# Nanostructured Materials

## Machine Learning Basics

Attila Cangi

Center for Advanced Systems Understanding,
Helmholtz-Zentrum Dresden-Rossendorf

a.cangi@hzdr.de

June 17, 2025

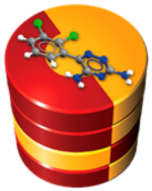
# Lecture Schedule

| Teaching team (Shirong/Ebru/Chenchen/Attila) | | | | |
|---|---|---|---|---|
| **Lecture No.** | **Module** | **Lecture Topic** | **Date (Tue.)** | **Lecturer** |
| 1 | M1: Introduction | General Introduction | 08.04 | Dr. Shirong Huang/Dr. Ebru Cihan |
| 2 | M2: Fabrication and characterization | Nanofabrication: Top-down/Bottom-up approaches | 15.04 | Dr. Chenchen Wang |
| 3 | M2: Fabrication and characterization | Imaging/characterization techniques: Electron microscopy based techniques | 22.04 | Dr. Ebru Cihan |
| 4 | M2: Fabrication and characterization | Scanning probe microscopy (SPM) 1: Scanning tunneling microscopy (STM) | 29.04 | Dr. Ebru Cihan |
| 5 | M2: Fabrication and characterization | Scanning probe microscopy (SPM) 2: Atomic force microscopy (AFM) | 06.05 | Dr. Ebru Cihan |
| 6 | M3: Low-d materials and application | Carbon Nanostructures | 13.05 | Dr. Shirong Huang |
| 7 | M3:Low-d materials and application | Introduction to low-dimensional materials (LDMs) | 20.05 | Dr. Shirong Huang |
| 8 | M3:Low-d materials and application | Invited talk: Nanostructured materials-based Biosensors | 27.05 | Dr. Shirong Huang, Prof. Anthony Guiseppi-Elie |
| 9 | M3:Low-d materials and application | Nanostructured materials-based Gas sensors | 03.06 | Dr. Shirong Huang |
| 10 | M3:Low-d materials and application | Machine Learning Basics | 17.06 | Dr. Attila Cangi |
| 11 | M3:Low-d materials and application | Artificial Neural Networks | 24.06 | Dr. Attila Cangi |
| 12 | M3:Low-d materials and application | Biomimetic electronic noses (e-noses): AI-powered Sensors | 01.07 | Dr. Shirong Huang |
| 13 | M3:Low-d materials and application | Machine learning for molecule sensing | 08.07 | Dr. Shirong Huang, Prof. Cuniberti |
| 14 | / | Exam day | 15.07 | Dr. Shirong Huang |

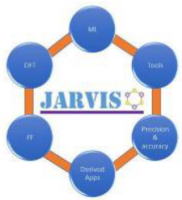
- **Exercise class**: Tomorrow, June 18, 13:00 - 14:30 (MOL/213/H)
- **Q&A session** on Scientific Project "AI-Driven Prediction of Material Properties"

# The Data Flood in Materials Science

- High-throughput calculations (first-principles, molecular dynamics, etc.), combinatorial experiments, and open databases have increased significantly
- Manual analysis or brute-force simulation cannot keep pace
- Hidden patterns = discovery opportunities → need **Machine Learning**


The Cambridge Crystallographic Data Centre

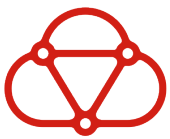
OQMD
The Open Quantum Materials Database




NOMAD
NOVEL MATERIALS DISCOVERY

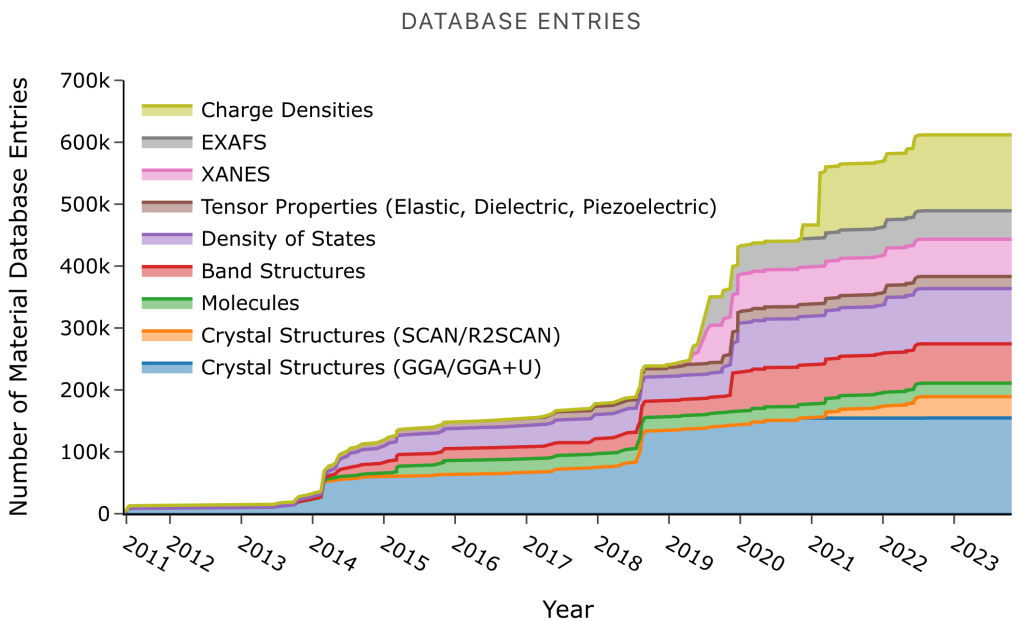
MATERIALS PROJECT


AFLOW
Automatic - FLOW for Materials Discovery


MATERIALSCLOUD

# The Data Flood in Materials Science

- High-throughput calculations (first-principles, molecular dynamics, etc.), combinatorial experiments, and open databases have increased significantly
- Manual analysis or brute-force simulation cannot keep pace
- Hidden patterns = discovery opportunities → need **Machine Learning**



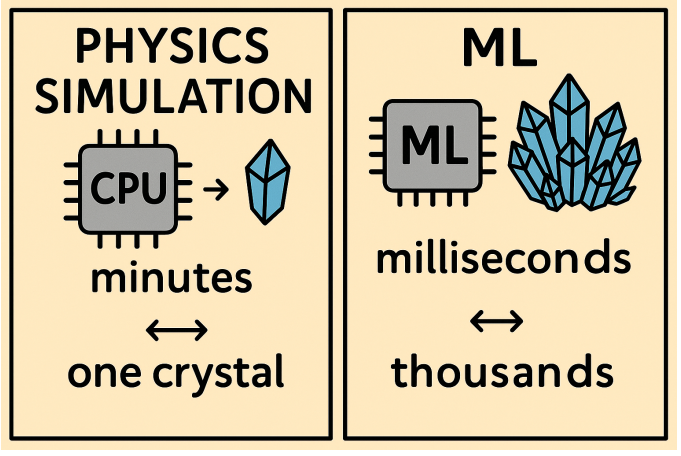
# Traditional Modeling vs Machine Learning

**Physics-based simulation**

- DFT / MD deliver high fidelity but at high cost: $\mathcal{O}$(minutes to hours) per structure.

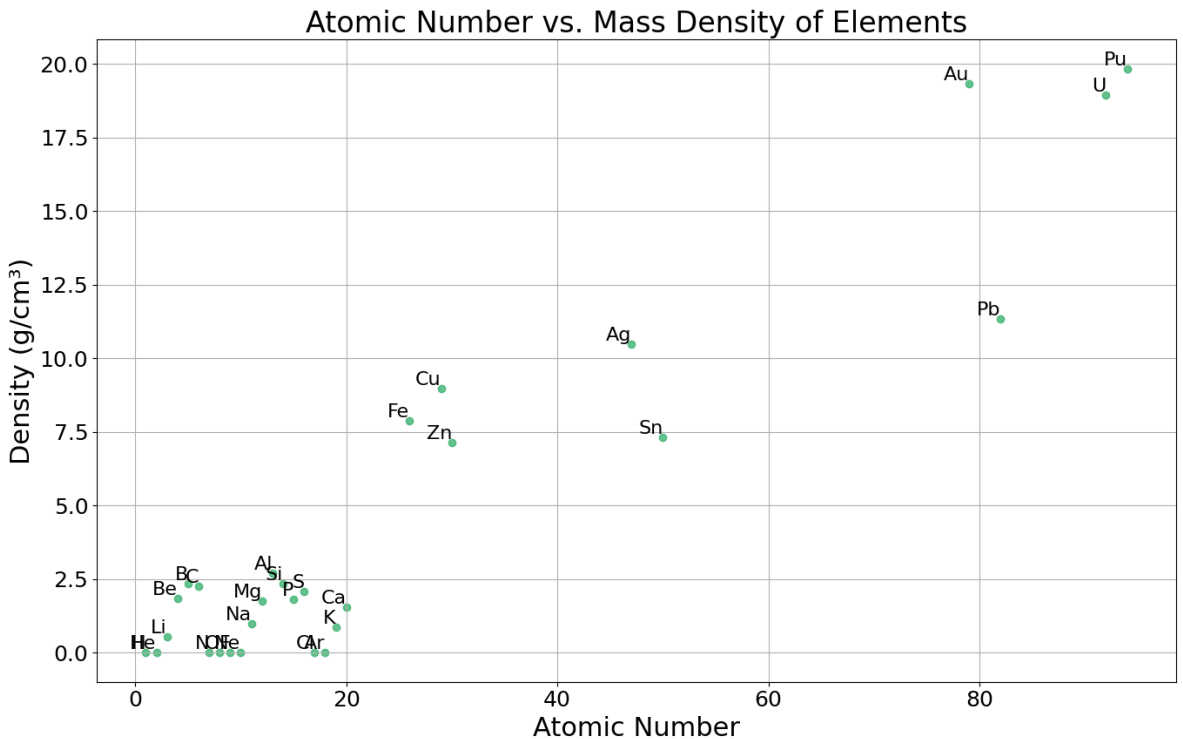
**ML surrogate model**

- Train once, then predict new materials in $\mathcal{O}$(milliseconds).
- Enables rapid screening and inverse design.



