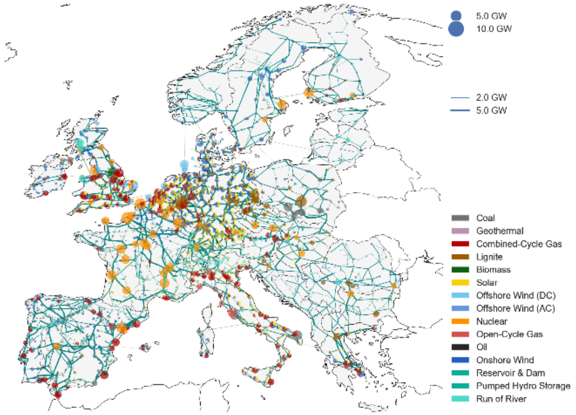
Energy System Modeling with Python



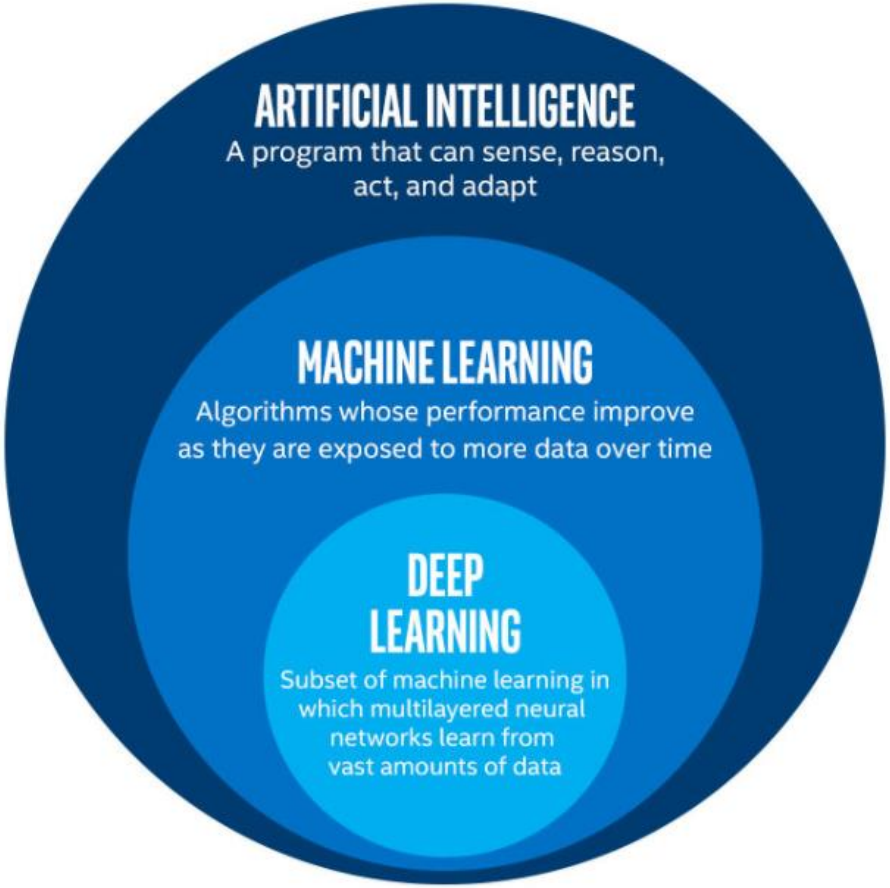
University of Freiburg (Germany) | Faculty of Engineering
Department of Sustainable Systems Engineering | INATECH
Chair for Control and Integration of Grids

Tuesday, 3. June 2025

Introduction

What is AI and ML?

- AI refers to the broader field of creating machines or software that can perform tasks that typically require human intelligence
- AI aims to mimic human cognitive functions in a way that allows machines to make autonomous decisions and perform tasks without explicit programming
- ML is a subfield of AI, which improves its performance on a task through experience (data)



