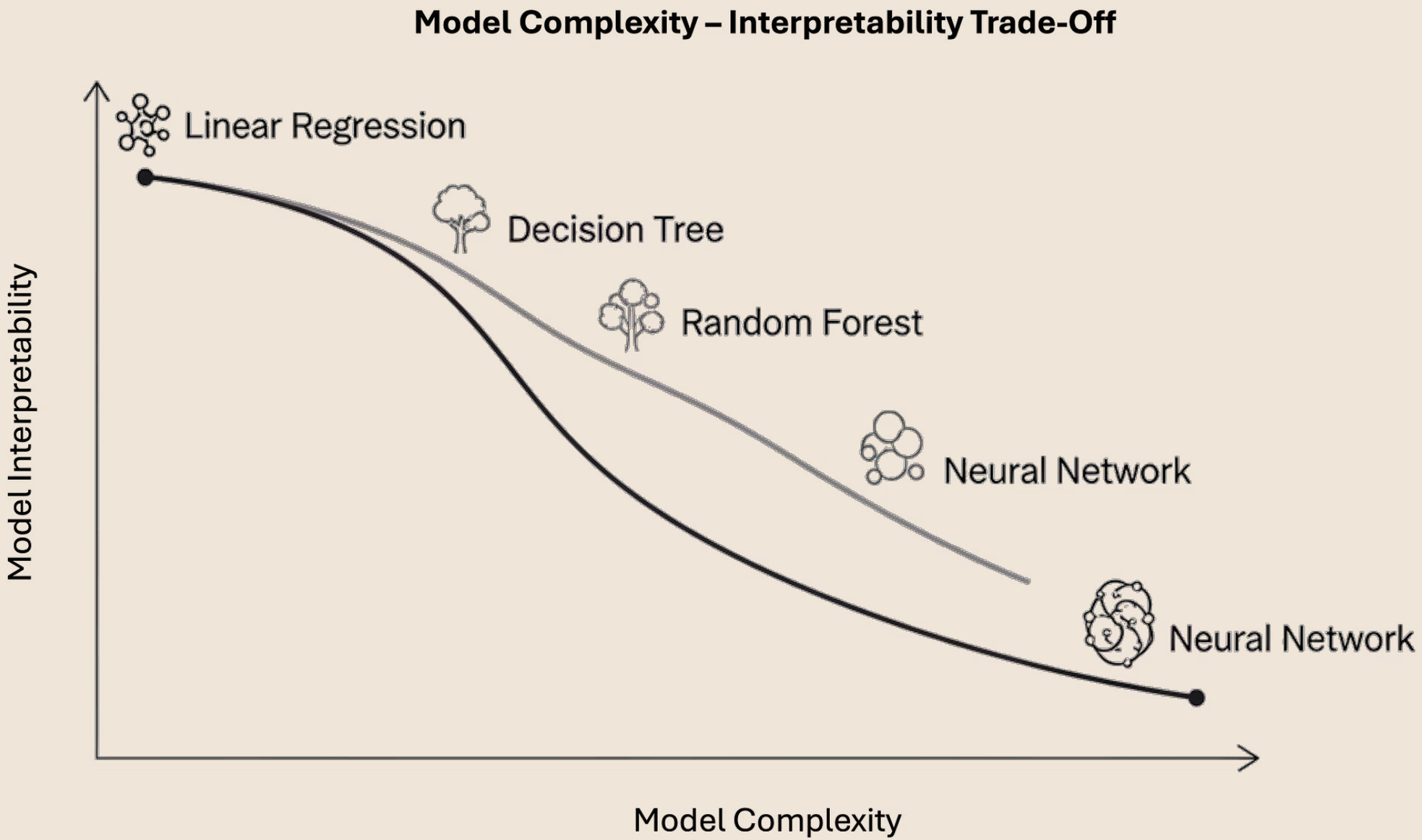
Focus: Prediction and Accuracy

- Results-driven with fewer assumptions
- Splits data into training/validation/test sets
- Success measured by test set performance
- **Interpretability is a bonus**

Key Question: How accurate are predictions on new data?

The Complexity-Interpretability Trade-off

As model complexity increases, interpretability typically decreases. This fundamental tension shapes every modelling decision we make.



Supervised vs. Unsupervised Learning

Supervised Learning

Learning with an Answer Key

Dataset contains both **input features (X)** and **correct answers (y)**. We "supervise" the model by showing examples with outcomes.

Goal: Learn mapping function h such that:

$$\hat{h} = h(X)$$

Divided into **Regression** (predicting numbers) and **Classification** (predicting categories).

Unsupervised Learning

Learning without an Answer Key

Dataset contains only **input features (X)** with no correct answers. Goal is finding **hidden structure or patterns**.

Goal: Discover natural groupings or relationships in data

Primary task is **Clustering**—automatically grouping similar data points together.

