

1. General context
2. Supervised vs. unsupervised learning
3. Machine learning in Physical Chemistry
4. Python vs. other tools – at the era of generative AIs
5. Typical workflow – including xAI
6. Conclusion

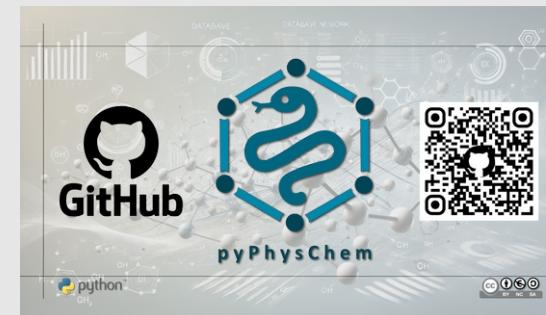
Part of this lecture will rely on live sessions using the data and **Jupyter notebooks** available in this GitHub repository



Python in the  
Physical Chemistry Lab  
[ pyPhysChem ]



<https://github.com/rpoteau/pyPhysChem>



# General context

## Artificial Intelligence (AI)

*intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals*

### Goals

reasoning & (basic) problem solving

knowledge representation

planning: making choices and hierarchy of events

learning (*i.e.* machine learning)

natural language processing

perception of the world from sensors

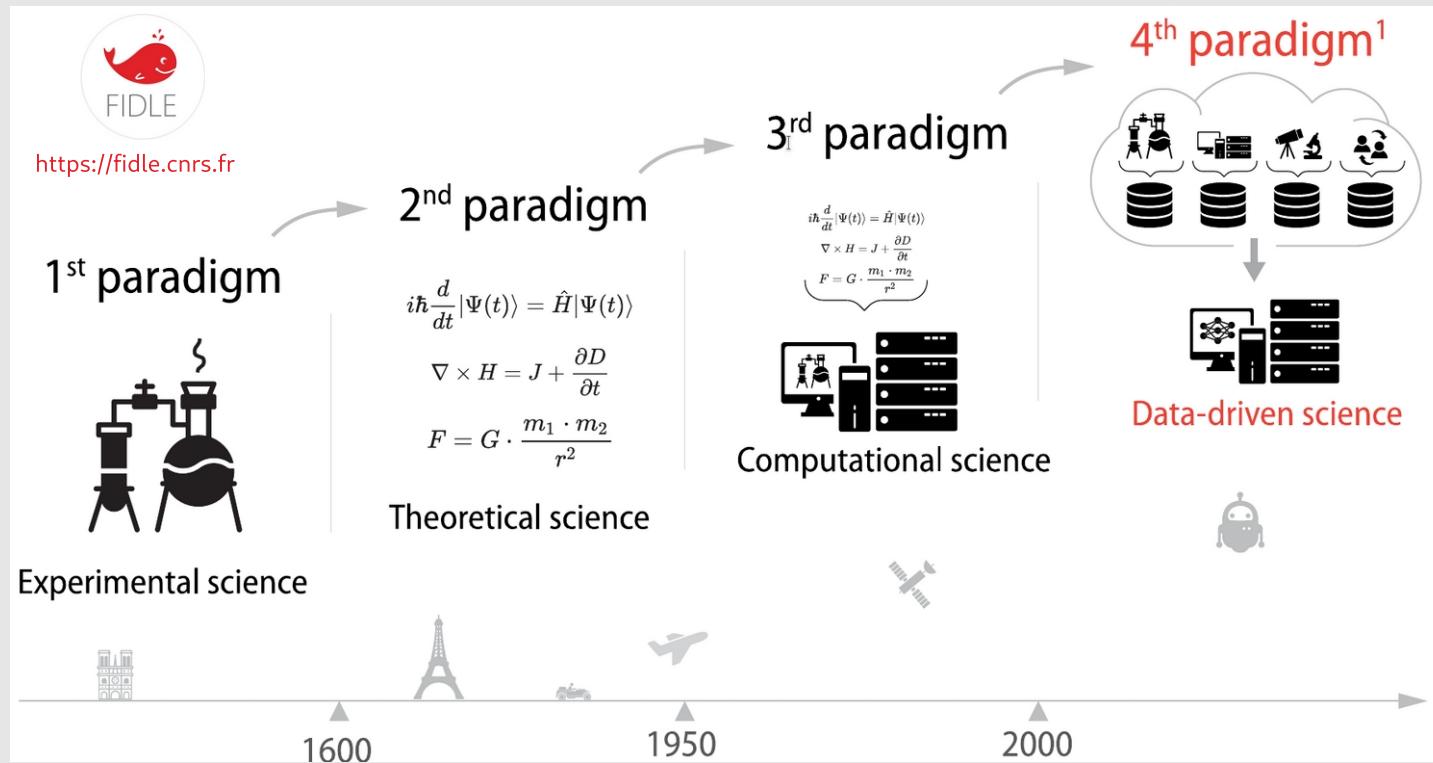
ability to move and manipulate objects

simulating human affects

long-term goal: ability to solve an arbitrary problem

**N.B. in some fields "artificial intelligence" means "machine learning with neural networks"**

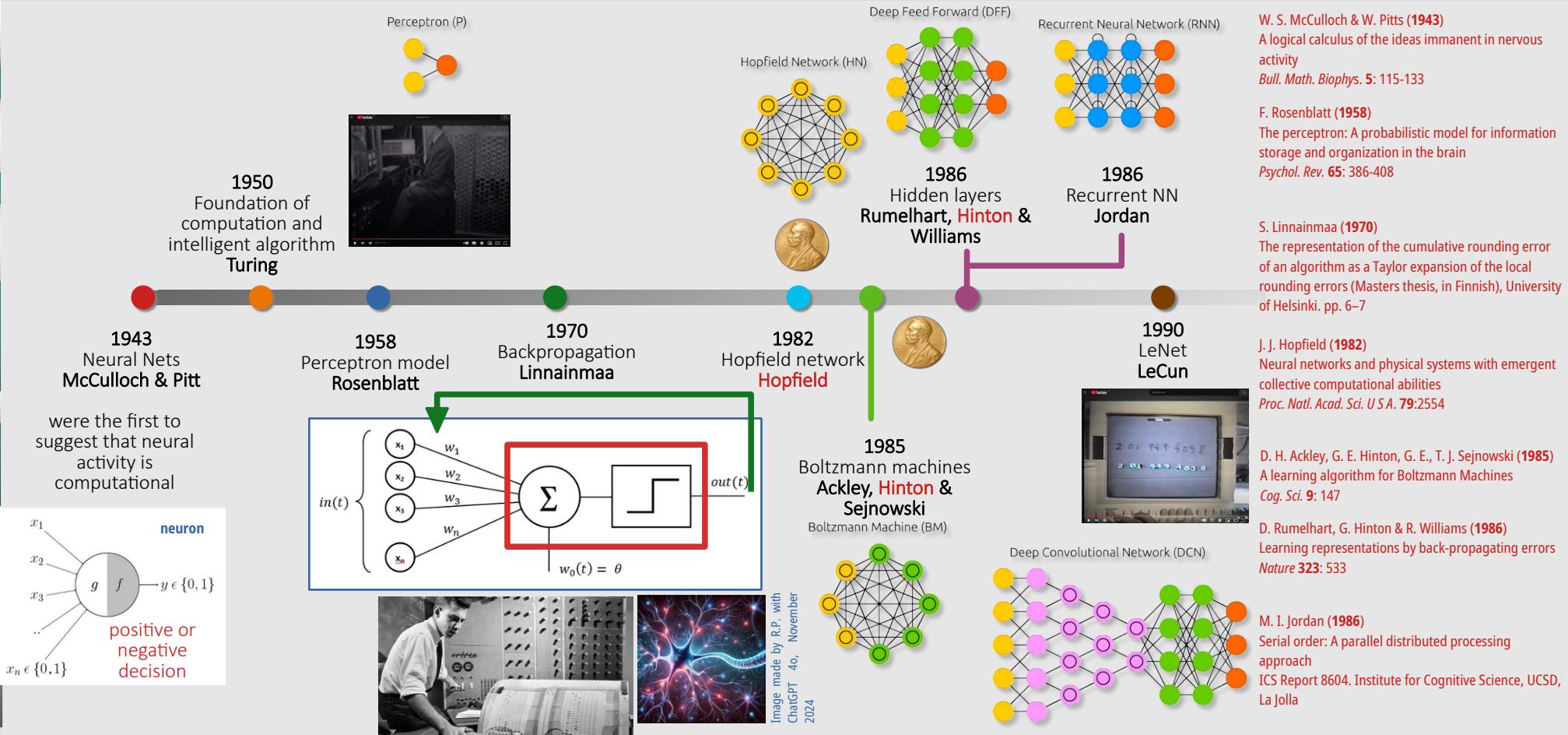
## Artificial Intelligence and Scientific Research