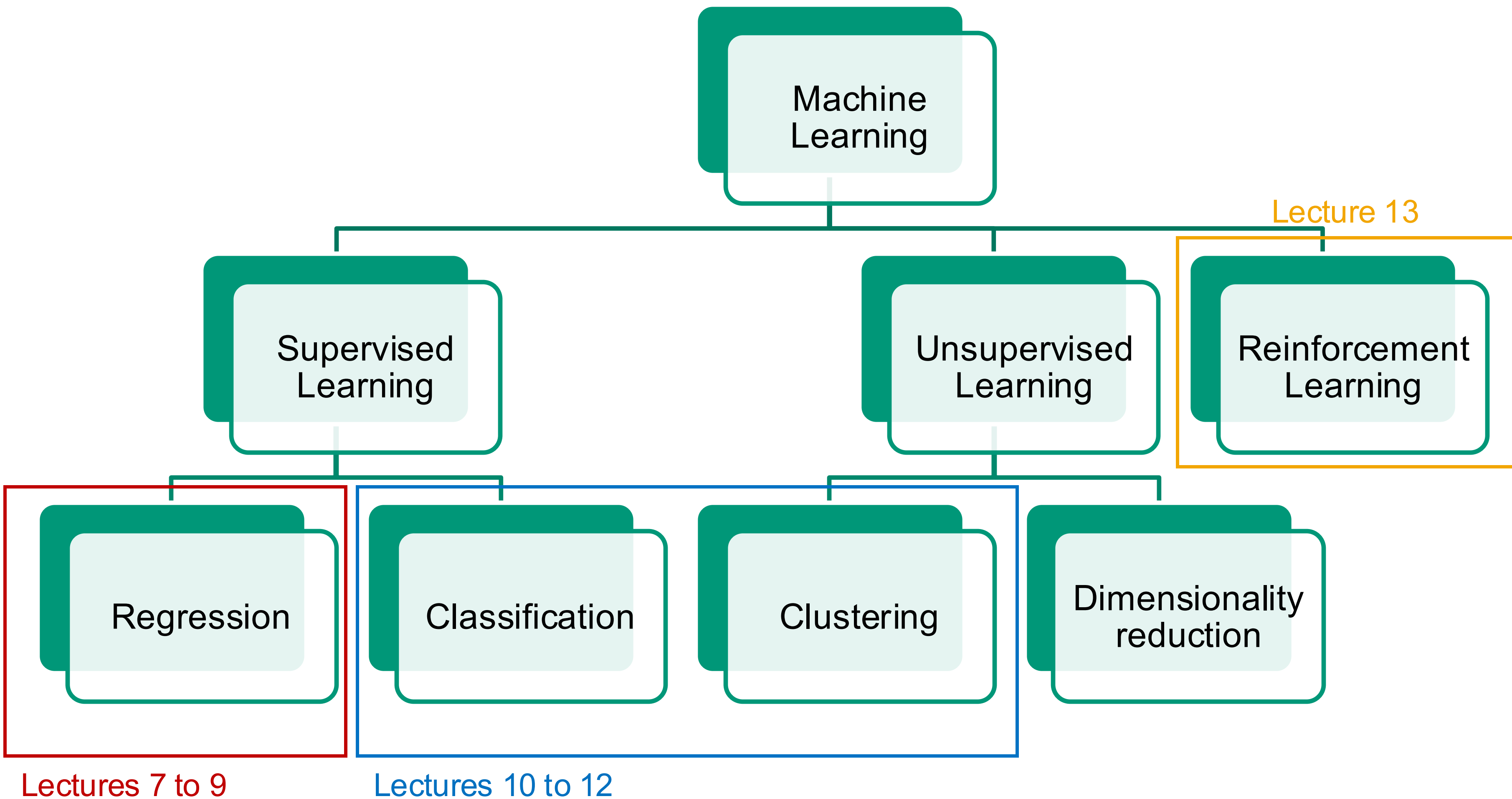
Let's define Machine Learning

Tom Mitchell (1998):