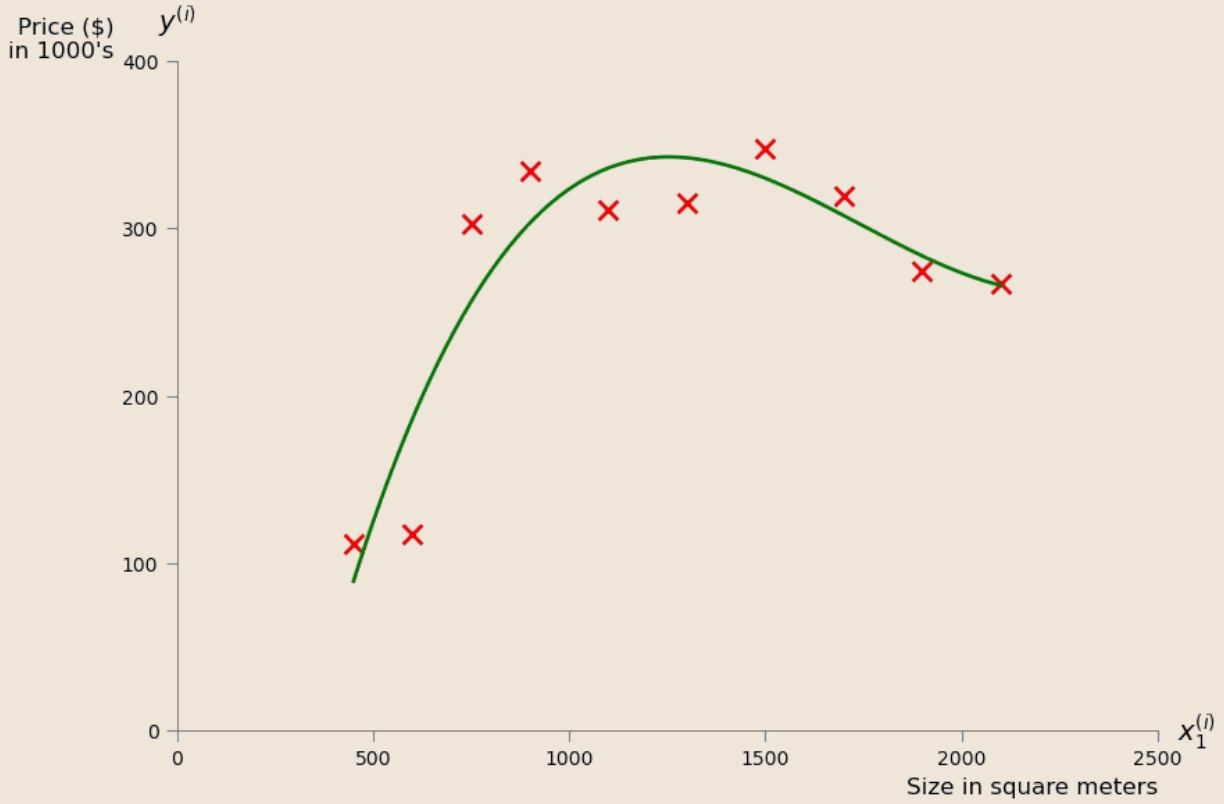
Regression: Predicting Numbers

In regression problems, the output variable *y* is a **continuous numerical value**. The goal is finding a function (like a line or curve) that best fits continuous data points.

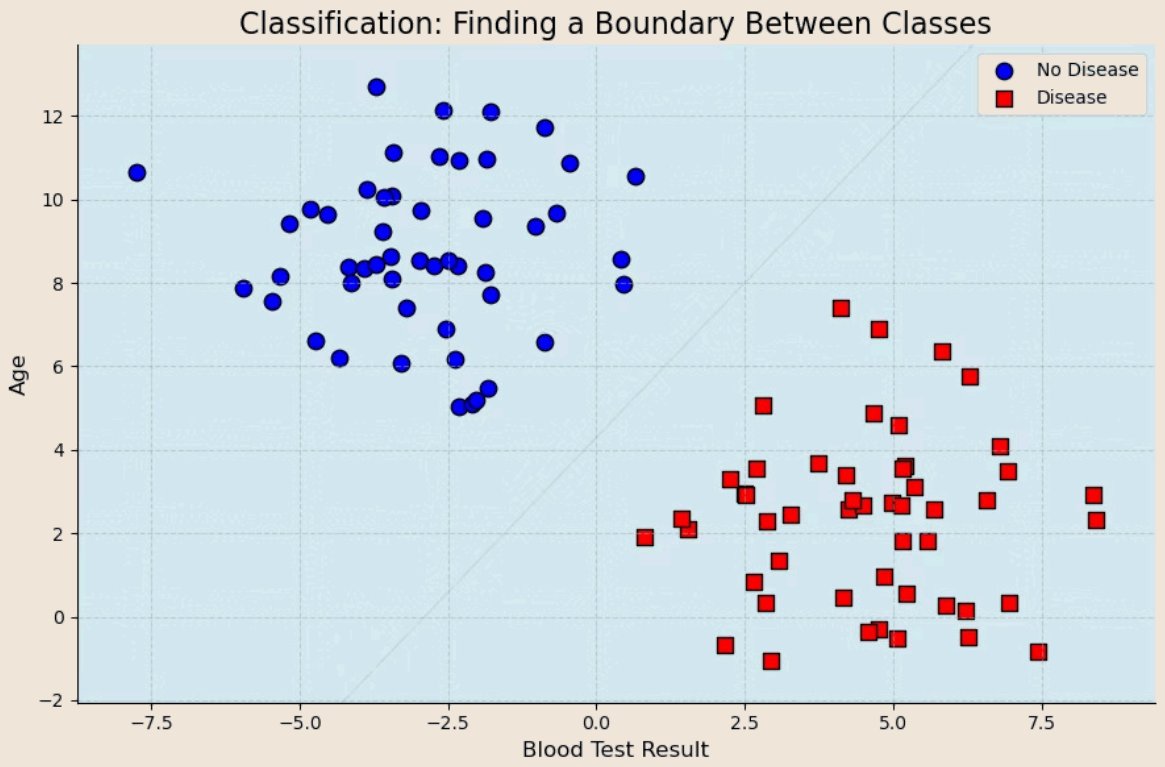
> **Key Question:** "How much?" or "How many?"

Examples include:

- Predicting house prices from square metreage
- Forecasting tomorrow's stock price from historical data
- Estimating a student's exam score from study hours and attendance
- Calculating expected insurance claims based on customer profiles



Classification: Predicting Categories



In classification problems, the output variable *y* is a **discrete category or label**. The goal is finding a boundary that best separates different labelled groups.

> **Key Question:** "Which one?" or "Is this A or B?"

Examples include:

- **Binary:** Spam detection, medical diagnosis, customer churn prediction
- **Multiclass:** Image recognition (cat, dog, or bird), sentiment analysis (positive, neutral, negative)

Unsupervised Learning: Finding Hidden Structure

In **unsupervised learning**, our dataset contains only input features (X)—no correct answers. The goal isn't predicting an outcome, but discovering hidden structure, patterns, or groupings within the data itself.