# Outlook: data science and machine learning

## Artificial Intelligence and Artificial Neural Networks (ANN) timeline

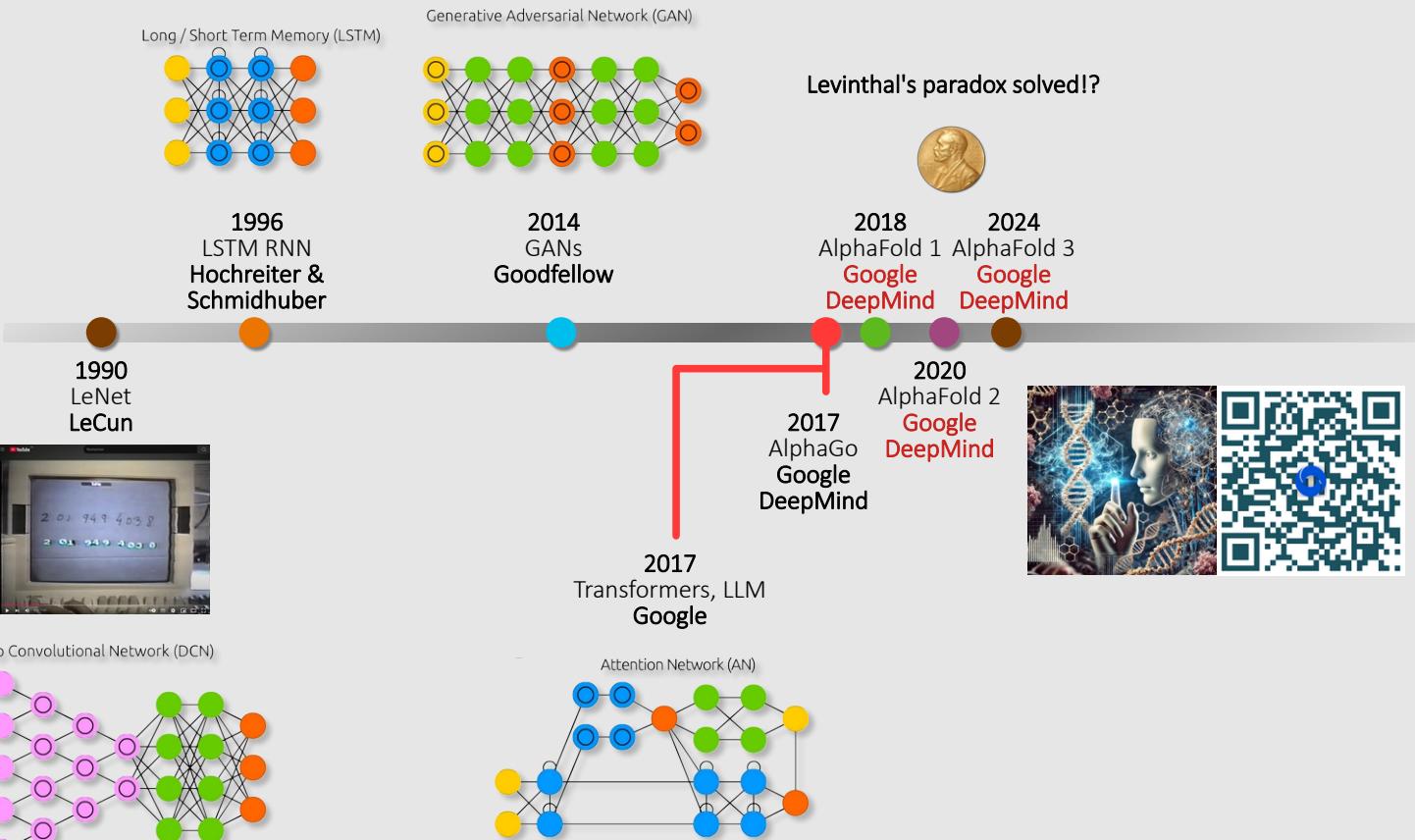
Version: Sunday, December 7, 2025



# Outlook: data science and machine learning

## Artificial Intelligence and Artificial Neural Networks (ANN) timeline

Version: Sunday, December 7, 2025



S. Hochreiter & J. Schmidhuber (1996)  
LSTM can solve hard long time lag problems  
Advances in Neural Information Processing Systems 9:  
Proceedings of The 1996 Conference, MIT Press 1997,  
p. 473

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio (2014)  
Generative Adversarial Nets. Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014), p. 2672

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin (2017)  
Attention is All you Need  
in Advances in Neural Information Processing Systems. 30. Curran Associates, Inc.

J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, S. W. Bodenstein, D. A. Evans, C.-C. Hung, M. O'Neill, D. Reiman, K. Tunyasuvunakool, Z. Wu, A. Žemgulytė, E. Arvaniti, C. Beattie, O. Bertoli, A. Bridgland, A. Cherepanov, M. Congreve, A. I. Cowen-Rivers, A. Cowie, M. Figurnov, F. B. Fuchs, H. Gladman, R. Jain, Y. A. Khan, C. M. R. Low, K. Perlin, A. Potapenko, P. Savy, S. Singh, A. Stecula, A. Thillaisundaram, C. Tong, S. Yakneen, E. D. Zhong, M. Zielinski, A. Žídek, V. Bapst, P. Kohli, M. Jaderberg, D. Hassabis & J. M. Jumper (2024)  
Accurate structure prediction of biomolecular interactions with AlphaFold 3  
Nature 630: 493