# Can We Learn the Mass Density from the Atomic Number?

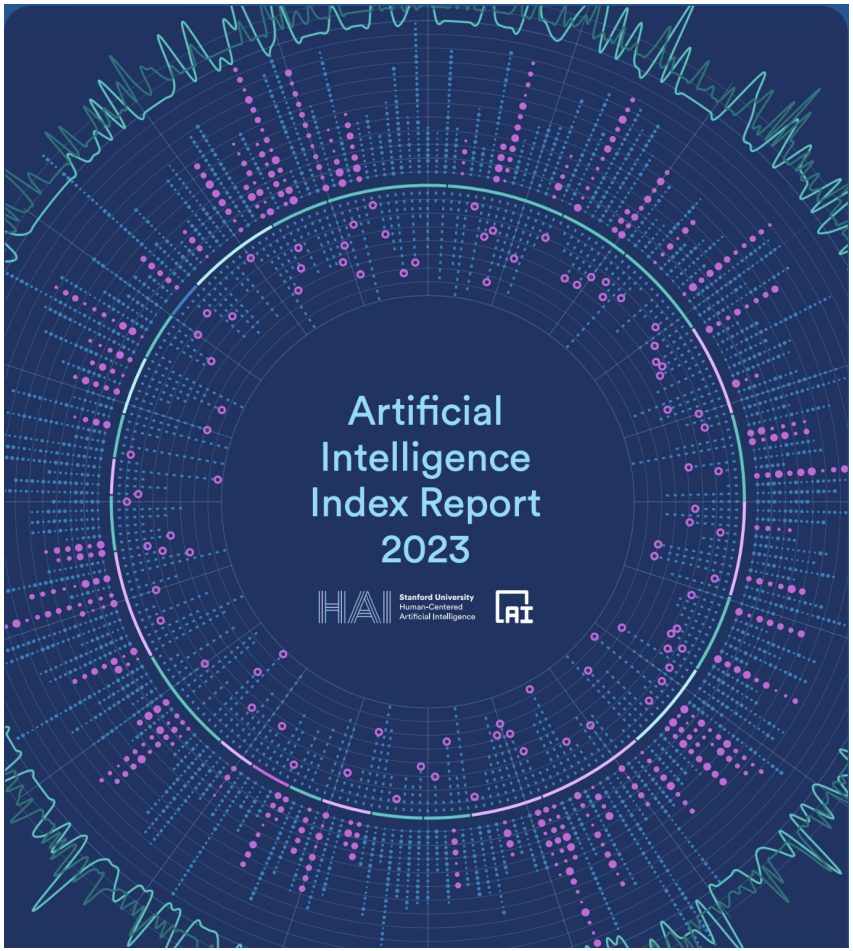
- Toy example: atomic number $Z$ vs mass density – relationship is non-linear
- **Question**: "Could an algorithm automatically learn this trend... and then far more complex ones like band gap or elasticity?"




To answer, we need ML basics: vocabulary, models, and evaluation.

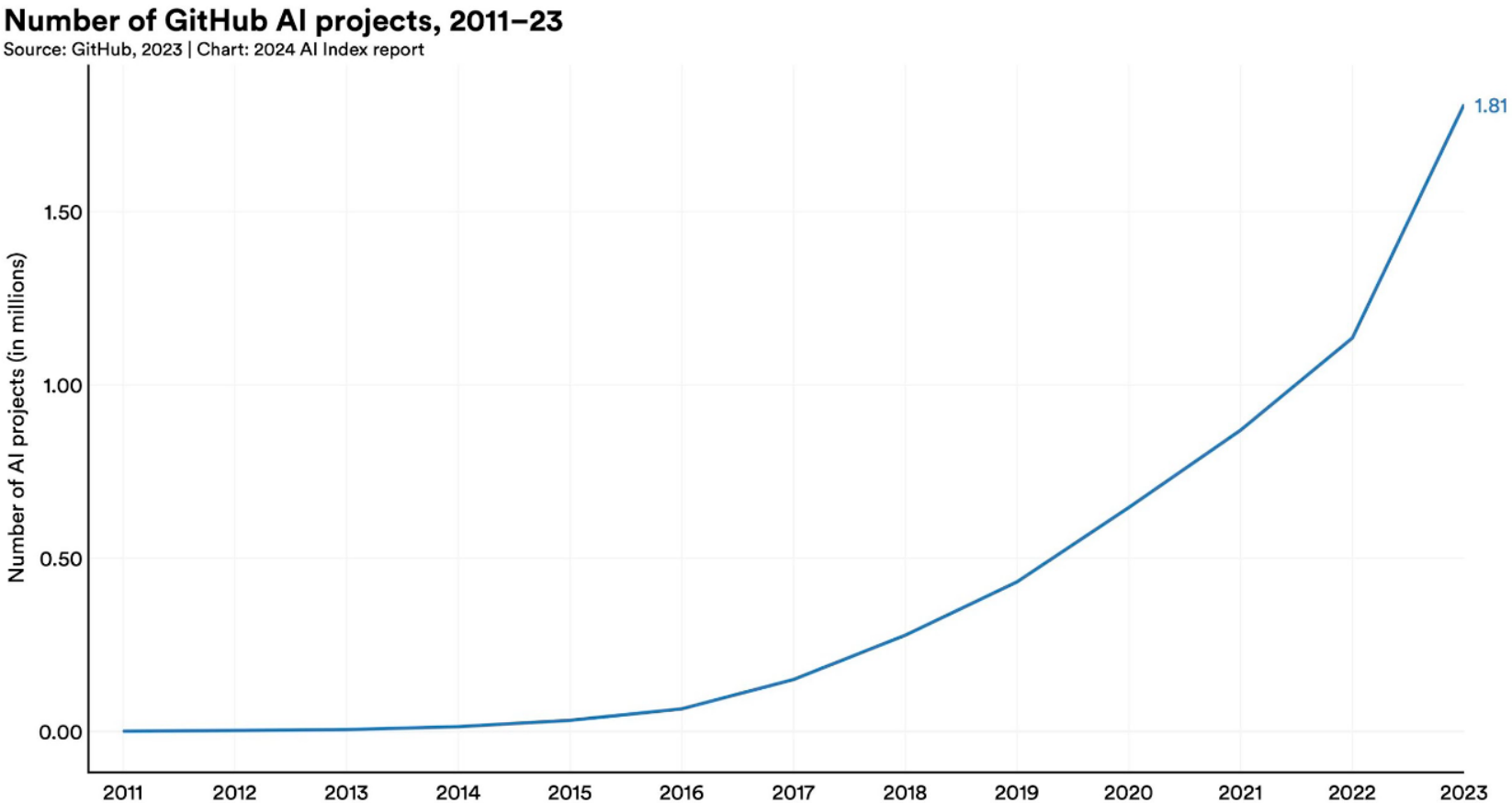
# What is AI?