Clustering

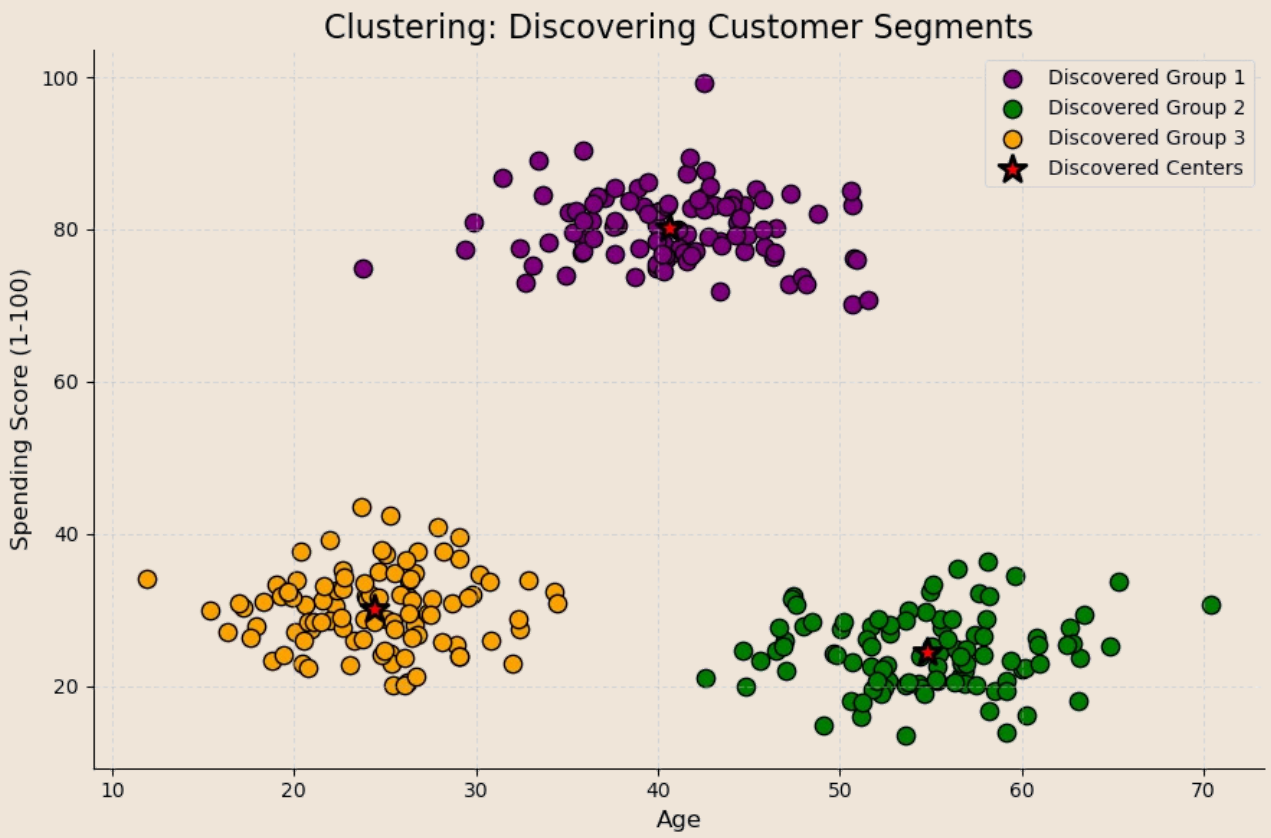
Goal: Automatically group similar data points into clusters based on their features. **Example:** Customer segmentation discovering "Young Savers", "Prime Spenders", and "Older Savers" from age and spending data, enabling targeted marketing.

Dimensionality Reduction

Goal: Reduce the number of variables whilst preserving essential information. **Example:** Compressing high-dimensional gene expression data to two dimensions for visualisation whilst retaining biological patterns.

Anomaly Detection

Goal: Identify unusual patterns that don't conform to expected behaviour. **Example:** Detecting fraudulent transactions by finding patterns that deviate significantly from normal customer behaviour.



Clustering Example

Segmenting customers of a bank into three groups based on Age and Spending:

- Young Savers (Group 3)
- Prime Spenders (Group 1)
- Old Savers (Group 2)

Differently from supervised learning, nobody is giving us the label and we have to find them ourselves.

Once this segmentation has been performed, we can design targeted marketing campaigns.

Parametric vs. Non-Parametric Models

Parametric Models

The "Fixed" Approach

Makes strong assumptions about function form with a **fixed number of parameters** regardless of dataset size.

Examples: Linear Regression, Logistic Regression, Naive Bayes

✓ Simpler and faster to train
✓ Requires less data
✓ More interpretable

 Constrained by assumptions
 Risk of underfitting

Non-Parametric Models

The "Flexible" Approach

Makes few assumptions with **parameters that grow** as training data increases. Lets data "speak for itself."

Examples: K-Nearest Neighbors

✓ Highly flexible
✓ Captures complex patterns
✓ Less bias

 Requires much more data
 Slower to train
 Risk of overfitting

This choice embodies the **Bias-Variance Tradeoff**: parametric models are high-bias/low-variance, while non-parametric models are low-bias/high-variance.

The Learning Principle: Empirical Risk Minimisation

How do we find the "best" model? We need a formal framework.

Let $\mathcal{X}$ and $\mathcal{Y}$ be two spaces of objects. We can see them as the sample spaces of two random variables X and Y , with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ being two realizations of the random variables: $X = x$ and $Y = y$.

The goal of predictive modelling is to find a function (or **hypothesis**) h:

$$h : \mathcal{X} \to \mathcal{Y}$$

which outputs an object $\hat{y} \in \mathcal{Y}$, given an object $x \in \mathcal{X}$. We use the symbol y to refer to the **true value** (or **ground truth**) associated with x, while (empty) **\hat y will represent the prediction** obtained using our function h :

$$\hat{y} = h(x)$$

> **Example 1: Medical Diagnosis**
> We can imagine $\mathcal{X} = \Re^m$ and $\mathcal{Y} = \{0, 1\}$, with $\mathbf{x} \in \Re^m$ being a numerical vector representing the results of different blood tests, while $y \in \{0, 1\}$ is a response variable indicating whether the subject has (1) or does not have (0) a given disease. Finding an appropriate h will allow us to predict $\hat{y} = h(\mathbf{x})$.

> **Example 2: Simple Spam Filter**
> We want to build a **spam filter** where $\mathcal{X}$ is the set of all possible emails and $\mathcal{Y} = \{0, 1\}$. Let $f(x)$ be a function that counts the number of orthographical mistakes in an email x. We can define our hypothesis h as:

$$h(x) = \begin{cases} 1 & \text{if } f(x) > \theta \\ 0 & \text{otherwise} \end{cases}$$

Parameters: What the Model Learns

The function h is typically a **parametric model**, meaning its form is defined by a fixed set of **parameters**. The 'learning' process involves finding the optimal values for these parameters.

For instance, the threshold θ in our spam filter example is a parameter—a 'knob' the algorithm tunes. Each θ value creates a distinct hypothesis function h.

More complex models, like a linear regressor, have multiple parameters such as coefficients $\beta_0, \beta_1, \ldots, \beta_n$. The entire set of possible functions h achievable by adjusting these parameters is called the **class of functions (or hypothesis space)** $\mathcal{H}$.

The core question remains: how do we find the **best** parameters?