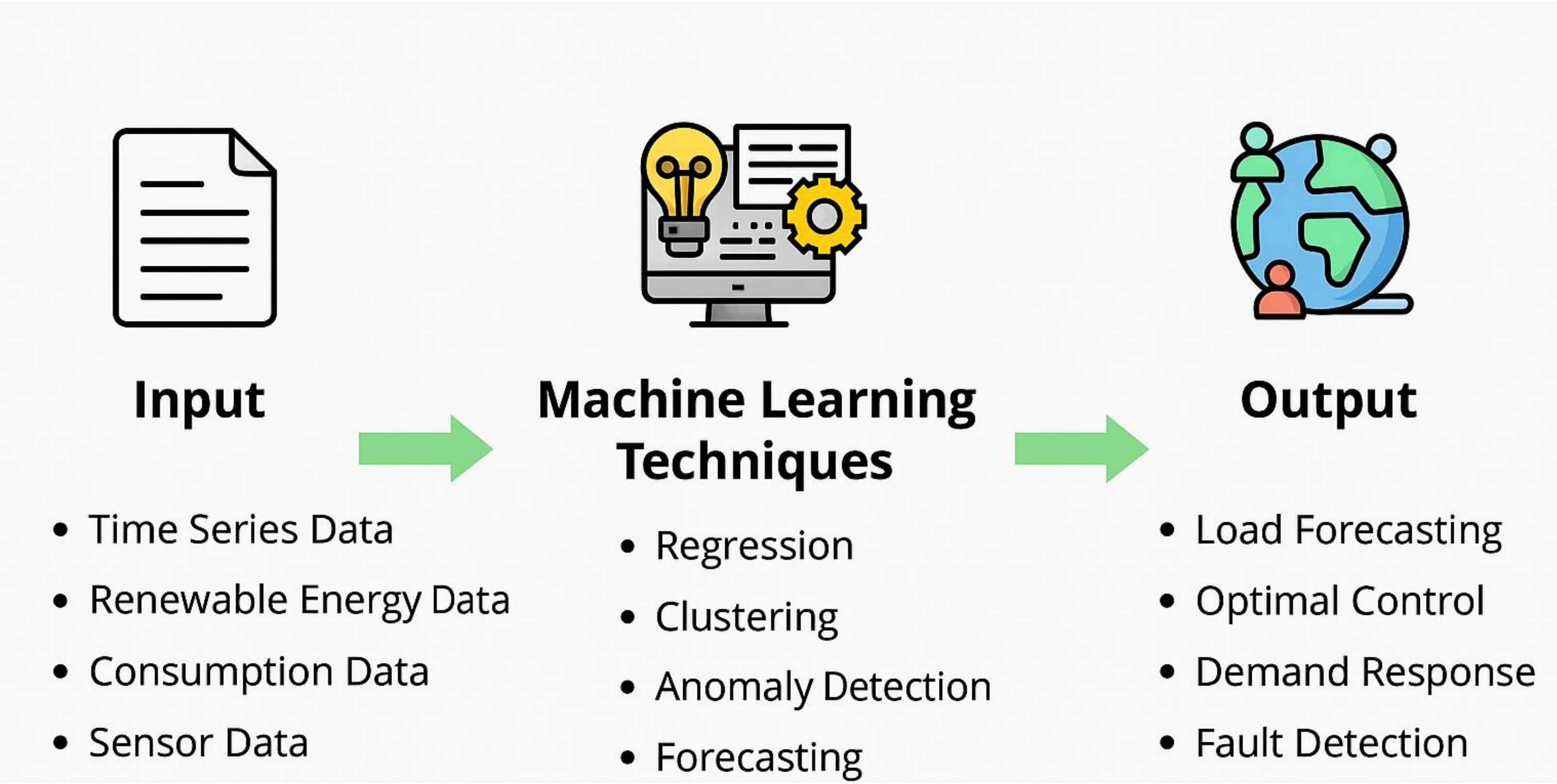
Machine learning is a well-posed learning problem where a computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

So, what is Experience E, Task T, and Performance measure P?

Branches of ML



What is the process of ML?



So, what is Experience E, Task T, and Performance measure P?

How can ML help us in the field of energy systems?

Energy Systems

- Complex, nonlinear, and coupled dynamics
- High uncertainty from renewables and demand
- Incomplete or limited observational data
- Intractable to simulate all operating conditions
- Need for real-time control and decision-making

ML Capabilities

- Learns complex behaviors without explicit system knowledge
- Adapts to variability and uncertainty
- Generalizes well from sparse or noisy data
- Approximates outcomes quickly without full models
- Enables fast, data-driven decisions in real time

Limitations of ML in Energy Systems

Energy Systems

- Energy systems are critical infrastructure
- Decisions must be explainable and auditable
- Operations must be reliable under all scenarios
- Failure can have serious physical and societal consequences
- Subject to strict regulations and accountability

ML Capabilities

- ML models must be trustworthy, safe, and robust
- Many ML models, especially deep learning, are black boxes
- ML cannot guarantee behavior in rare or unseen cases
- ML errors may be hard to detect and correct in real time
- The EU AI Act classifies energy systems as high-risk applications

Before applying ML, **carefully assess whether it's the right tool for the job** — interpretability, robustness, and compliance must come first.