# Some trends about AI



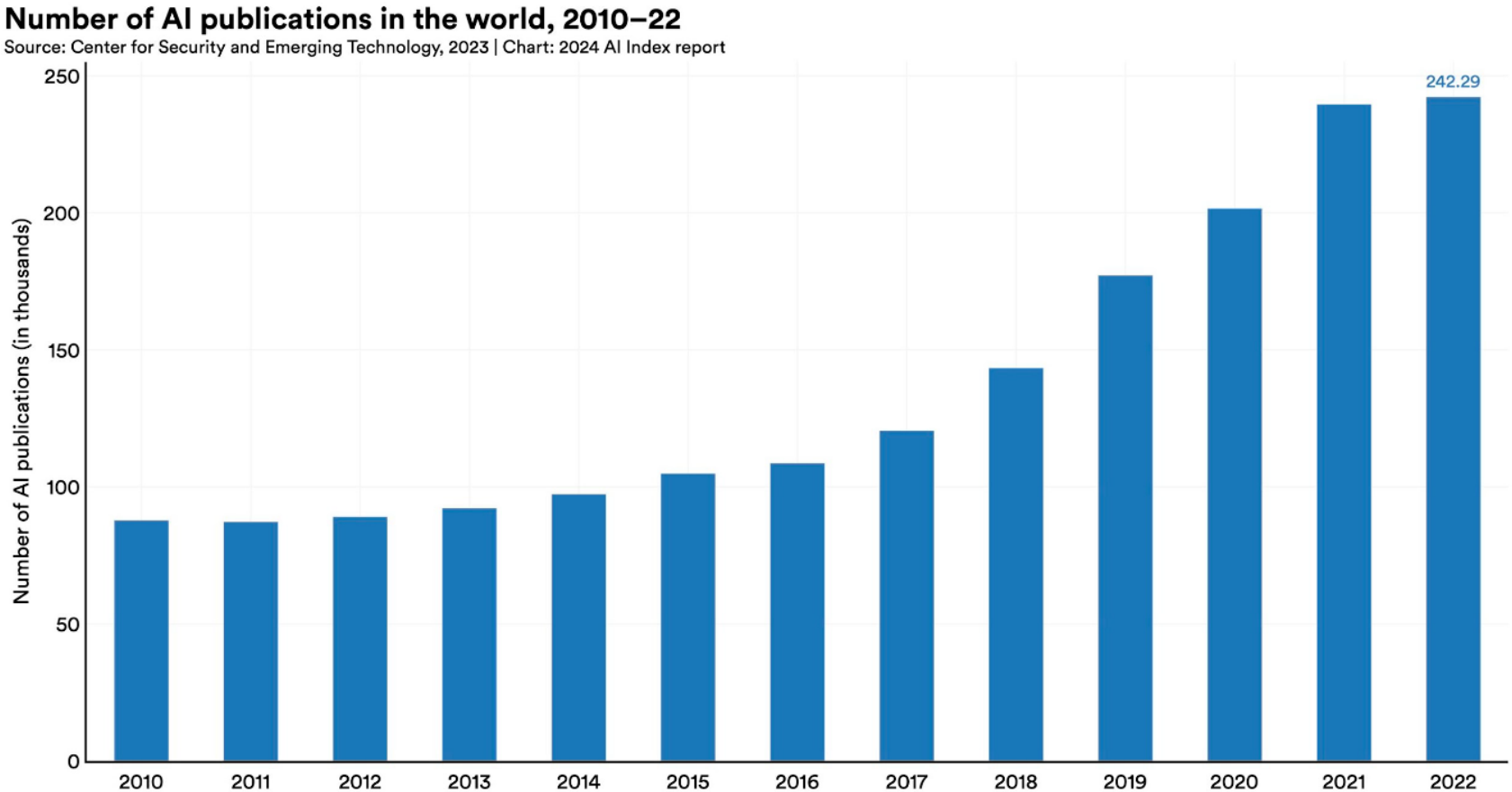

https://hai.stanford.edu/ai-index/2023-ai-index-report

# Some trends about AI



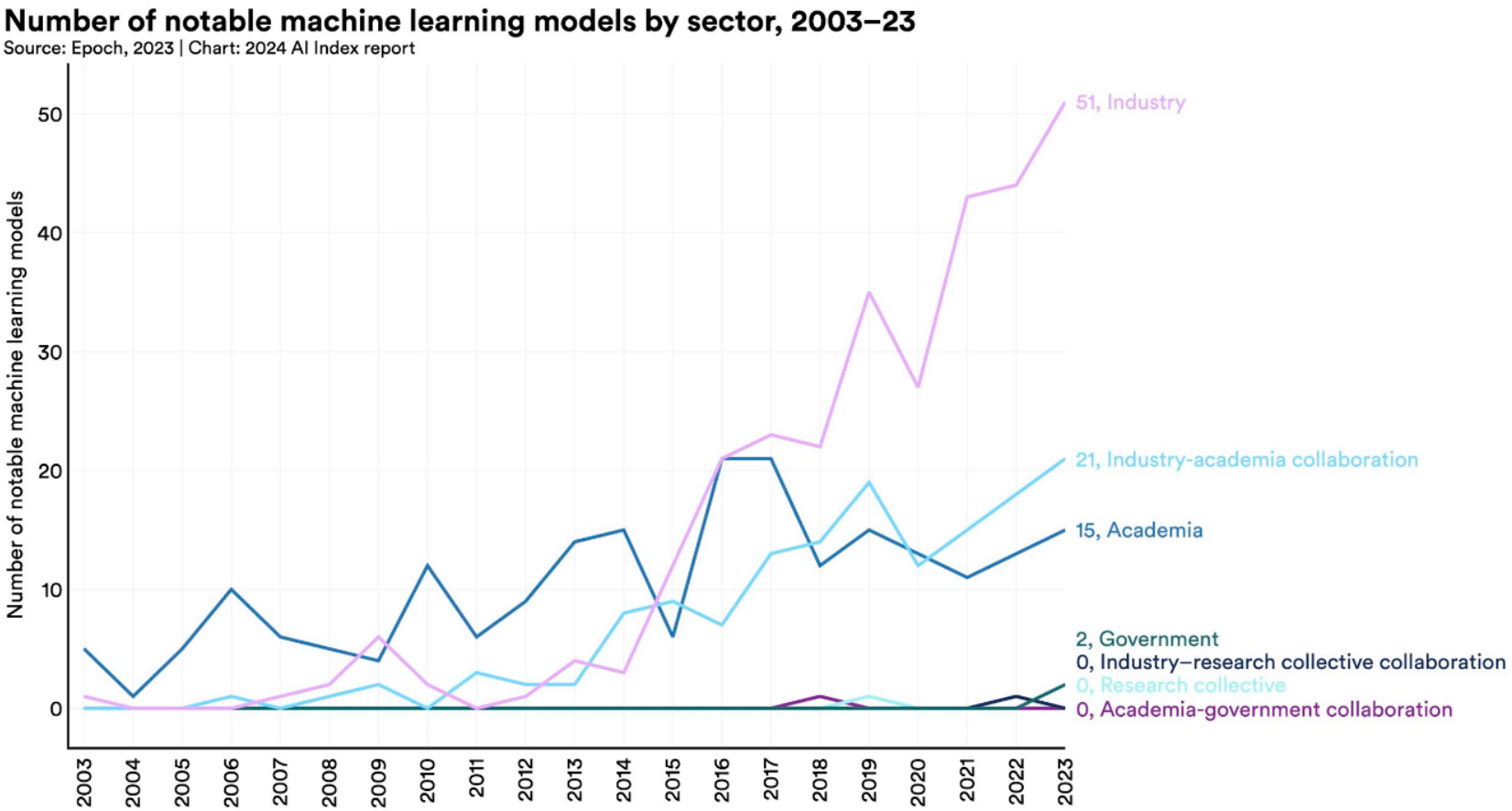

https://hai.stanford.edu/ai-index/2023-ai-index-report

# Some trends about AI



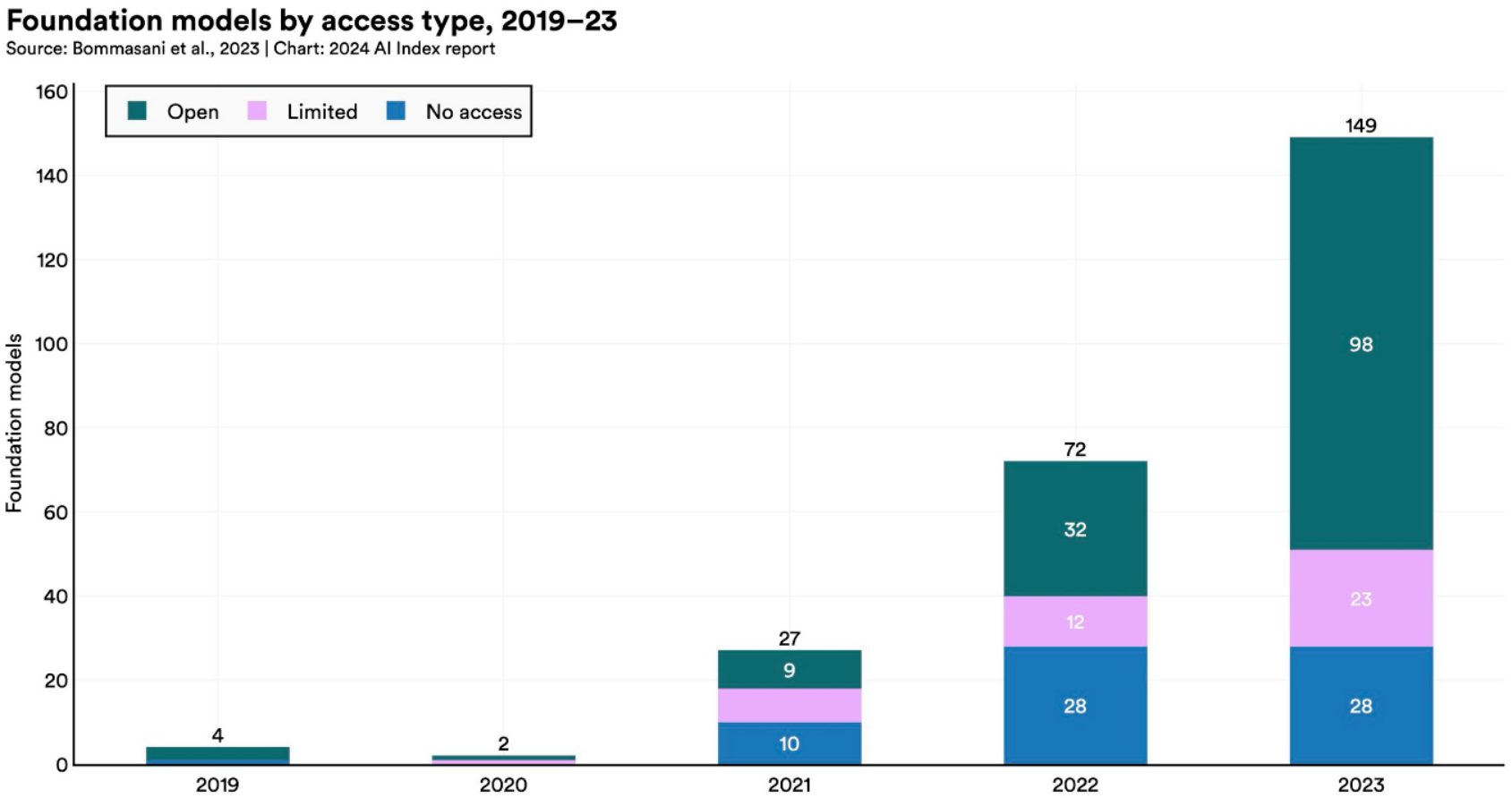

https://hai.stanford.edu/ai-index/2023-ai-index-report

# Some trends about AI



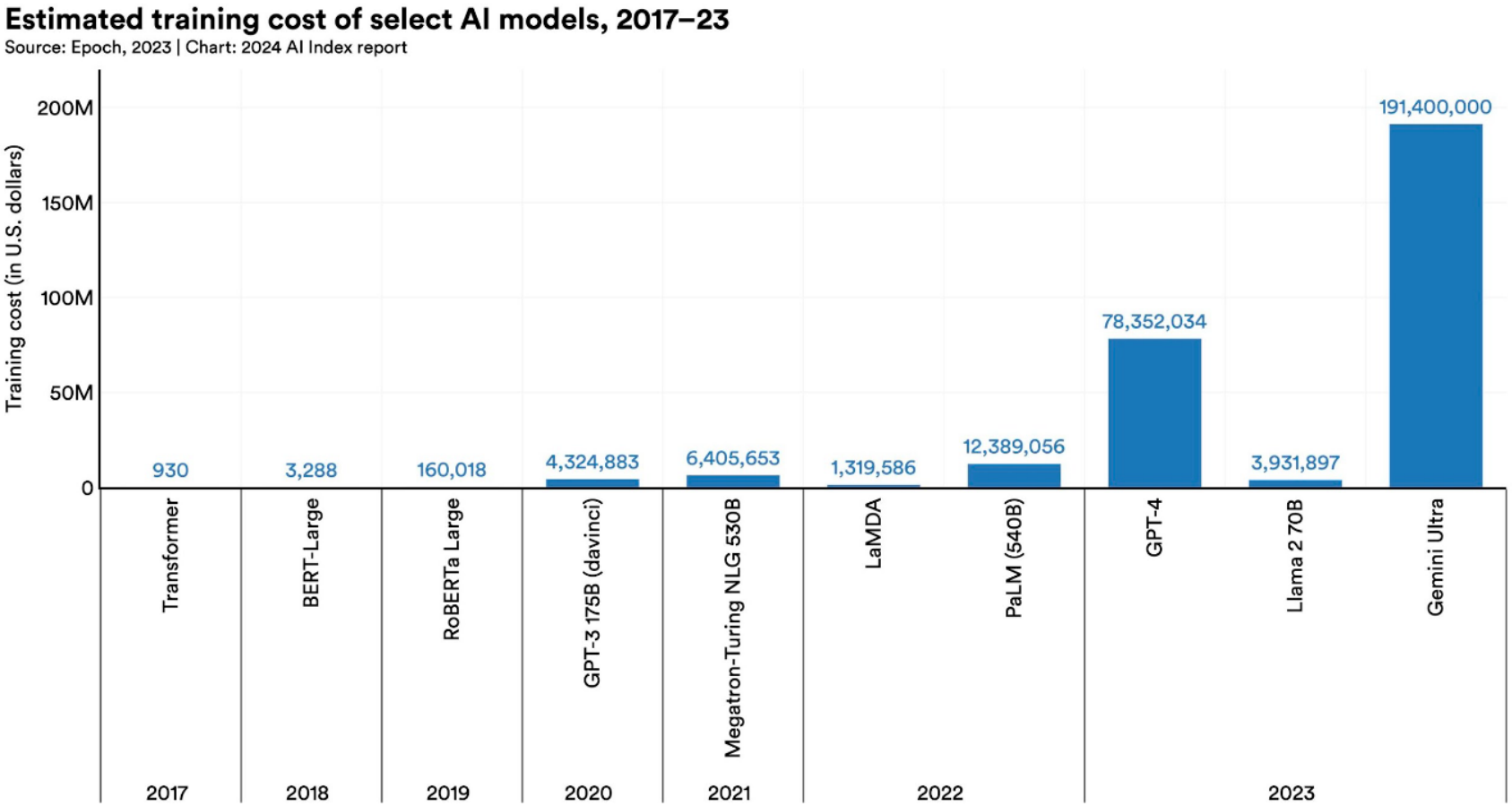

https://hai.stanford.edu/ai-index/2023-ai-index-report

# Some trends about AI




https://hai.stanford.edu/ai-index/2023-ai-index-report

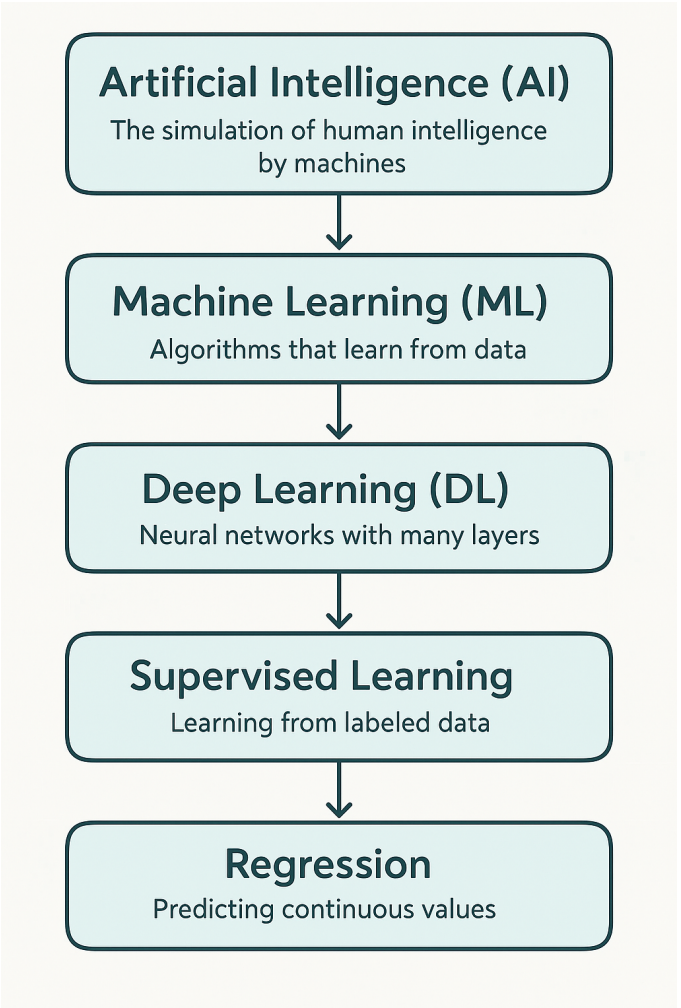
# Some trends about AI




https://hai.stanford.edu/ai-index/2023-ai-index-report

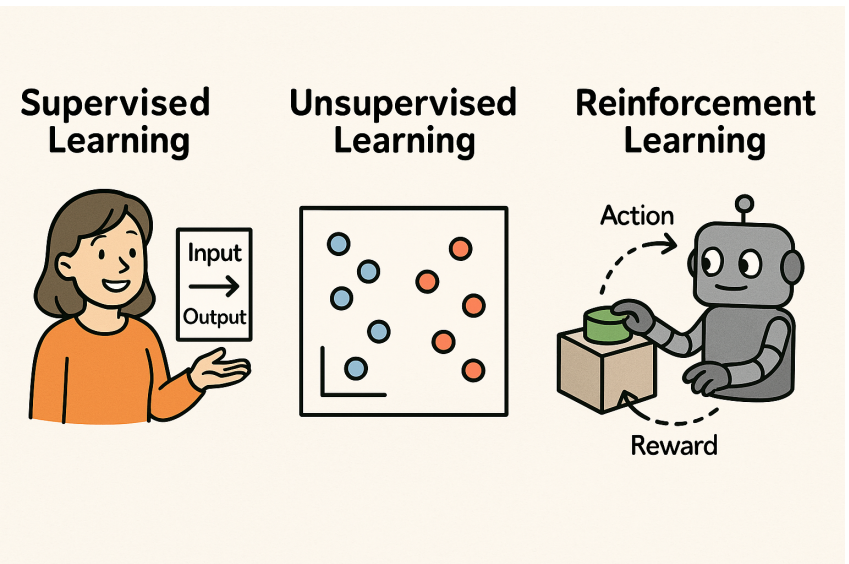
# Zoom-Out : What Is AI?

- **Artificial Intelligence (AI)** – umbrella term for algorithms that perform tasks normally requiring human intelligence (planning, perception, reasoning)
- **Machine Learning (ML)** – *subset* of AI; systems that *learn patterns from data* instead of being hard-coded
- **Deep Learning (DL)** – *subset* of ML; models based on multi-layer neural networks



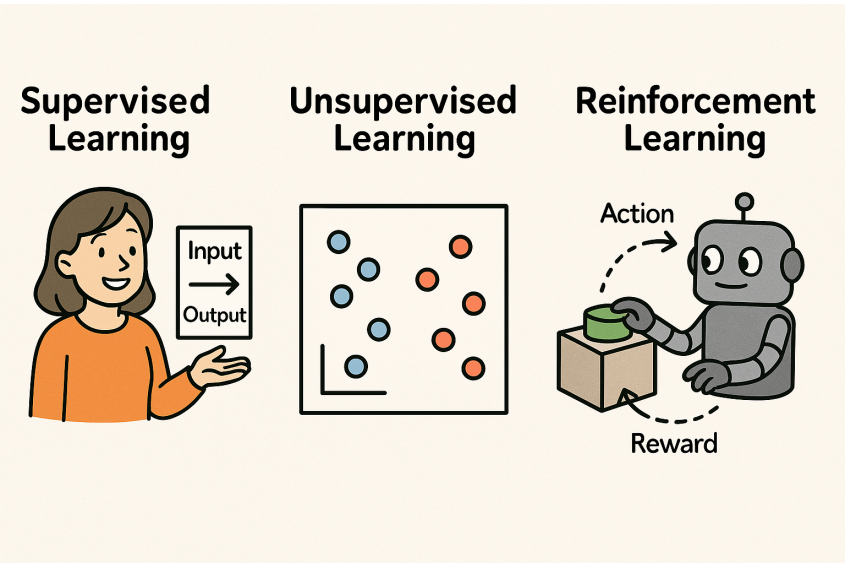
# Anatomy of Machine Learning

- Three core **learning paradigms**:
  1. **Supervised** – learn mapping from inputs $x$ to labels $y$
  2. **Unsupervised** – discover structure in unlabeled data
  3. **Reinforcement** – learn via trial-and-error rewards
- Each paradigm branches into many *methods* (linear regression, k-means, Q-learning)
- Common thread: define a goal (loss or reward), optimize it



# Where the Paradigms Help in Materials Science

| **Paradigm** | **Example materials tasks** |
|---|---|
| Supervised | Predict band gap, elastic modulus, phase label |
| Unsupervised | Cluster high-entropy alloys; reduce descriptor space with PCA; anomaly detection in synthesis logs |
| Reinforcement | Active-learning loop in self-driving lab; Bayesian optimization of alloy composition |




Property prediction dominates our immediate needs, so we now focus on the **supervised** learning paradigm.

# Anatomy of a Dataset

- A **dataset** = collection of *samples* (rows)