The Learning Process: Statistical Learning and Empirical Risk

To find the "best" h from our hypothesis space $\mathcal{H}$, we need a way to measure how "good" it is. We do this by defining a **loss (or cost) function**, which measures the penalty for making a prediction $\hat{y}$ when the true value is y:

$$L(\hat{y}, y)$$

Ideally, we want to find the function h^* that has the lowest possible loss *on average* over *all possible data*, not just the data we have. This "true" average loss is called the **Risk** $R(h)$, defined as the **expected loss** under the true (but unknown) joint probability distribution $P(X,Y)$:

$$R(h) = E_{(x,y)\sim P(X,Y)}[L(h(x), y)]$$

However, we cannot compute true Risk because we don't know the true distribution P(X,Y). Instead, we estimate it using our training set of N examples: $\mathrm{TR} = \{(x_i, y_i)\}_{i=1}^{N}$. This leads to **Empirical Risk**:

$$R_{emp}(h) = \frac{1}{N}\sum_{i=1}^{N} L(h(x_i), y_i)$$

Once we have this expression, we find our optimal h^* by solving this optimisation problem::

$$\hat{h} = \arg\min_{h\in\mathcal{H}} R_{emp}(h)$$

01

Define Loss Function

Choose how to penalise prediction errors (e.g., squared error for regression)

02

Calculate Empirical Risk

Average the loss over all training examples

03

Minimise Risk

Find parameters that minimise empirical risk using optimisation algorithms

Model Capacity: Flexibility vs. Complexity

Before moving on, we need to define the crucial concept of model capacity. In simple terms, a model's capacity refers to its **flexibility** or **complexity**, directly related to the **size and richness of its hypothesis space** $\mathcal{H}$.

Low Capacity Model: Simple & Constrained

Has a **small, constrained** hypothesis space $\mathcal{H}$, representing a limited set of simple functions.

Example: A simple linear regression model ($h(x) = \beta_0 + \beta_1 x$) can *only* learn linear patterns. It is "biased" towards simplicity.

Risk: May be too simple to capture the true underlying data patterns, leading to **underfitting**.

High Capacity Model: Flexible & Complex

Possesses a **large, rich** hypothesis space $\mathcal{H}$, capable of representing complex functions.

Example: A 10th-degree polynomial can fit highly "wiggly" functions, perfectly matching almost any training points.

Risk: So flexible it can fit random *noise* in training data, not just the signal. Leads to **overfitting** and poor generalisation.

Assessing Model Accuracy

Once we use Empirical Risk Minimisation (ERM) to "learn" a model $\hat{h}$, we must evaluate its effectiveness. A model that perfectly fits our training data (i.e., has a low R_{emp}) is not necessarily a good model. Our ultimate goal is a model that **generalises** well—that is, one that makes accurate predictions on new, unseen data.

To do this, we must first define *how* to measure performance.

Measuring the Quality of Fit

To measure the "quality of fit" or performance of a model, we need a concrete **evaluation metric**. This metric is sometimes the same as the **loss function** $L(\hat{y}, y)$ we used in the ERM principle. It quantifies the cost or error of our predictions. The specific metric we choose depends on the problem (e.g., regression vs. classification).

For **regression problems**, the most common metric is the **Mean Squared Error (MSE)**.

If we choose our loss function $L(y, \hat{y}) = (y - \hat{y})^2$, then the **Mean Squared Error (MSE)** is precisely the **Empirical Risk** we are trying to minimise:

$$R_{emp}(h) = \frac{1}{N} \sum_{i=1}^{N} (y_i - h(x_i))^2 = \text{MSE}$$

This leads to a critical question: is a model with a very low MSE on our *training data* a good model? Not necessarily.

Overfitting and Underfitting

The central challenge of predictive modelling: a model that perfectly fits training data isn't necessarily good. Models can be too simple (underfitting) or too complex (overfitting). Finding the "just right" complexity is crucial.

For example, this model:

$$\hat{h}(x) = y \text{ s.t. } (x, y) \in \text{TR}$$

Leads to an empirical risk equal to zero, but the function is not even defined for values outside the training set! This is an extreme case of overfitting where the model is just "memorising the data".

Underfitting

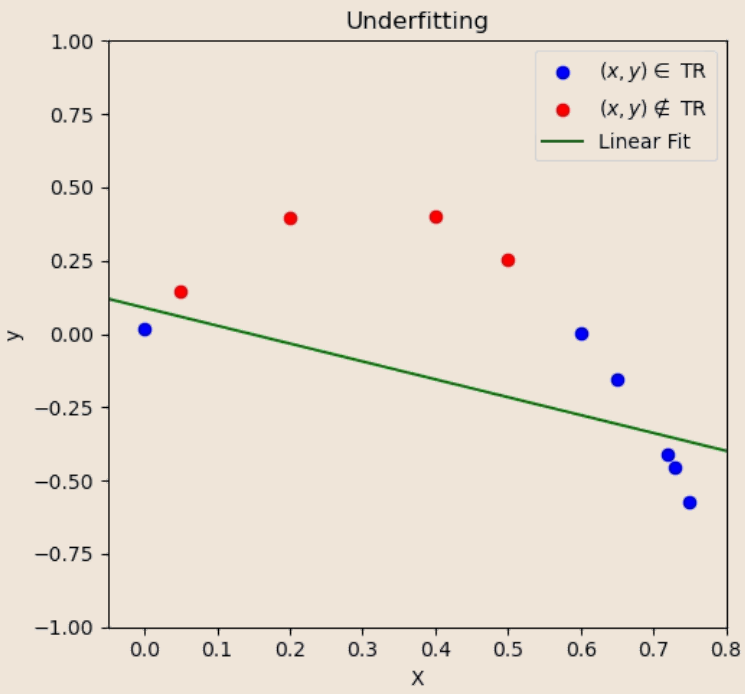
Model is too simple to capture underlying patterns. High training error, high test error. Example: fitting a straight line to curved data. Solution: increase model capacity.