Some terminology to remember

Data and Learning Setup

- **Features:** Input variables used by the model to make predictions.
- **Labels:** True output values used in supervised learning.
- **Model:** A function that maps inputs (features) to predicted outputs.
- **Training:** Process where the model learns from data by minimizing error

Model Evaluation

- **Testing / Evaluation:** Applying the trained model to unseen data to assess its generalization performance.
- **Performance Metrics:** Quantitative measures of model quality, such as:
    - **Accuracy** (classification)
    - **Precision / Recall / F1-score** (imbalanced classification)
    - **RMSE / MAE** (regression)
- **Loss Function:** A function that quantifies the difference between predicted and true values during training (e.g., Mean Squared Error, Cross-Entropy). The model learns by minimizing this.

Supervised Learning

Regression

How supervised learning works

Terminology:

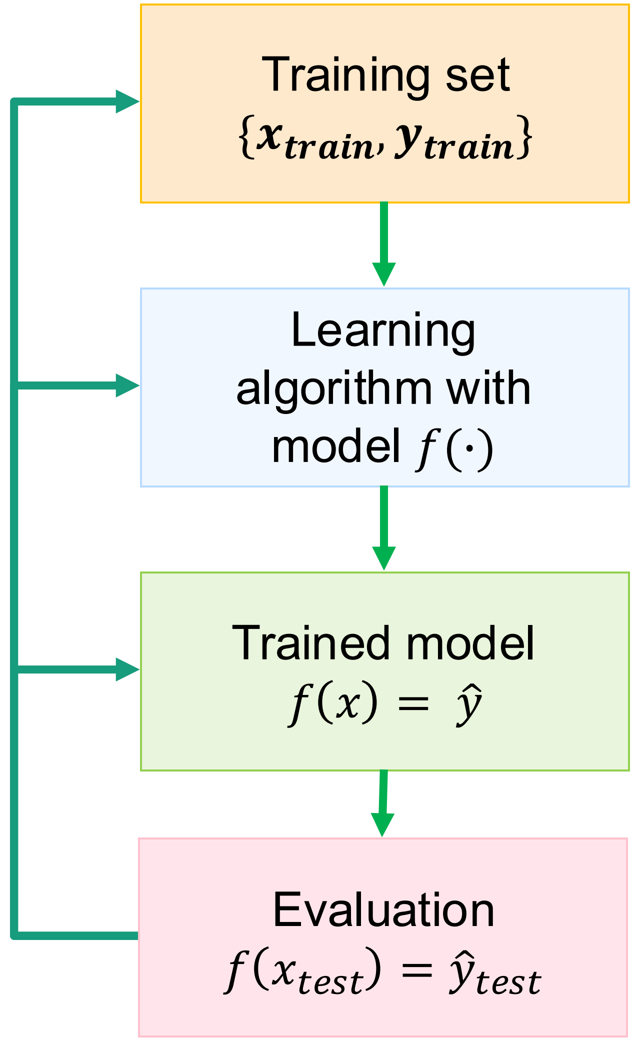
X: features vector, where $x \in X$

Y: labels vector, where y $\in Y$

x_{train}, y_{train} : training dataset

x_{test}, y_{test} : testing dataset that the model has never seen before

$f(\cdot)$: a model used to approximate the values



Supervised learning in simplest form: linear regression

The most basic equation to describe our model is known as
linear regression equation:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

where:

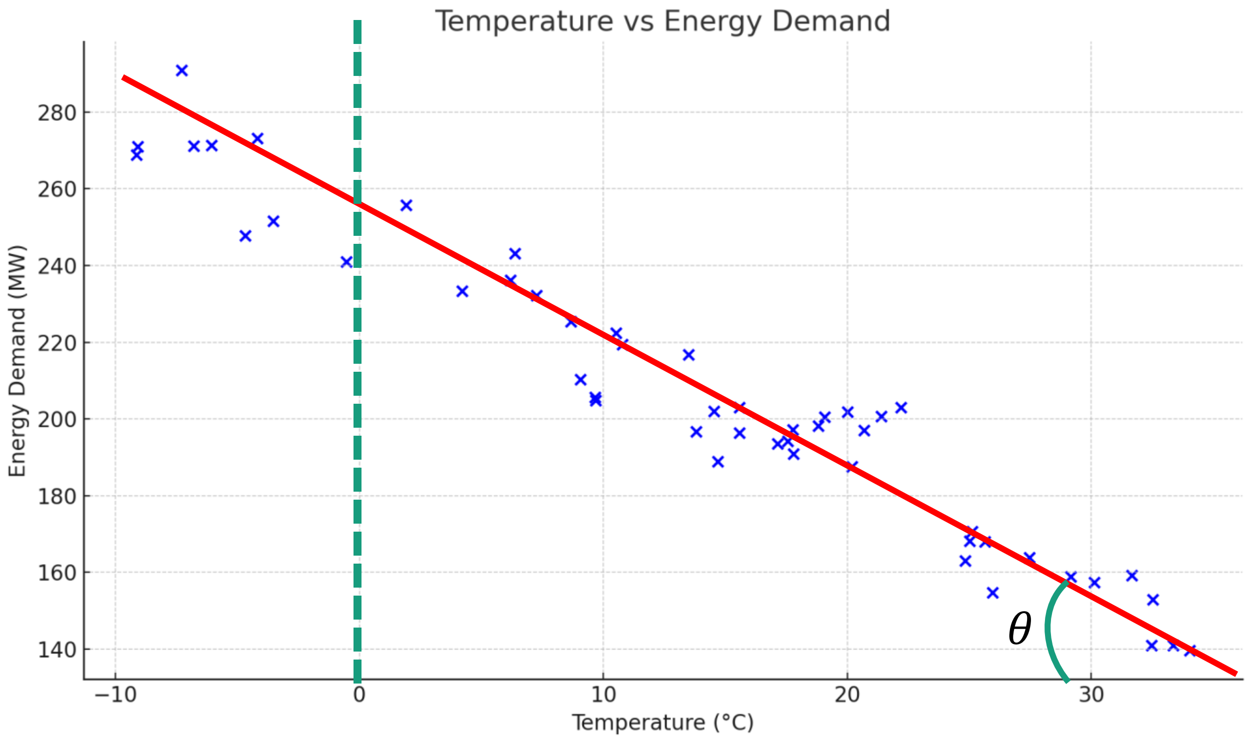
$\hat{y}$: predicted **output** (e.g., energy demand),

$x_1, x_2, \dots, x_n$: **input features** (e.g., temperature, wind speed, etc.),

β_0: **bias term** (intercept),

$\beta_1, \beta_2, \dots, \beta_n$: **coefficients** corresponding to each input feature.

Supervised learning in simplest form: linear regression



$$\hat{y} = \beta_0 + \beta_1 x_1$$

$$\beta_0 = y \text{ when } x = 0$$

$$\beta_1 = tan(\theta)$$

How to train the model: finding the parameters

Steps:

1) Define a model: $\hat{y} = \beta_0 + \beta_1 x_1$

2) Define a loss function as ***Squared Error:***

$$\mathcal{L}^{(i)} = \left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \left(y^{(i)} - \left(\beta_0 + \beta_1 x^{(i)}\right)\right)^2$$

3) Define a cost function as ***Mean Squared Error (MSE):***

$$J(\beta_0, \beta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \left(\beta_0 + \beta_1 x^{(i)}\right)\right)^2$$

4) Solve the equation:

- **Analytically**: Take derivatives of J with respect to β_0 and β_1, set them to zero, and solve the resulting equations,
- **Numerically**: Use an iterative method like **gradient descent** to minimize the cost

Minimizing the Cost — Gradient Descent

We want to minimize the *Mean Squared Error (MSE):*

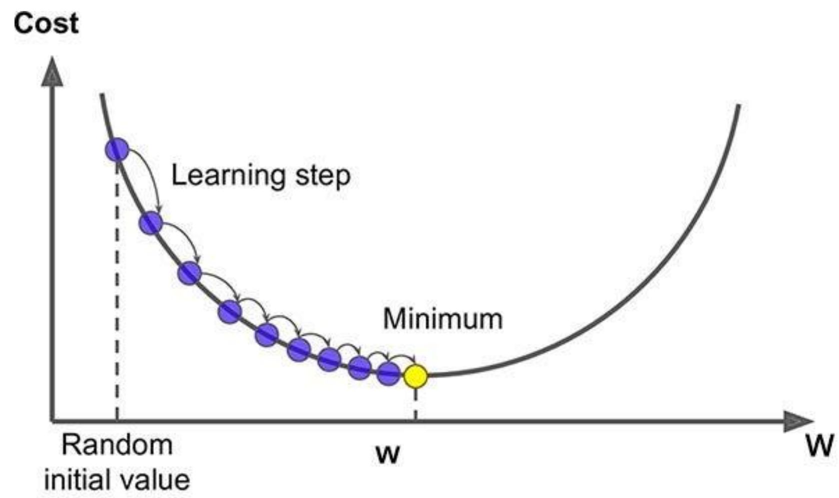
$$J(\beta_0, \beta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \left(\beta_0 + \beta_1 x^{(i)}\right)\right)^2$$