- Each sample records one material/formula
- Many columns make up the **features** (descriptors)
- One column is **label** (target property)

```
df_12_struct.head()
```

| | structure | log10(K_VRH) | density | vpa | packing fraction | mean ordering parameter shell 1 | mean ordering parameter shell 2 | mean ordering parameter shell 3 |
|---|---|---|---|---|---|---|---|---|
| 0 | [[1.01010394 3.03031182 0. ] Li, [3.030... | 1.477121 | 1.925667 | 17.820044 | 0.774182 | 0.411544 | 0.152153 | 0.102496 |
| 1 | [[2.32540495 2.38773782 3.89634095] Al, [0. 0.... | 1.963788 | 3.675852 | 11.659282 | 0.258595 | 0.421807 | 0.181921 | 0.146386 |
| 2 | [[0. 0. 0.] Fe, [1.76869807 1.76869807 0. ... | 2.307496 | 8.700535 | 11.066011 | 0.958160 | 0.333333 | 0.030303 | 0.014245 |
| 3 | [[4.91140724 0.82858616 4.10907037] F, [3.1732... | 1.301030 | 5.266007 | 20.142174 | 0.886432 | 0.476271 | 0.230530 | 0.181574 |
| 4 | [[4.68532551 4.68532749 4.68533345] Sc, [1.561... | 2.235528 | 8.631463 | 15.237572 | 0.734461 | 0.551982 | 0.241225 | 0.132669 |

Example snippet: structure, mass density, ordering parameters, packing fraction, bulk modulus

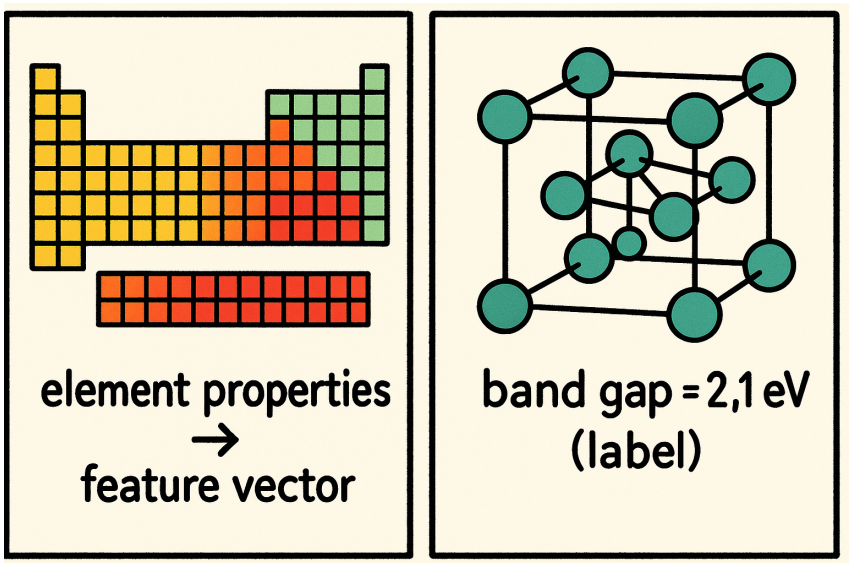
# Features & Labels in Materials Science

**Features (Descriptors)**

- Numerical representation of composition/structure
- Examples: average electronegativity, packing fraction, atomic structure descriptor, atomic-site symmetry

**Labels (Targets)**

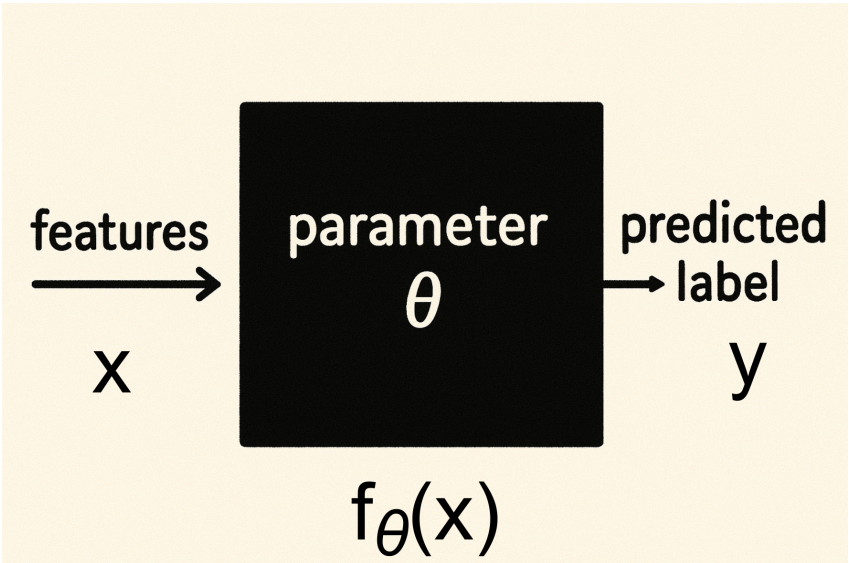
- Property we wish to predict:
  **Continuous** → band gap [eV], formation energy [eV/atom]
  **Categorical** → metal vs insulator, phase (hcp/fcc)




Descriptor (feature) vector feeds model; label guides learning

# Model, Parameters & Loss

- **Model** $\hat{y} = f_\theta(x)$ – mathematical function mapping features to label
- **Parameters** $\theta$ – weights the algorithm must learn
- **Loss** $L(y, \hat{y})$ – quantifies "how wrong" predictions are:
  **Regression** $\rightarrow$ MSE
  **Classification** $\rightarrow$ cross-entropy
- Learning = find $\theta^* = \arg\min_\theta \sum L(y, \hat{y})$