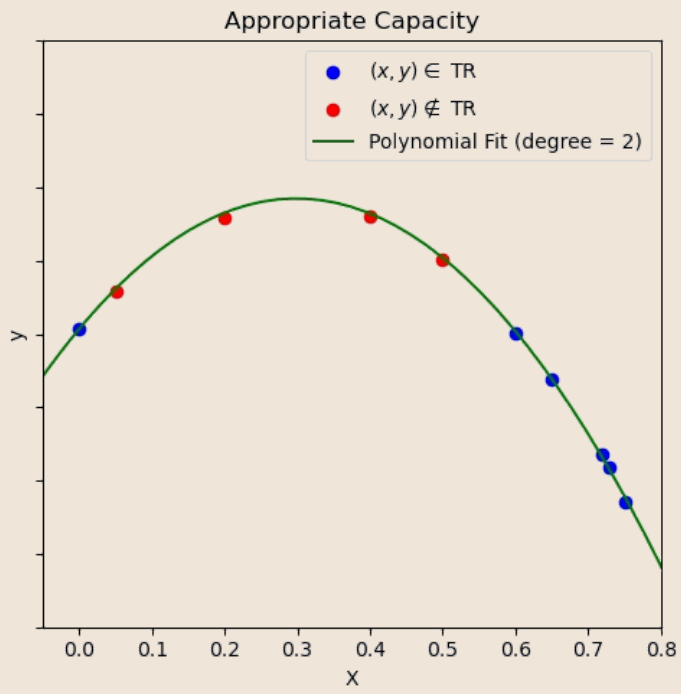
Good Fit

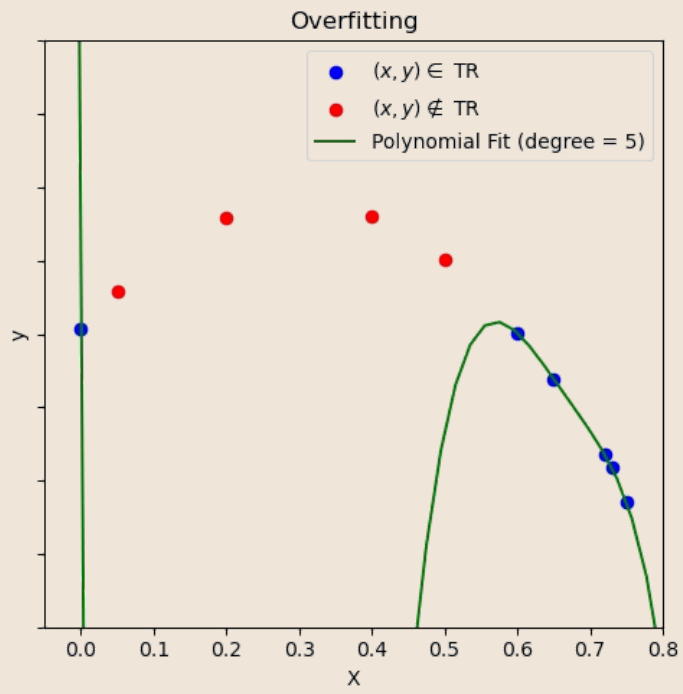
Model captures signal without memorising noise. Moderate training error, low test error. Generalises well to unseen data. Optimal bias-variance trade-off.

Overfitting

Model memorises training data including noise. Very low training error, high test error. Example: high-degree polynomial fitting random fluctuations. Solution: reduce model capacity or add regularisation.







The Bias-Variance Trade-off

The expected test error at a given point x_0 decomposes into three fundamental quantities that explain model performance:

$$E[(y_0 - \hat{f}(x_0))^2] = [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\hat{f}(x_0)) + \text{Var}(\epsilon)$$

Squared Bias

Error from underfitting. High-bias models are "stubborn"—stuck in simple assumptions. Low-capacity models have high bias. They miss the true pattern, producing high error on both training and test sets.

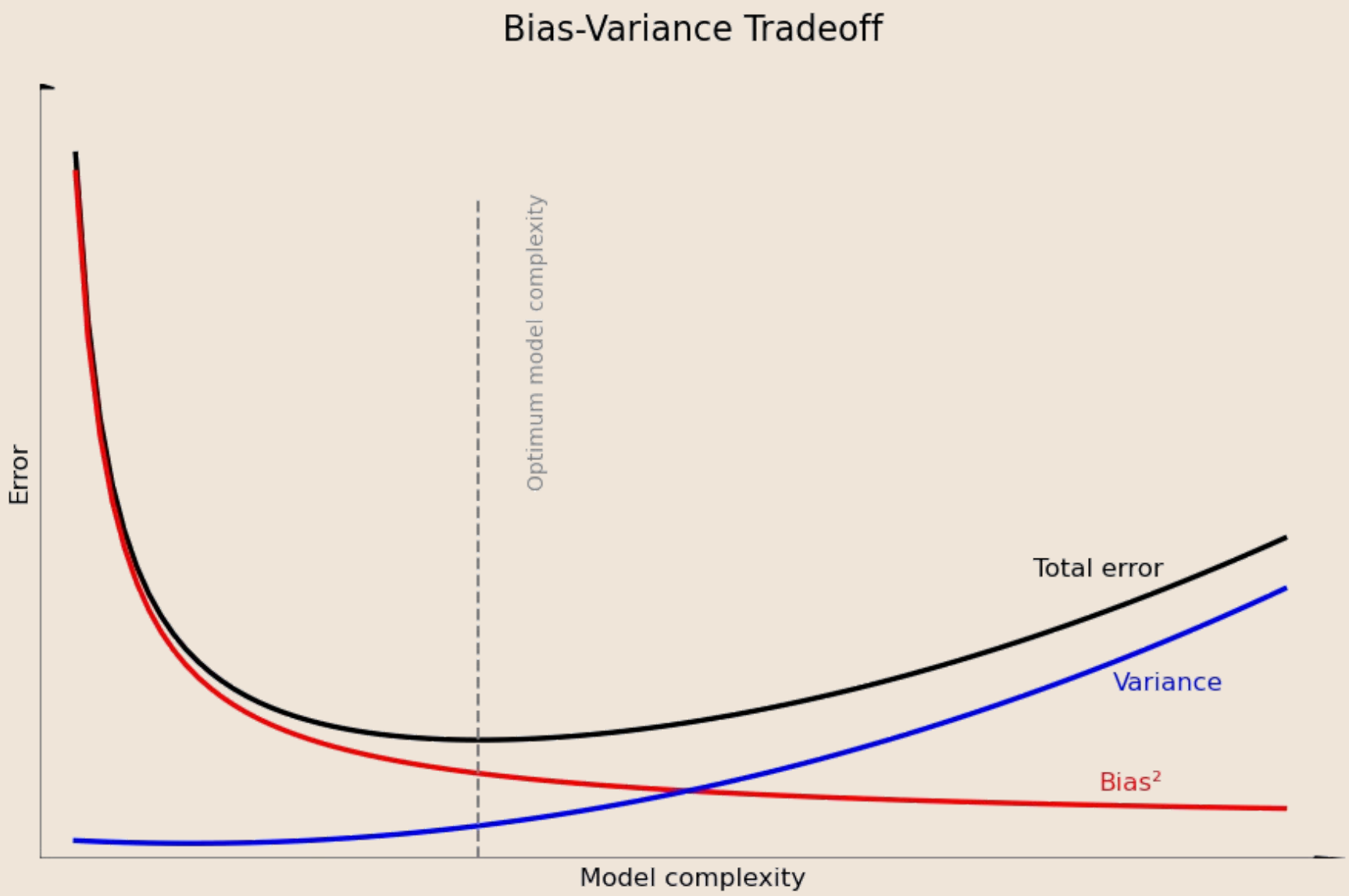
Variance

Error from overfitting. High-variance models are "nervous"—too sensitive to training data specifics. High-capacity models have high variance. They fit training noise perfectly but fail on test data.

Irreducible Error

Inherent noise in real-world data that no model can eliminate. Represents the upper limit on achievable accuracy. Exists due to unmeasured variables and random fluctuations.