We will apply Gradient Descent: Iteratively update parameters

$$\beta_j \leftarrow \beta_j - \alpha \cdot \frac{\partial J}{\partial \beta_j}$$

where:
α: is the ***learning rate*** (how fast we update our parameters)

Gradient descent is an iterative optimization algorithm that adjusts the model parameters step by step in the direction that **reduces the error** the most. It uses the **slope (gradient)** of the cost function to decide how to update the parameters, gradually moving toward the minimum.



INATECH UNI FREIBURG

Minimizing the Cost — Gradient Descent

Compute the partial derivatives:

$$\frac{\partial J}{\partial \beta_0} = -\frac{2}{m}\sum_{i=1}^{m}\left(y^{(i)} - \left(\beta_0 + \beta_1 x^{(i)}\right)\right)$$

$$\frac{\partial J}{\partial \beta_1} = -\frac{2}{m}\sum_{i=1}^{m}\left(y^{(i)} - \left(\beta_0 + \beta_1 x^{(i)}\right)\right) \cdot x^{(i)}$$

Gradient descent update rules:

$$\beta_j \leftarrow \beta_j - \alpha \cdot \frac{\partial J}{\partial \beta_j}$$

$$\beta_0 \leftarrow \beta_0 + \frac{2\alpha}{m}\sum_{i=1}^{m}\left(y^{(i)} - \left(\beta_0 + \beta_1 x^{(i)}\right)\right)$$

$$\beta_1 \leftarrow \beta_1 + \frac{2\alpha}{m}\sum_{i=1}^{m}\left(y^{(i)} - \left(\beta_0 + \beta_1 x^{(i)}\right)\right) \cdot x^{(i)}$$

How to evaluate the performance

Mean Absolute Error (MAE)

Definition:
MAE measures the **average magnitude of the errors**, ignoring their direction.

Formula:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y^{(i)} - \hat{y}^{(i)}\right|$$

Key Points:

- Measures **average absolute deviation**
- **Linear penalty** for all errors
- Same units as the target (e.g., €/MWh)

Root Mean Squared Error (RMSE)

Definition:
RMSE is the square root of the **mean of squared errors**.

Formula:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}$$

Key Points:

- **Quadratic penalty** — large errors weigh more
- Always ≥ MAE
- Same units as the target (e.g., €/MWh)

Rule of thumb:
MAE for simple, balanced accuracy;
RMSE when large errors must be minimized.

Coffee Break

INATECH

UNI FREIBURG

Time to put everything into code

What You'll Do in Code Today

1. **Implement Linear Regression from Scratch**
   Understand how a model learns the best-fitting line using gradient descent
2. **Visualize the Learning Process**
   Plot how the model gradually improves over time — see the cost drop and the line converge
3. **Apply Regression to Real Energy Data**
   Use real-world demand and price data from the German day-ahead market
4. **Compare Your Model to Scikit-learn**
   Benchmark your results against a standard ML library
5. **Evaluate Forecast Quality**
   Calculate error metrics (MAE, RMSE) to assess how good your predictions are
6. **Multiple linear regression**
   You will implement multiple regression with more features of your choice

Takeaways

✓ You now understand what **supervised learning** is — and how it applies to energy systems.

✓ You've seen how a **linear regression model** can forecast energy demand or prices from input features.

✓ You implemented **gradient descent** — the key algorithm for training many ML models.

✓ You explored how **learning rate and normalization** affect model convergence.

➢ How do MAE and RMSE differ — and when would you use each?

➢ How does gradient descent *know* when it's done learning?

➢ In which real-world energy tasks might **simple models** like linear regression already be "good enough"?

➢ And where might they fail — what can't they capture?