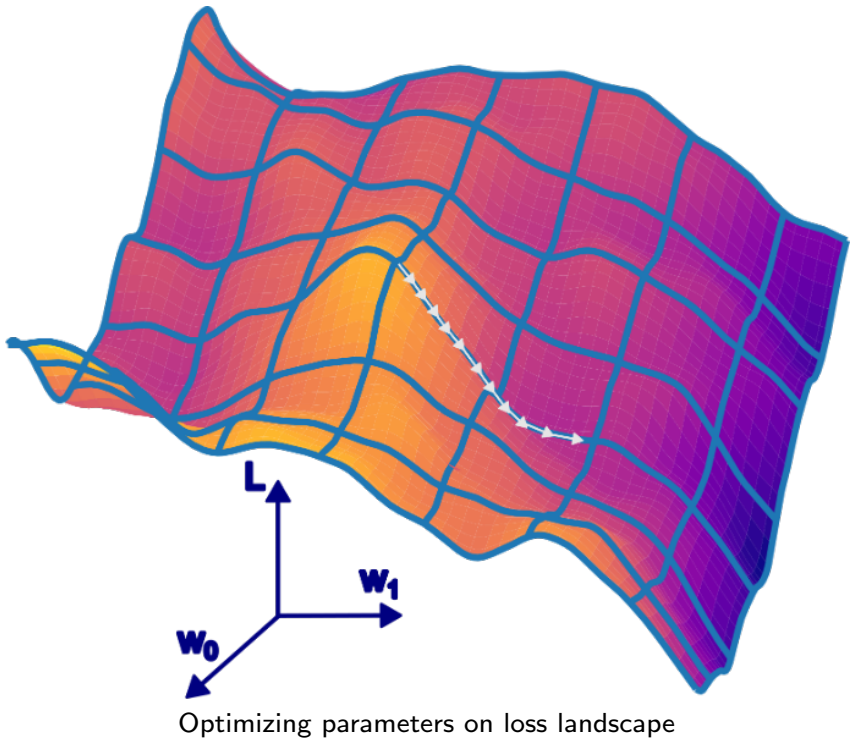


Black-box ML model

Feature scaling: Standardize descriptors so weight magnitudes are comparable

# Model, Parameters & Loss

- **Model** $\hat{y} = f_\theta(x)$ – mathematical function mapping features to label
- **Parameters** $\theta$ – weights the algorithm must learn
- **Loss** $L(y, \hat{y})$ – quantifies "how wrong" predictions are:
  **Regression** $\rightarrow$ MSE
  **Classification** $\rightarrow$ cross-entropy
- Learning = find $\theta^* = \arg\min_\theta \sum L(y, \hat{y})$

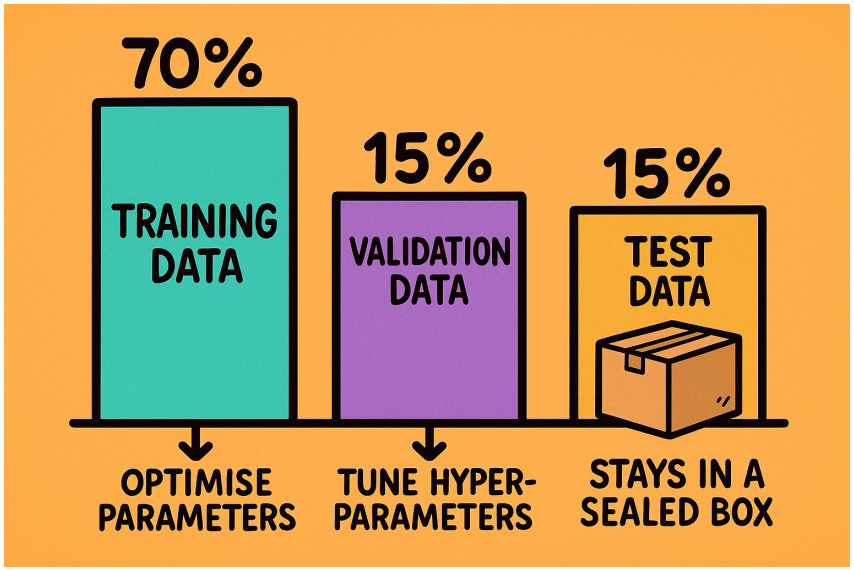


Optimizing parameters on loss landscape

Feature scaling: Standardize descriptors so weight magnitudes are comparable

# Train · Validation · Test Split

- **Training set** – model sees labels; fits weights
- **Validation set** – monitors generalization; tunes hyper-parameters, triggers early stopping
- **Test set** – sealed until final evaluation; unbiased estimate of real-world performance
- Typical split: 70 / 15 / 15 % (flexible)

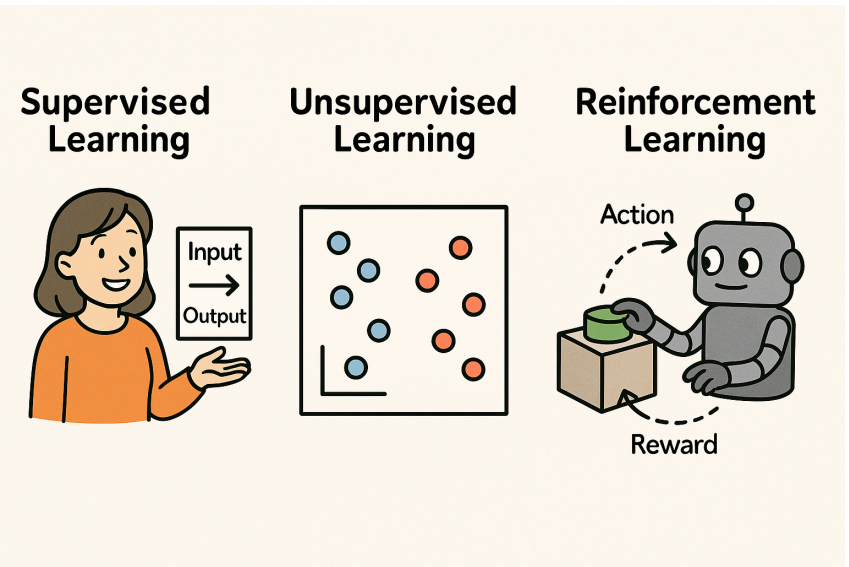



## Golden Rule

Never use the test set during model development.

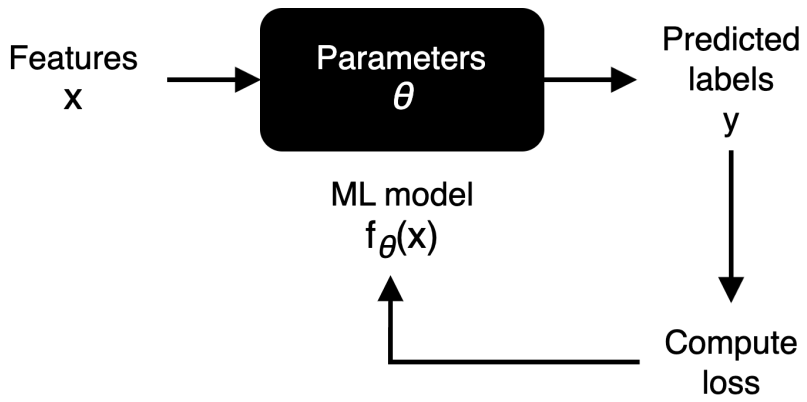
# Three Ways to Learn from Data

- **Supervised learning**: learn mapping $x \mapsto y$ from labelled data
- **Unsupervised learning**: find structure/patterns in *unlabelled* data
- **Reinforcement learning**: learn a policy by maximizing cumulative reward through interaction
- Common pattern: define objective $\Rightarrow$ optimize it



# Supervised Learning in Materials Science

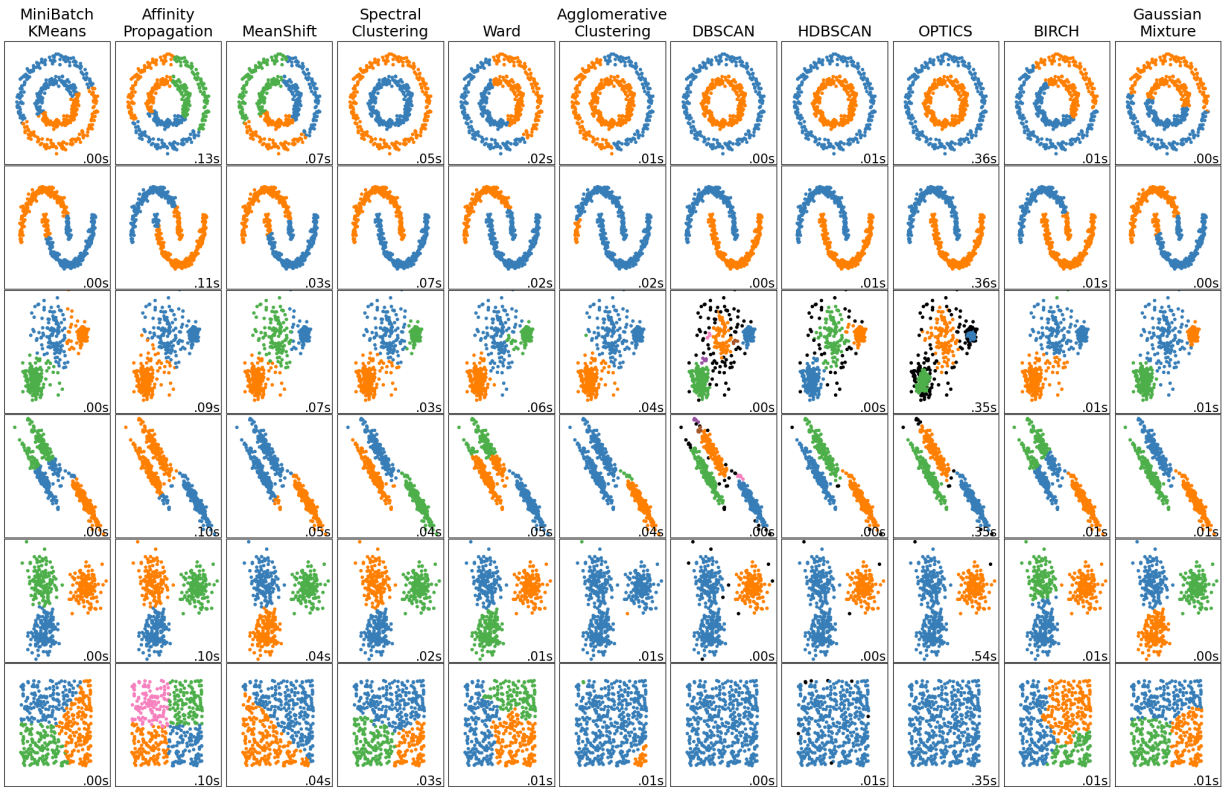
- Inputs: composition / structure descriptors
- Targets: *continuous* properties (band gap, elastic modulus) or *categories* (metal vs insulator)
- Typical algorithms: linear & kernel regression, random forests, neural networks
- Enables high-throughput **property prediction** and rapid screening




Supervised learning workflow: Descriptor vector → model → predicted property

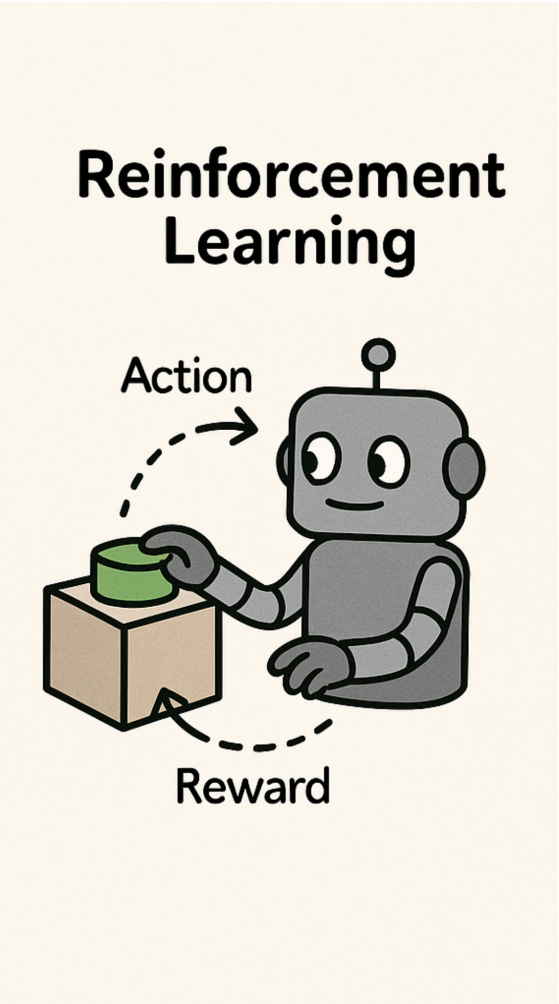
# Unsupervised Learning – When Labels Are Scarce

- Goal: discover latent structure.
- Tasks: cluster unknown alloys, reduce high-dimensional descriptor space via PCA, anomaly detection in synthesis logs.
- Methods: k-means, DBSCAN, autoencoders, PCA, t-SNE.