As model capacity increases, bias decreases (good) but variance increases (bad). Our goal: find the optimal capacity that minimises total error.



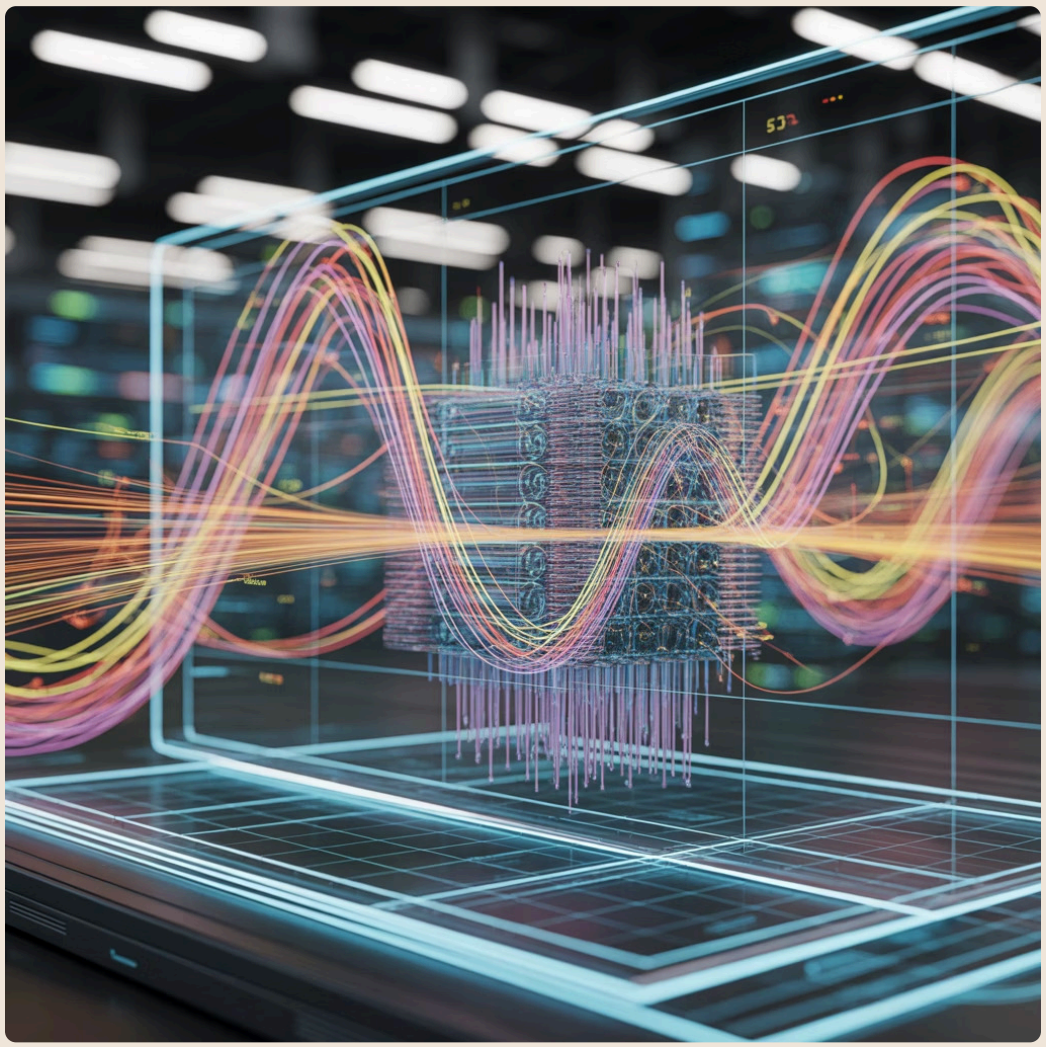
Parameters vs. Hyperparameters

Model Parameters

Values the model **learns from data** during training through Empirical Risk Minimization. Internal to the model, direct output of learning algorithm.

Examples:

- Coefficients β_0, β_1, ... in linear regression
- Weights in neural networks
- Probabilities in Naive Bayes