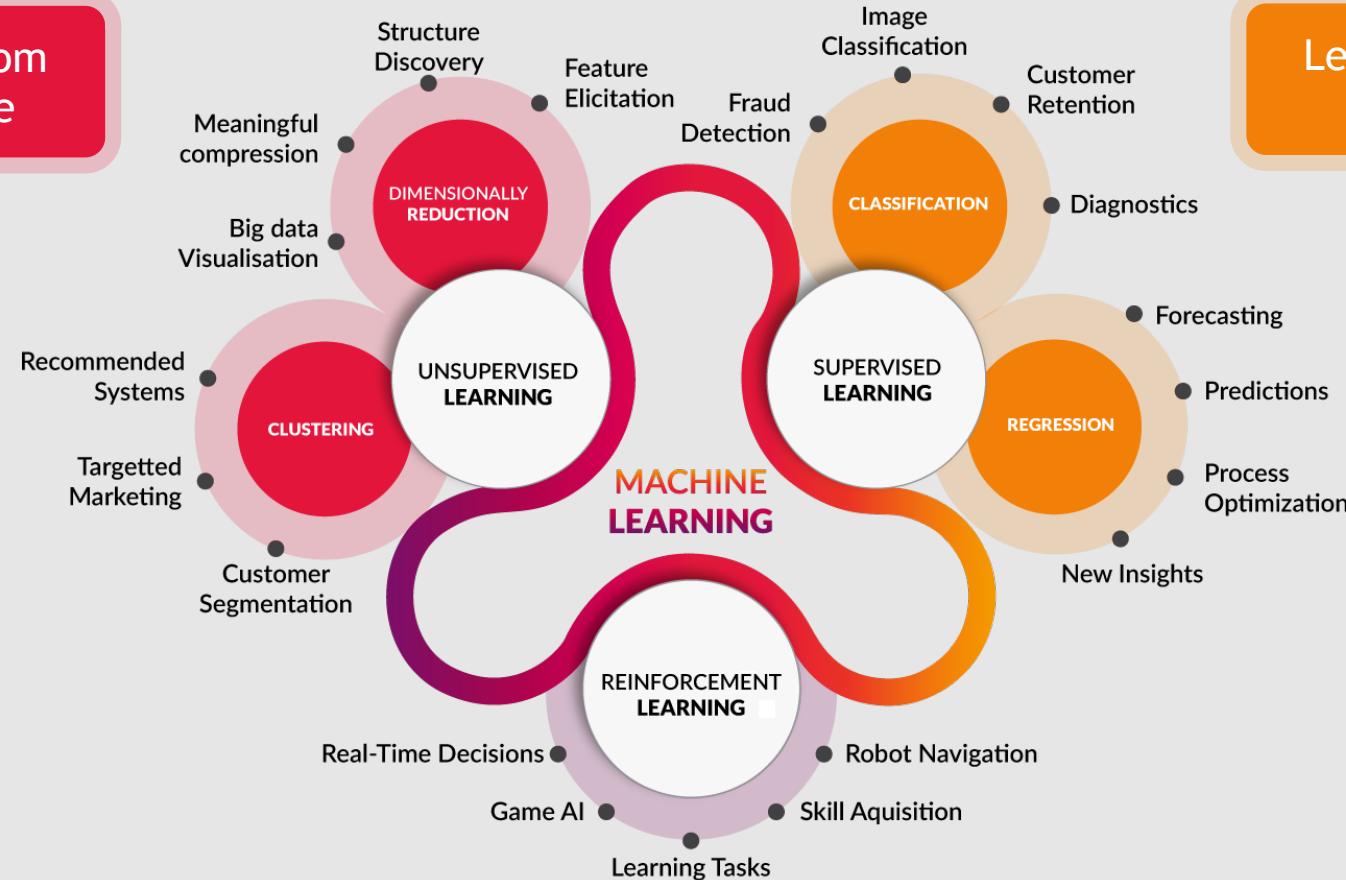
# Machine Learning

# Machine learning

Version: Sunday, December 7, 2025

Learning from  
data alone

Learning from  
examples



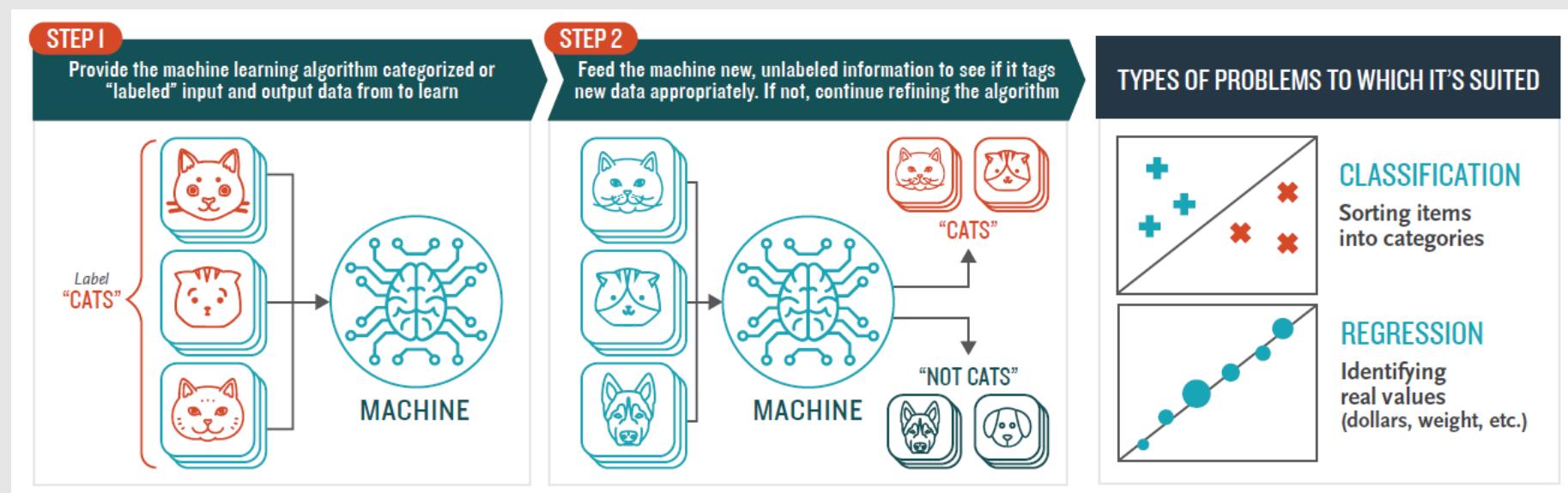