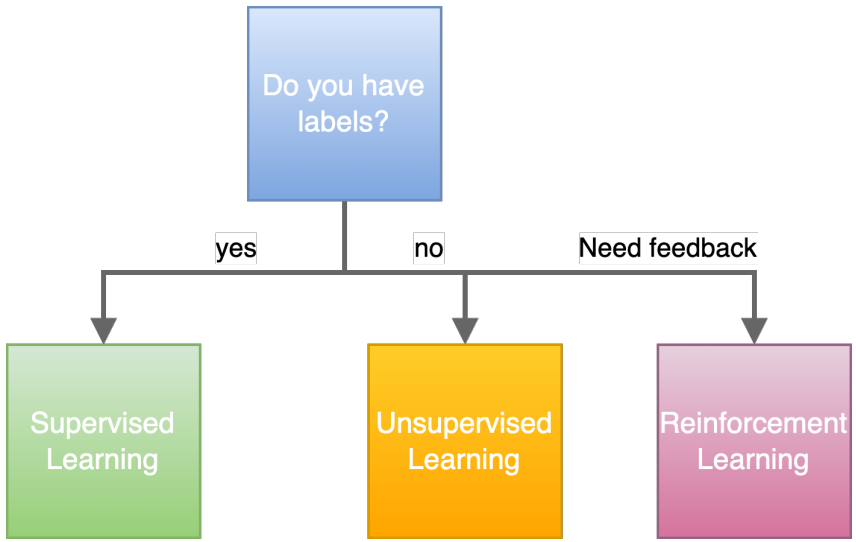
# Reinforcement Learning – When Labels Are Scarce

- Goal: choose actions to maximize reward.
- Tasks: active-learning loop in self-driving lab, Bayesian optimization of composition, autonomous furnace tuning.
- Methods: Q-learning, policy gradients, GP-based Bayesian optimization.



# Which Paradigm Fits Your Problem?

- **Do you have labels?**
  - Yes ⇒ **Supervised**.
  - No ⇒ **Unsupervised**.
- Need sequential decisions and real-time feedback? ⇒ **Reinforcement**.
- Example walk-through:
  - Predict elastic modulus → **Supervised**
  - Cluster 10 000 unknown materials into categories (metal, insulator, oxides, etc.) → **Unsupervised**
  - Decide next sample or processing step in autonomous lab → **RL**




## Take-away

We'll focus on supervised learning next, since it underpins most materials-property prediction tasks.

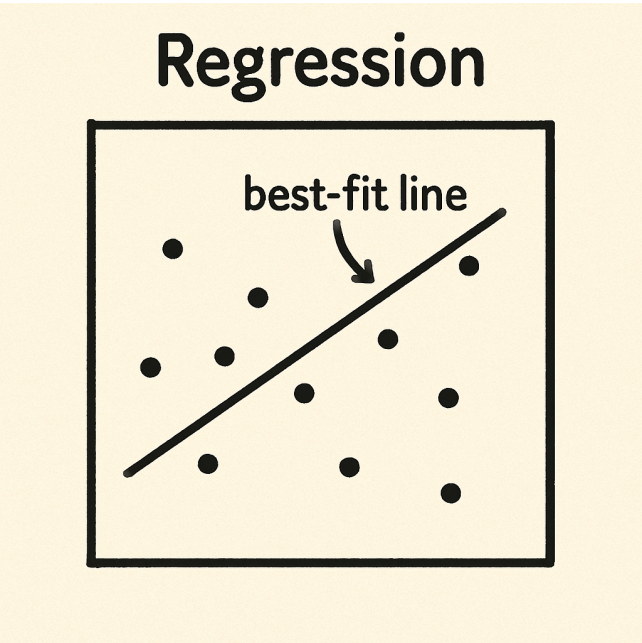
# Two Supervised Question Types

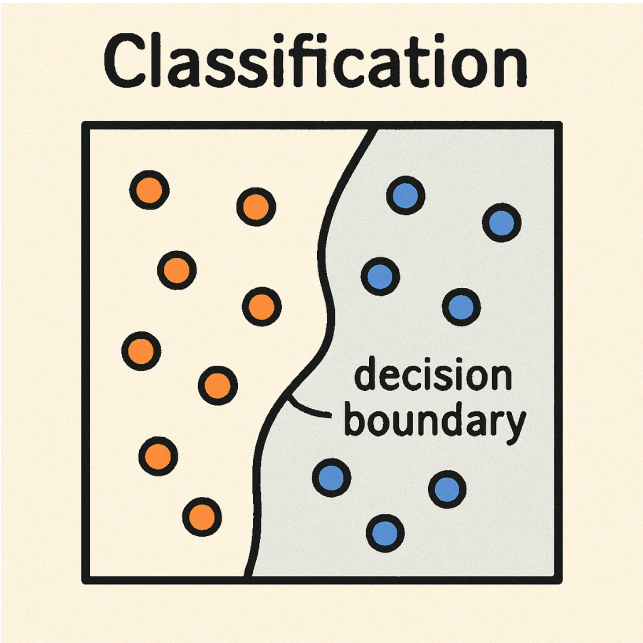
**Regression**

- Predict a *continuous* value
- Example: band gap (eV), elastic modulus (GPa)
- Loss: MSE / MAE

**Classification**

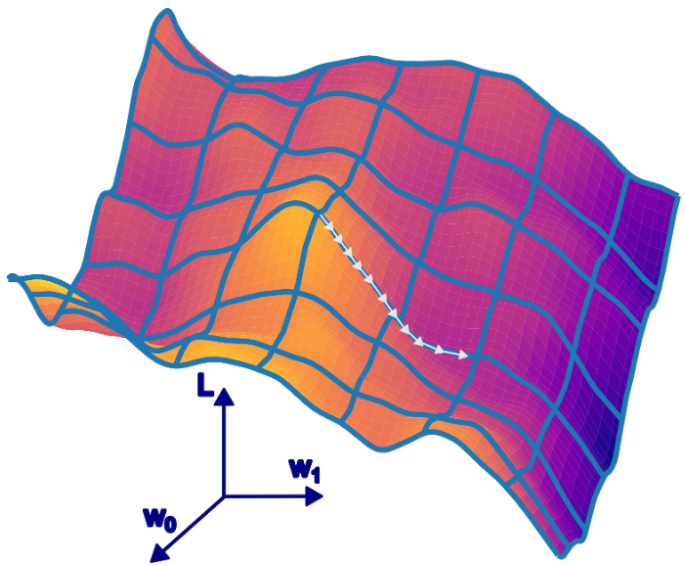
- Predict a *category* label
- Example: metal vs insulator, crystal system (hcp/fcc)
- Loss: cross–entropy



# Why We Need a Loss

- Model outputs a *prediction* $\hat{y}$; reality provides the *true value* $y$
- We need a numerical measure of "how wrong" each prediction is
- **Loss function** $L(y, \hat{y})$ quantifies this error
- Training = adjust parameters to minimize $\sum L(y, \hat{y})$



# Mean–Squared Error (MSE)

- Standard loss for regression tasks
- Formula: $\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$
- Squared term penalizes large errors more strongly
- Differentiable $\Rightarrow$ gradient-based optimization works smoothly

## Loss Functions Overview

- **Mean–Squared Error (MSE)**
  $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ - Squared term penalizes large errors more strongly

- **Mean Absolute Error (MAE)** – also for regression and more robust to outliers
  $\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$

- **Root Mean–Squared Error (RMSE)**
  $\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$

- **Binary Cross Entropy** – for classification
  $\text{BCE} = \frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$

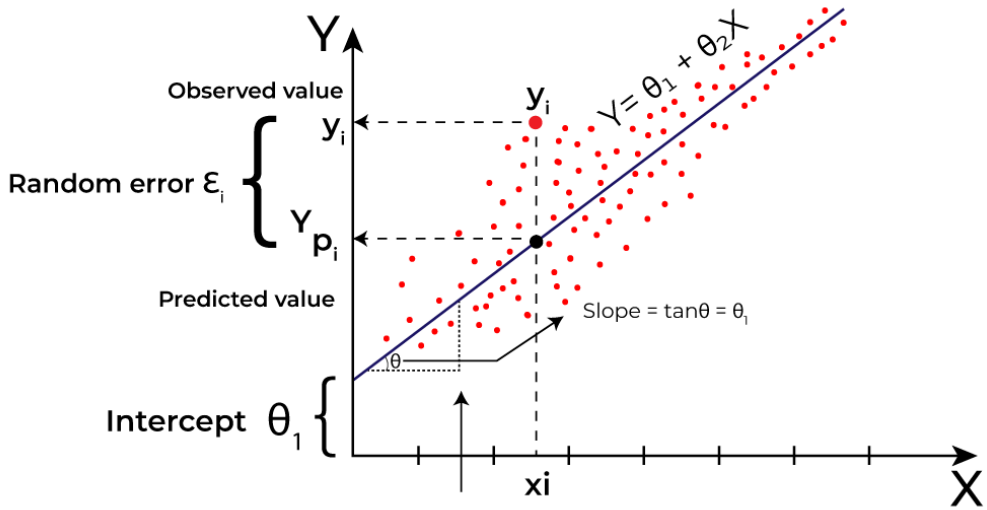
- Choice of loss depends on task and distribution of targets

### Key Take-away

Pick a loss that matches your prediction type; for regression we'll typically use MSE.

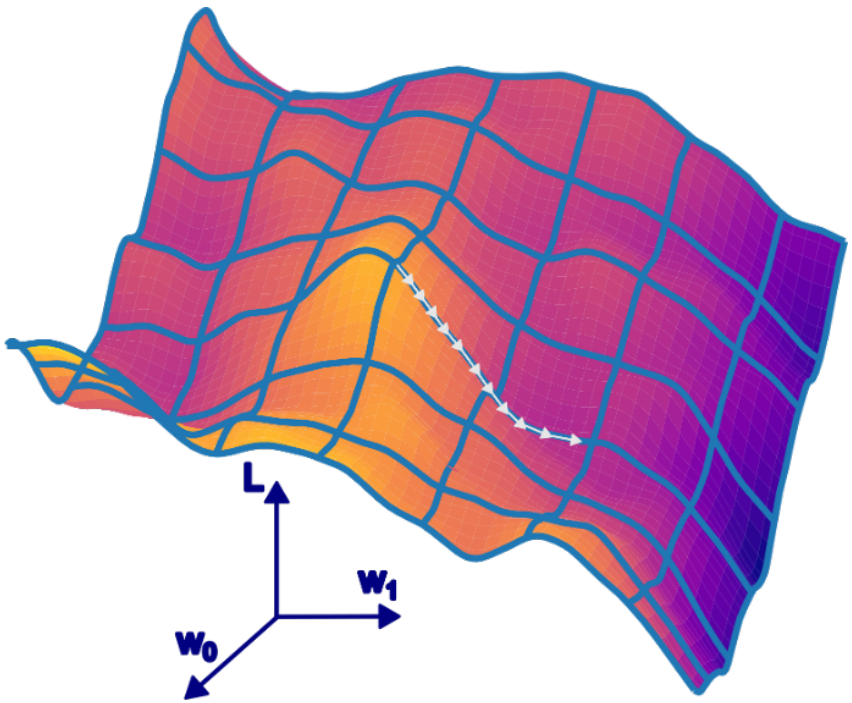
# Linear Model & Assumptions

- Linear regression is a type of supervised machine-learning algorithm
- Maps the data points with most optimized linear functions
- Hypothesis: $\hat{y} = \theta_1 + \theta_2 x$
- **Assumptions**:
  - Relationship between input and output is linear
  - Errors are independent and identically distributed
- Goal: find $\theta_1, \theta_2$ that minimize MSE.