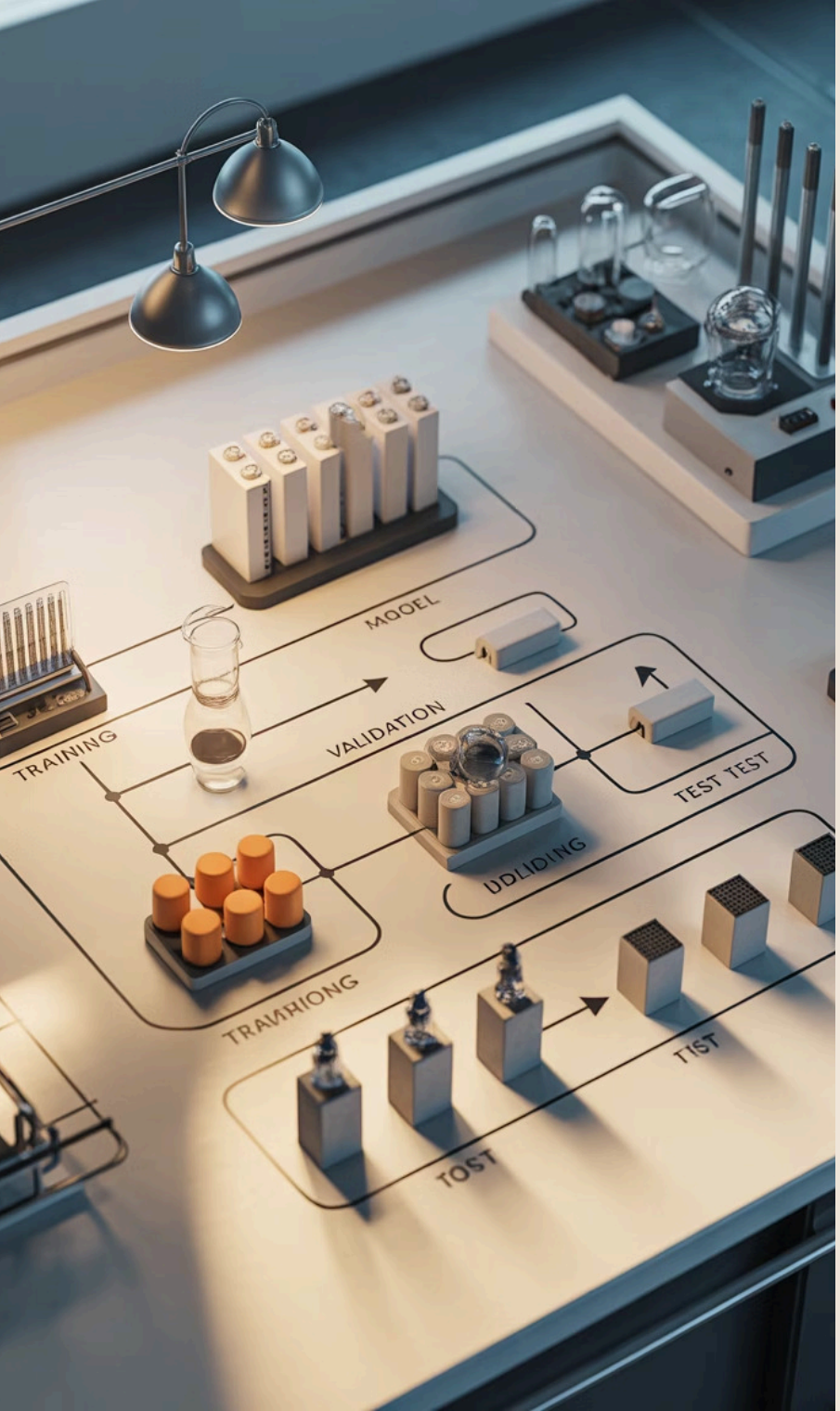
Hyperparameters

Configuration settings set **before training begins**. External to model, control capacity and learning process—the "knobs" we tune.

Examples:

- Polynomial degree k (controls capacity)
- Regularization parameter λ
- Number of clusters k in K-Means

Critical Challenge: Learning algorithms find best model parameters for *given* hyperparameters, but cannot choose optimal hyperparameters. That's our job—how we find the bias-variance sweet spot.



Model Selection Strategies

Statistical Selection (Understanding)

For **inference goals**, select models based on explaining observed data, favoring simplicity.

Uses entire dataset with statistical measures:

- p-values for variable significance
- Adjusted R^2 balancing fit and complexity

Not trying to predict future, but find trustworthy explanation.

Predictive Selection (Predicting)

For **prediction goals**, only criterion is performance on unseen data.

Splits data to simulate future:

- Training set: optimize model
- Validation set: choose hyperparameters
- Test set: final performance evaluation

Common ratio: 60% train, 20% validation, 20% test.

Golden Rule: Performance on training data is irrelevant. Only performance on unseen data matters.

Performance Measures

When our main goal is **prediction**, our selection criteria change entirely. We no longer care about statistical significance; we focus solely on performance on unseen data.

> **The Golden Rule of Predictive Modelling:** The performance of a model on the data it was trained on is *irrelevant*. The *only* measure that matters is its performance on new, unseen data.

To adhere to this rule, we simulate future performance by splitting our existing data into distinct sets for training, validation, and testing. This empirical approach is essential, especially for complex models where traditional statistical tests are difficult to apply.

Loss Function (for Training)

Used internally by the learning algorithm to guide model optimisation (e.g., during Empirical Risk Minimisation). Often chosen for its mathematical properties that facilitate efficient learning, even if it's an approximation of our true objective.

Performance Measure (for Evaluation)

Our true objective function to quantify a model's quality. It assesses how well the model performs on unseen data. These measures are typically 'the higher the better' (e.g., accuracy) or 'the lower the better' (e.g., error rate).

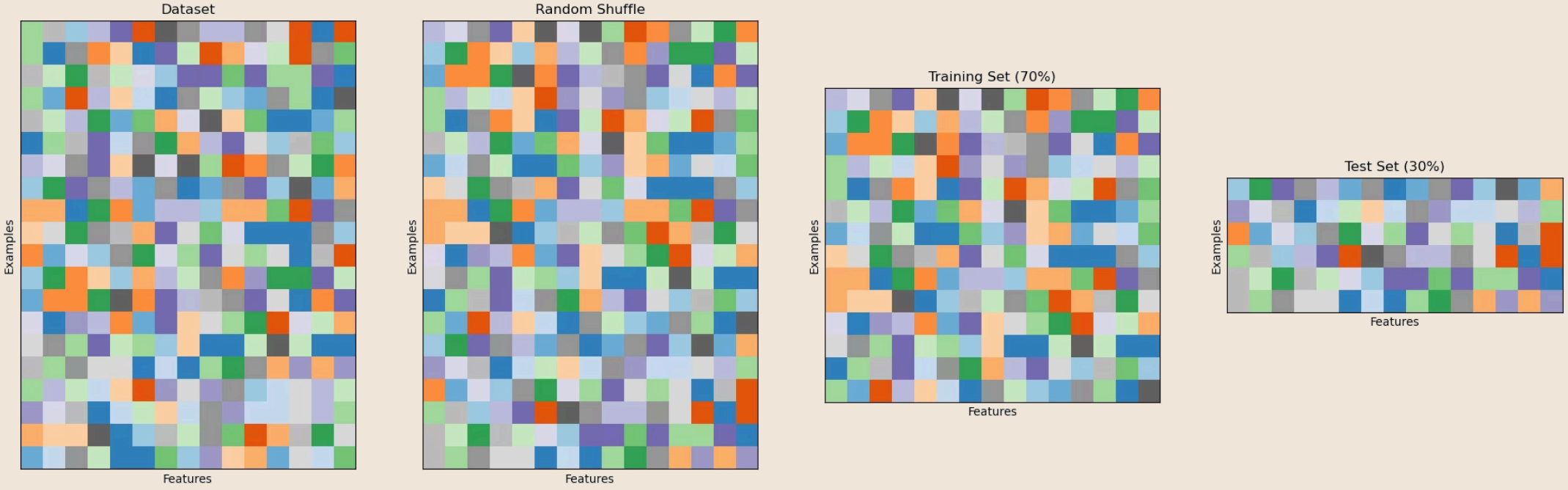
A performance measure p evaluates the set of ground truth values Y against the set of corresponding predictions $\hat{Y}$ for a given dataset:

$$p : \mathcal{Y}^N \times \mathcal{Y}^N \to \Re$$

While empirical risk can serve as a performance measure, we often use other, more intuitive metrics tailored to the specific problem.

Holdout Validation or Single Split

Split the data into two parts: training and test set following some ratio, e.g., using $70 - 80\%$ of the data for training and the rest for testing.