## Supervised learning

Data are provided along with the desired output (*i.e.* labelled data)

### Example of cats detection:

- collect thousands of images of cats
- draw a bounding box around each cat
- feed the entire dataset to the machine so it can learn all by itself

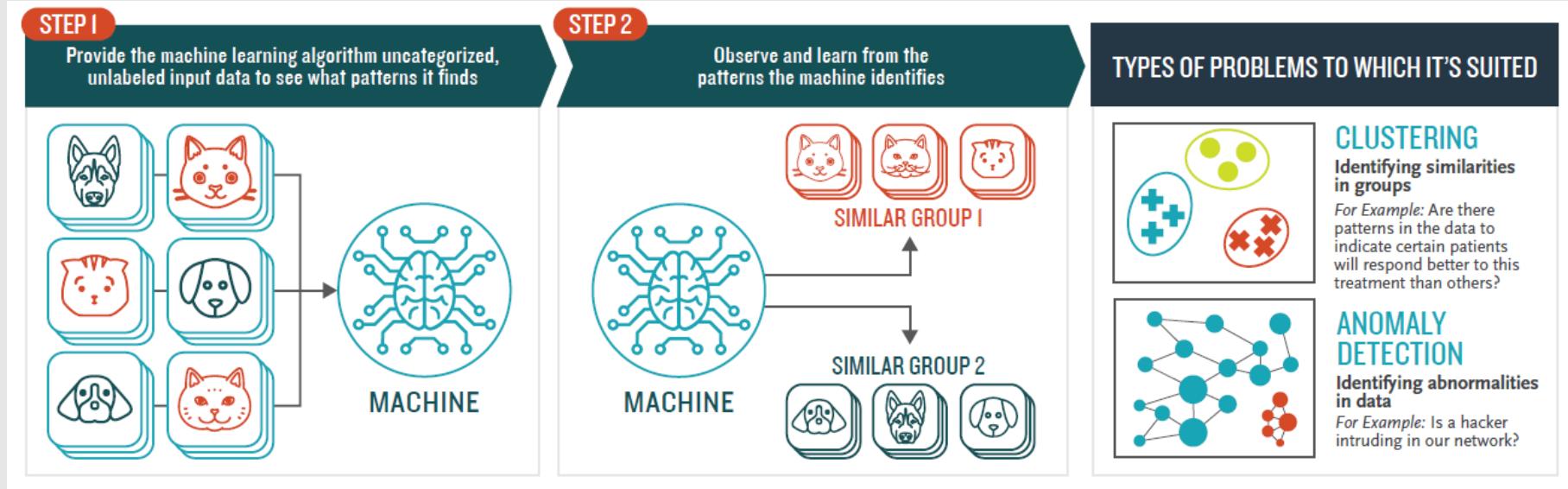
Learning from examples



## Unsupervised learning

Learning from data alone

- Just provide data
- Let the machine find out (or cluster) the patterns in the dataset