# Loss Function in Linear Regression

- Recall MSE loss
  $L = \frac{1}{N} \sum_i^N (\hat{y}_i - y_i)^2$
- Calculates the average of the squared errors between the predicted values and the actual values
- Purpose is to determine optimal values $\theta_1, \theta_2$ for the model $\hat{y} = \theta_1 + \theta_2 x$
- How do we find $\theta_1, \theta_2$?
- This is done by minimizing the loss function, for example by gradient descent



# Visualizing MSE: Geometric Intuition

- Each residual $r_i = y_i - \hat{y}_i$ shown as vertical bar
- Squaring = shaded area of each residual square
- **Best-fit line** minimizes total shaded area

Visualization

# Finding Model Parameters (Closed-form vs. Gradient Descent)

**Closed-form** (Normal equation):

$$\boldsymbol{\theta} = (\mathbf{x}^\top \mathbf{x})^{-1} \mathbf{x}^\top \mathbf{y}$$

- Good for small to moderate dimensionality of features
- Requires matrix inversion

**Gradient Descent** (iterative):

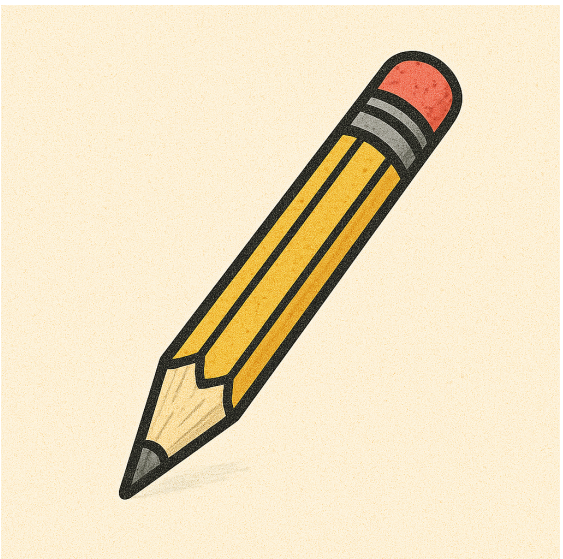
$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \cdot \text{“gradient update”}$$

- Scales to large datasets; underlies neural networks
- Converges if learning rate $\alpha$ chosen properly

Let's compute the gradient updates specifically for a simple example.

# Computing the Gradient Update

- Consider a simple linear regression model: $\hat{y} = \theta_1 + \theta_2 x$
- Let's compute the gradient update.



## Linear regression: Gradient descent

Deriving the gradient update

$$\theta = \theta - \alpha \cdot \text{"gradient update"}$$
$$= \theta - \alpha \cdot \Delta\theta$$

Consider a simple regression model:

$$\hat{y} = \theta_1 + \theta_2 \cdot x$$

This means we need to calculate how to update $\theta_1$ and $\theta_2$, so

$$\theta_1 = \theta_1 - \alpha \Delta\theta_1 = \theta_1 - \alpha \frac{\partial L(\theta_1, \theta_2)}{\partial \theta_1}$$
$$\theta_2 = \theta_2 - \alpha \Delta\theta_2 = \theta_2 - \alpha \frac{\partial L(\theta_1, \theta_2)}{\partial \theta_2}$$

Recall that $L(\theta_1, \theta_2) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$

(MSE loss)

For $\theta_1$, we then obtain

$$\frac{\partial L}{\partial \theta_1} = \frac{\partial}{\partial \theta_1} \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$
$$= \frac{1}{N} \sum_{i=1}^{N} 2 (\hat{y}_i - y_i) \frac{\partial}{\partial \theta_1} (\hat{y}_i - y_i)$$
$$= \frac{1}{N} \sum_{i=1}^{N} 2 (\hat{y}_i - y_i) \frac{\partial}{\partial \theta_1} (\theta_1 + \theta_2 x - y_i)$$
$$= \frac{1}{N} \sum_{i=1}^{N} 2 (\hat{y}_i - y_i)$$

And for $\theta_2$, we obtain

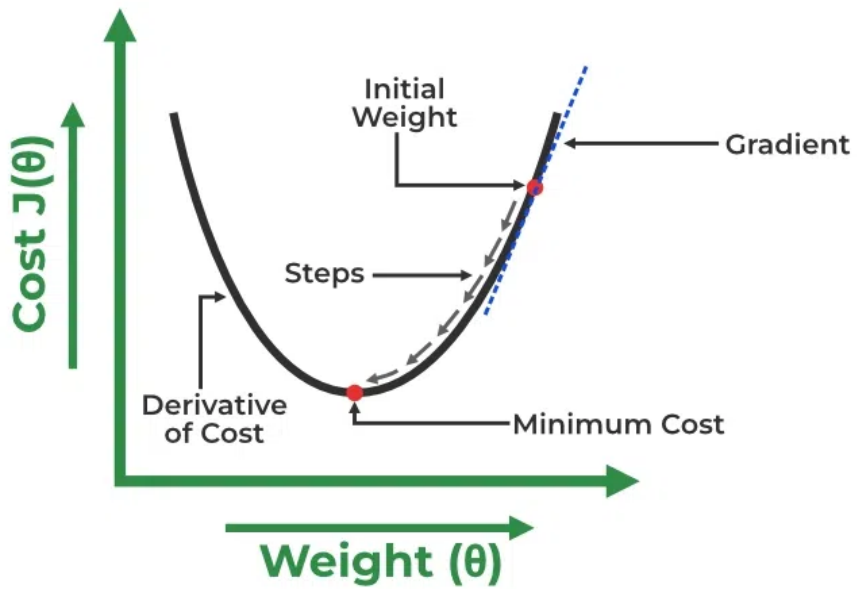
$$\frac{\partial L}{\partial \theta_2} = \dots = \frac{1}{N} \sum_{i=1}^{N} 2 (\hat{y}_i - y_i) \frac{\partial}{\partial \theta_2} (\theta_1 + \theta_2 \cdot x - y_i)$$
$$= \frac{1}{N} \sum_{i=1}^{N} 2 (\hat{y}_i - y_i) x$$

We obtain the following gradient updates:

$$\theta_1 = \theta_1 - \alpha \frac{2}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)$$
$$\theta_2 = \theta_2 - \alpha \frac{2}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i) x$$

# Computing the Gradient Update

- We derived the following gradient updates:
- Gradient update for intercept:
  $\theta_1 = \theta_1 - \alpha \frac{\partial L(\theta_1, \theta_2)}{\partial \theta_1} = \theta_1 - \alpha \frac{2}{N} \sum_i^N (\hat{y}_i - y_i)$
- Gradient update for slope:
  $\theta_2 = \theta_2 - \alpha \frac{\partial L(\theta_1, \theta_2)}{\partial \theta_2} = \theta_2 - \alpha \frac{2}{N} \sum_i^N (\hat{y}_i - y_i) \cdot x_i$



# Beyond Linear Regression

| | |
|---|---|
| Simple Linear Regression | $y = b_0 + b_1 x_1$ |
| Multiple Linear Regression | $y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$ |
| Polynomial Linear Regression | $y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$ |

# Regression Metrics

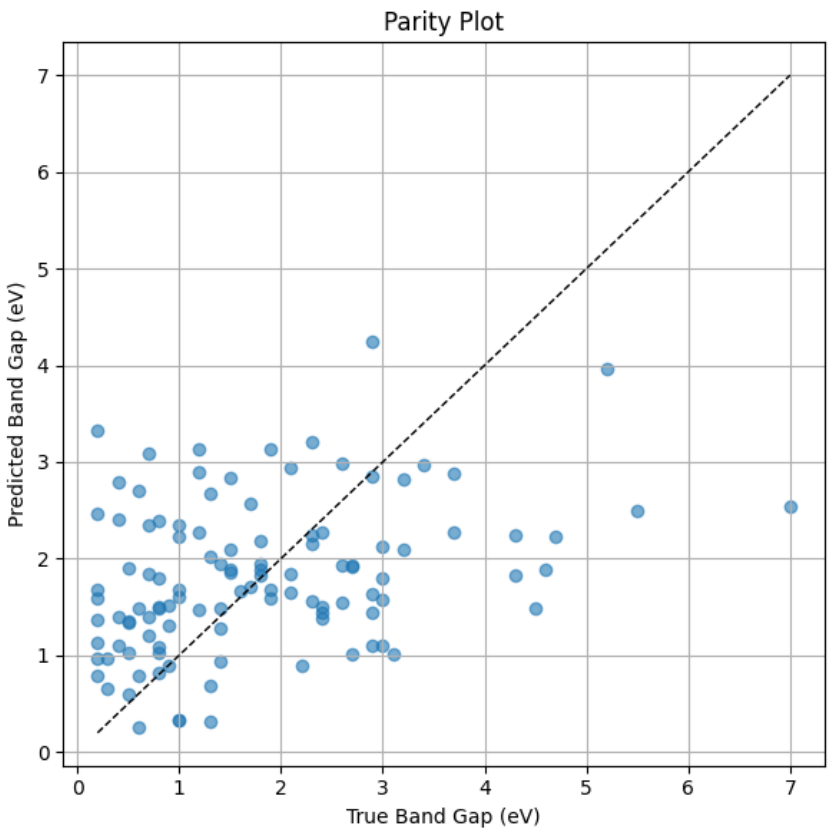
| Metric | Formula | Intuition |
|---|---|---|
| MSE | $\frac{1}{N}\sum_i^N(\hat{y}_i - y_i)^2$ | Penalizes large errors (squared) |
| MAE | $\frac{1}{N}\sum_i^N \lvert\hat{y}_i - y_i k\rvert$ | More robust to outliers |
| $R^2$ | $1 - \mathrm{SS}_{res}/\mathrm{SS}_{tot}$ | Fraction of variance explained (1=perfect) |

with
$\mathrm{SS}_{res} = \sum_i^N(\hat{y}_i - y_i)^2$
$\mathrm{SS}_{tot} = \sum_i^N(\bar{y} - y_i)^2$

- Always pair $R^2$ with an absolute error metric.
- Choose metric relevant to scientific tolerances.