K-Fold Cross-Validation

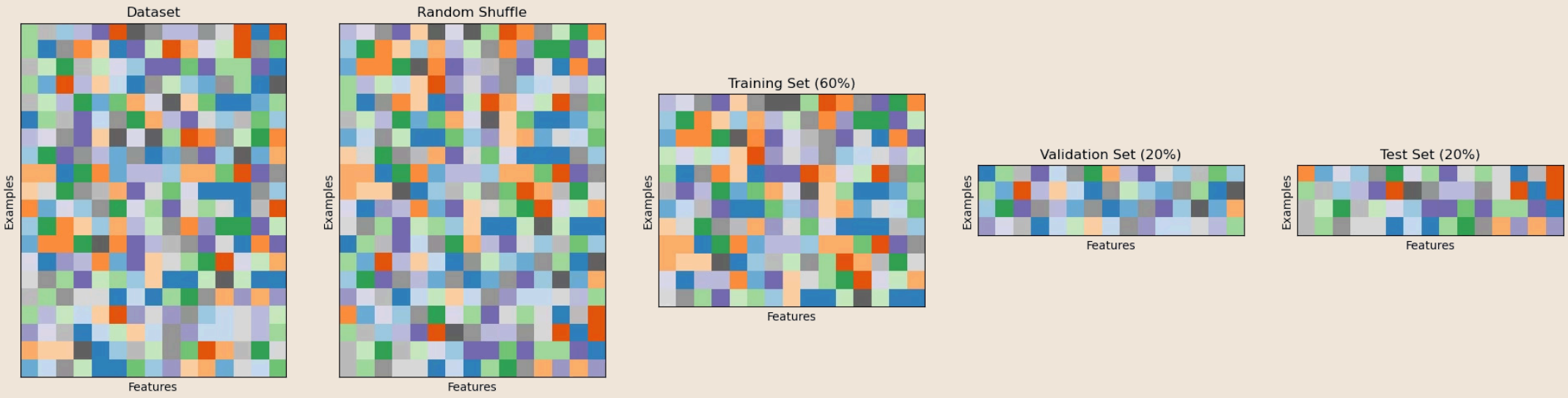
If the dataset is small, divide into K folds, then use $K-1$ folds for training and the rest for validation. Aggregate results across folds (e.g., with mean).

Leave-One-Out Cross-Validation

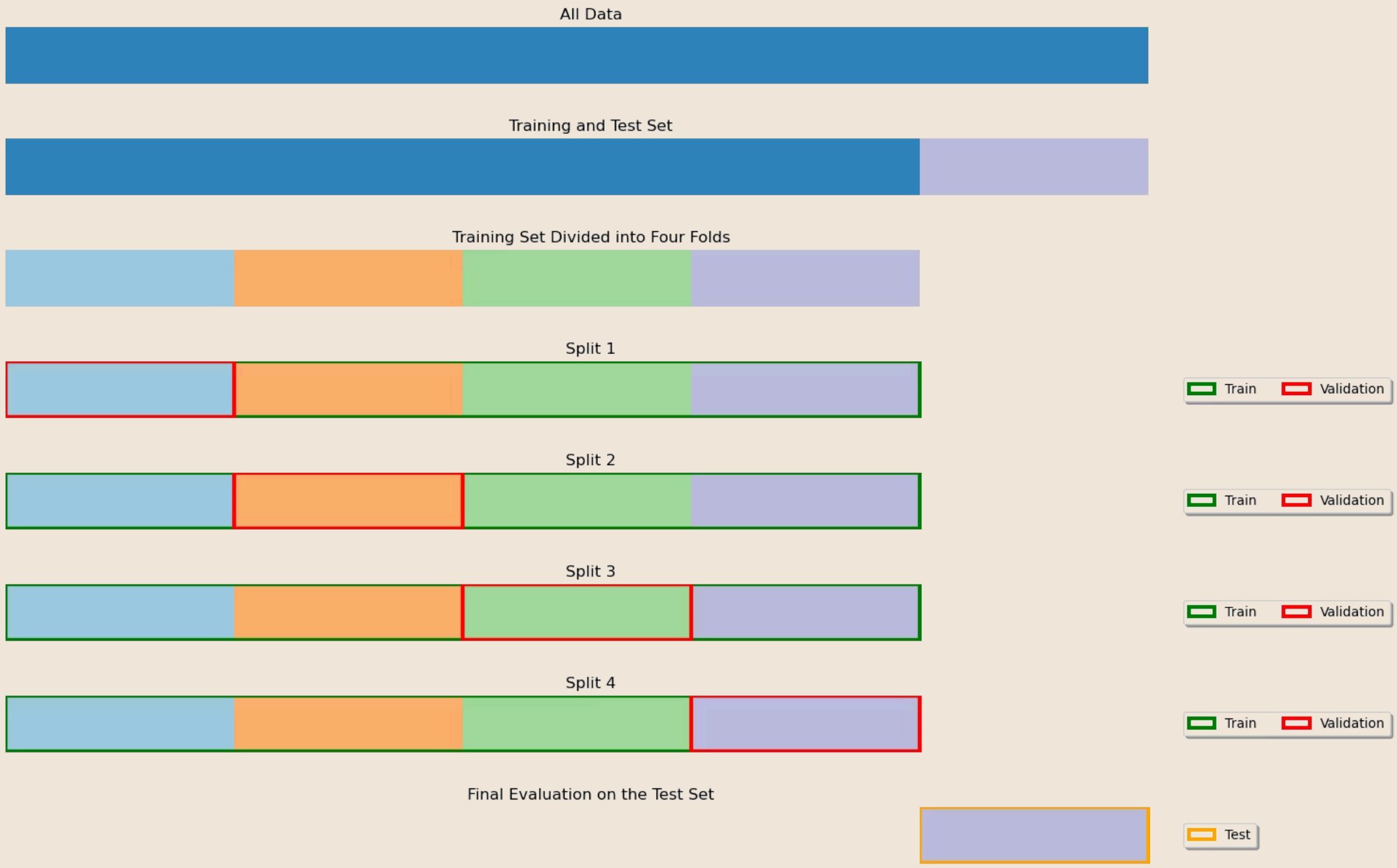
When the dataset is very small, we come to an extreme of K-Fold validation where each fold contains a single element.

Model Selection and Hyperparameter Optimisation

When we need to tune hyper-parameters, we need to have three sets: a training set (for trianing), a validation set (to choose hyperparameters), and a test set (to test final performance). Often we split in a $60:20:20$ ratio, but other ratios are also common.



Other combinations are also possible. For instance holdout validation + cross-validation only for hyperparameters:



Conclusions and Next Steps

We Have Explored:

- The fundamentals of predictive analysis
- Differences between prediction and inference
- Statistics vs machine learning
- Different machine learning problems
- Parametric vs non-parametric problems
- Empirical risk minimization
- Model accuracy and model selection

In the next lectures, we will look at the linear regression.

References

- Chapter 2 of James, Gareth Gareth Michael. An introduction to statistical learning: with applications in Python, 2023. https://www.statlearning.com
- https://en.wikipedia.org/wiki/Empirical_risk_minimization
- https://scikit-learn.org/stable/modules/cross_validation.html