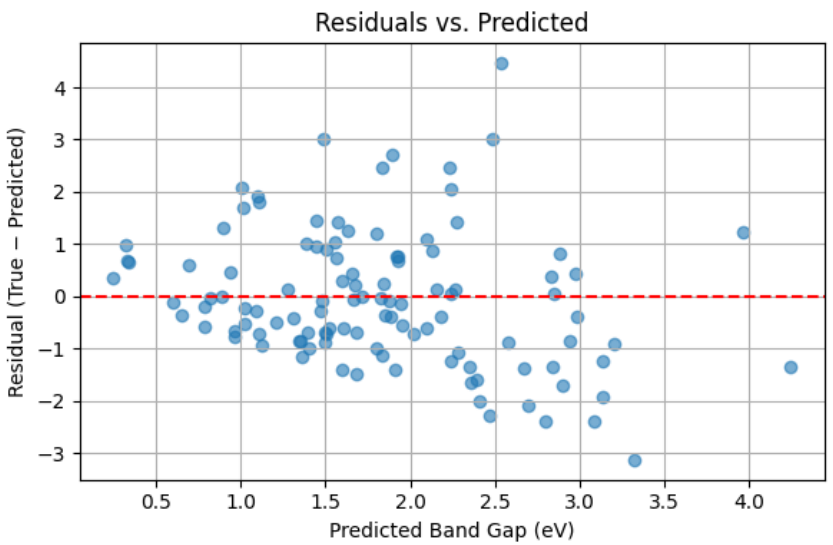
# Predicted vs Actual Plot (Parity Plot)

- Quick visual sanity check: perfect model lies on 45° line.
- Systematic bias shows as slope $\neq 1$.
- Colour/size points by uncertainty or density.



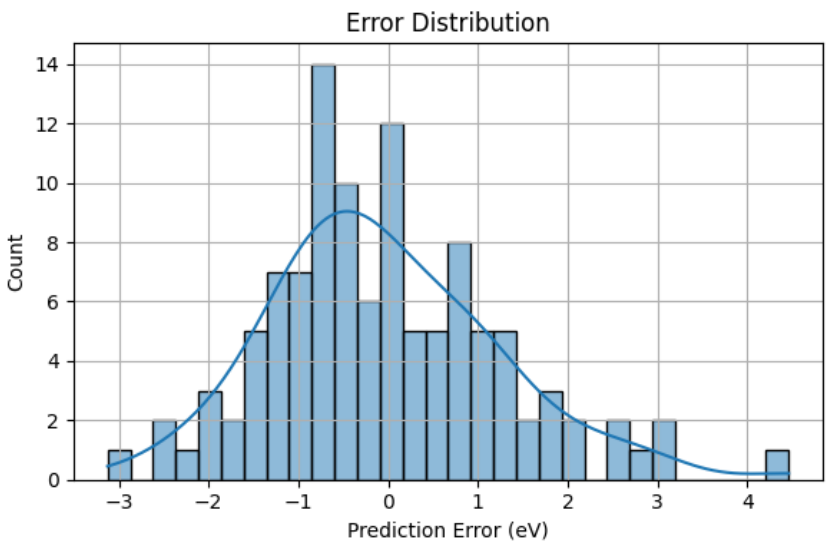
# Residual Analysis

- Plot residual $y - \hat{y}$ vs predicted $\hat{y}$.
- Patterns reveal issues:
  - Funnel shape $\Rightarrow$ non-constant variance.
  - Curved trend $\Rightarrow$ missing non-linear term.
- Suggest remedies: log-transform target, add polynomial features, or change model.



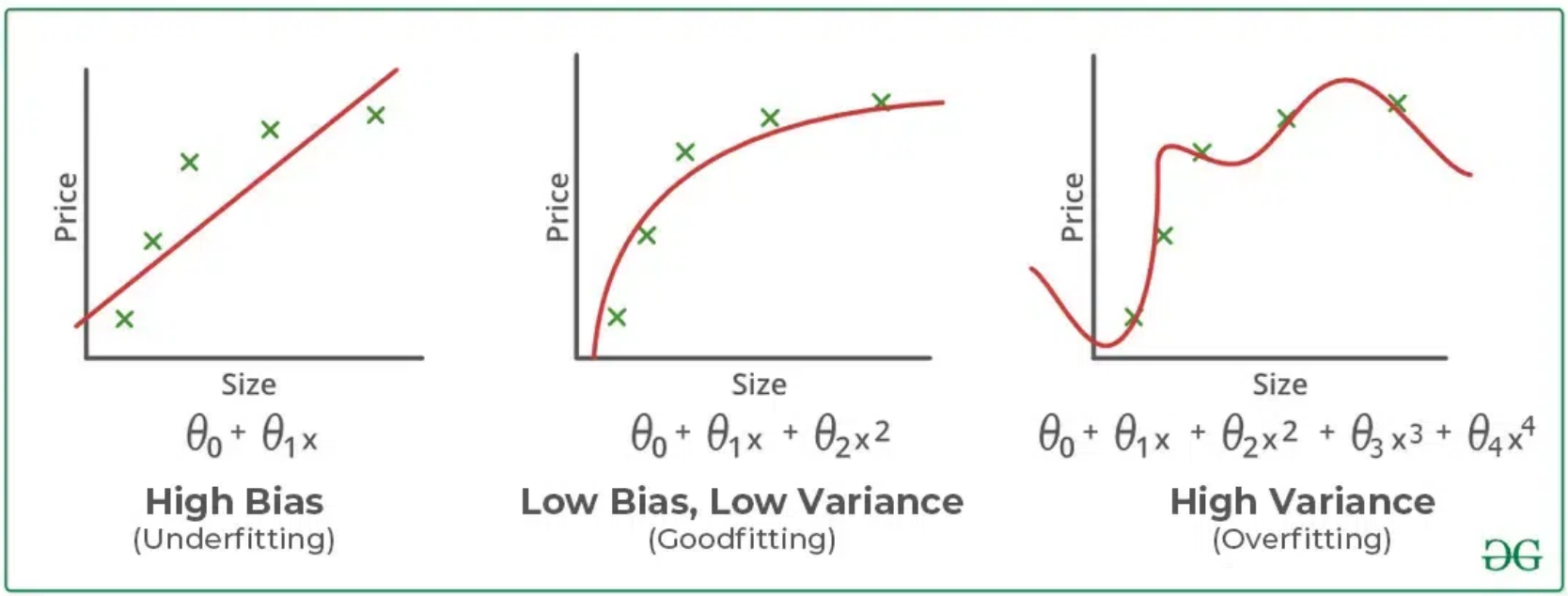
# Residual Analysis

- Plot residual $y - \hat{y}$ vs predicted $\hat{y}$.
- Patterns reveal issues:
  - Funnel shape $\Rightarrow$ non-constant variance.
  - Curved trend $\Rightarrow$ missing non-linear term.
- Suggest remedies: log-transform target, add polynomial features, or change model.



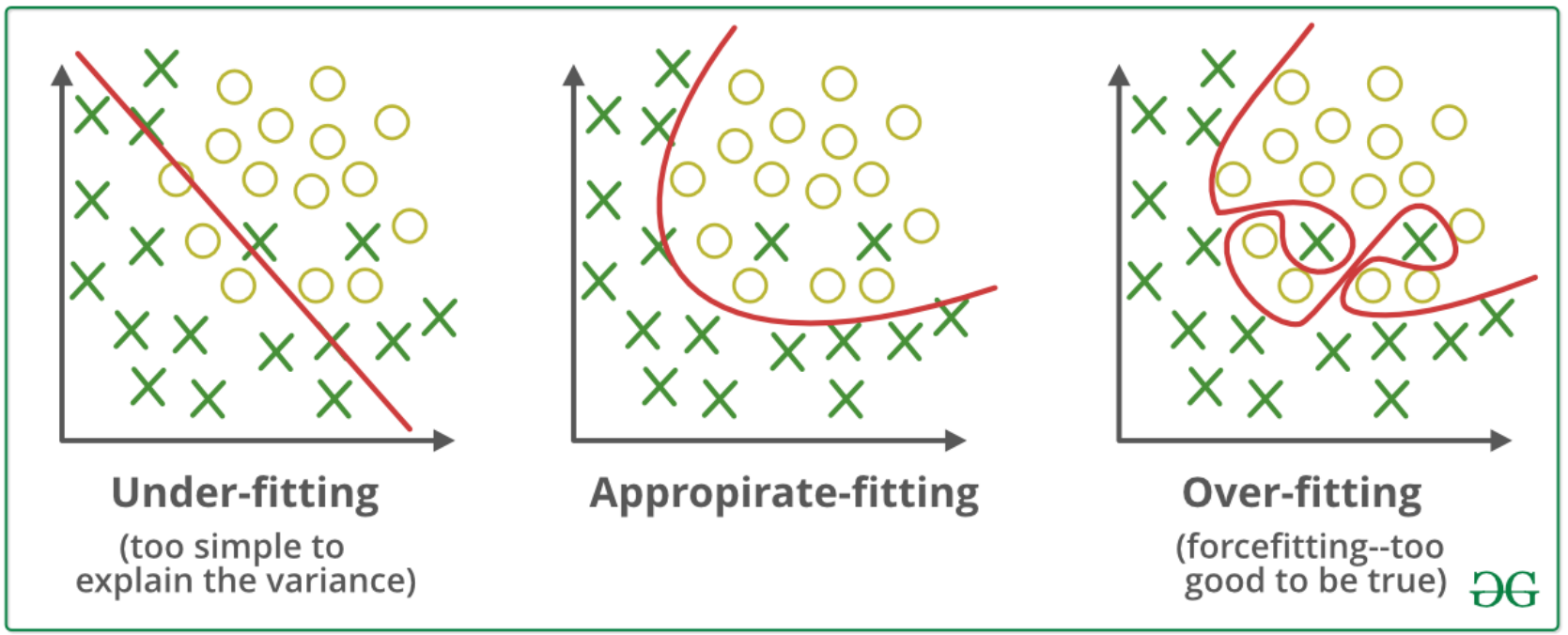
# Model Capacity Spectrum

- **Under-fit (high bias)**: model too simple, misses trend.
- **Good fit**: captures true signal, generalizes well.
- **Over-fit (high variance)**: model too complex, memorizes noise.



# Model Capacity Spectrum

- **Under-fit (high bias)**: model too simple, misses trend.
- **Good fit**: captures true signal, generalizes well.
- **Over-fit (high variance)**: model too complex, memorizes noise.



# Avoiding Overfitting: Lasso and Ridge Regularisation

**Lasso (L1)**

- Loss: $L = \frac{1}{N}\sum_i^N(\hat{y}_i - y_i)^2 + \lambda \sum_i^N |\theta_i|$.
- Lasso encourages sparsity by bringing some coefficients to exactly zero.
- Can set some weights *exactly* zero $\Rightarrow$ implicit feature selection.

**Ridge (L2)**

- Loss: $L = \frac{1}{N}\sum_i^N(\hat{y}_i - y_i)^2 + \lambda \sum_i^N \theta_i^2$
- Shrinks all weights smoothly toward zero.
- Encourages the model to have smaller and more balanced weights.

**Elastic net (L1+L2)**

- Loss: $L = \frac{1}{N}\sum_i^N(\hat{y}_i - y_i)^2 + \lambda \sum_i^N |\theta_i| + \lambda \sum_i^N \theta_i^2$
- Combines both lasso and ridge regularization.

Need to optimize additional hyperparameter $\lambda$.

# One Slide Summary

- Core **vocabulary**: dataset, feature, label, model, loss
- ML **paradigms**: supervised, unsupervised, reinforcement
- Regression *vs.* classification; MSE loss; linear-regression baseline
- Model **evaluation**: train/val/test, metrics, residuals
- Diagnose **under-/over-fit**; Ridge & Lasso for regularization