# Machine Learning in Physical Chemistry

# Transformative role of AI in sciences of matter research



Image made by R.P. with ChatGPT 4o, November 2024

Version: Sunday, December 7, 2025

## Acceleration of Material Discovery

- Property prediction prior to laboratory synthesis
- Exploration of a broad parameter space
- Inverse design of materials
- Catalyst design

## Understanding mechanisms and multi-factorial processes through explainable AI (XAI)

## Advanced Modeling

- Predictions comparable to *ab initio* calculations
- Simulations of physical and chemical processes at challenging scales

## Design of Experiments

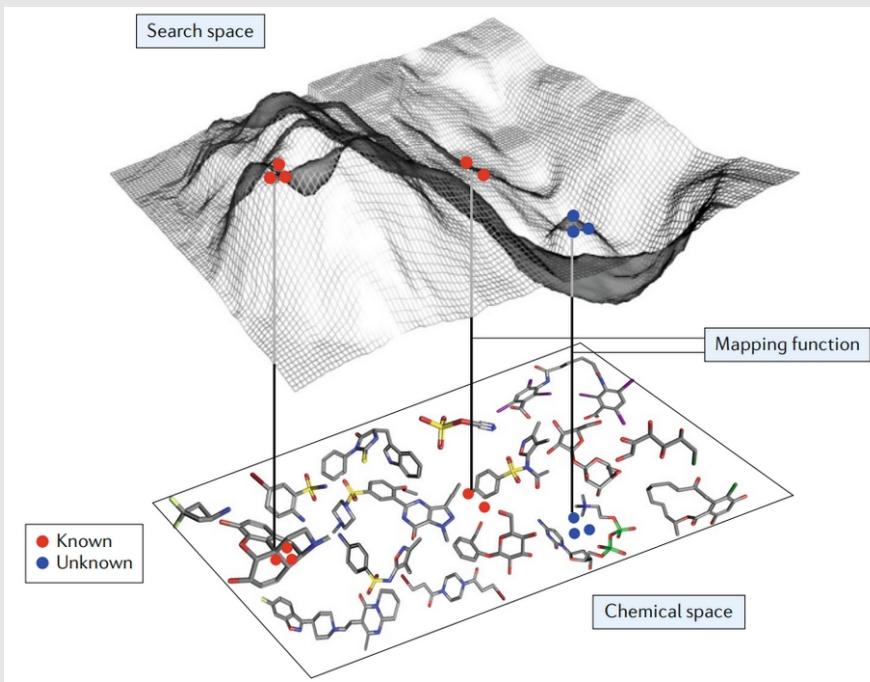
- Automated processing with robotic systems guided by AI and real-time analysis of experimental data
- Optimizing experimental setups through Design of Experiments (DoE)

## Data Exploration

- Detecting trends and anomalies using unsupervised methods
- Derive insights from diverse datasets like crystal structures or spectroscopy

The chemical space is unimaginably vast

~  $10^{60}$  drug-like molecules



Experimental and quantum chemistry methods:  
accurate but slow or costly

Most of the chemical space remains unexplored

## The plan

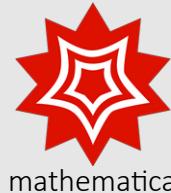
- to exploit available data (experimental or DFT, small or large)
- to train data-driven models using ML
- and use explainability tools to open the black box of predictions



# Development of AI tools with python

# How to develop home-made ML tools?

Version: Sunday, December 7, 2025



The ML galaxy within python™ is an exceptional ecosystem

Python is a high-level, interpreted, object-oriented, general-purpose programming language



<https://scikit-learn.org/>



<https://pytorch.org/>



<https://keras.io/>



<https://www.tensorflow.org/>



NumPy



pymatgen

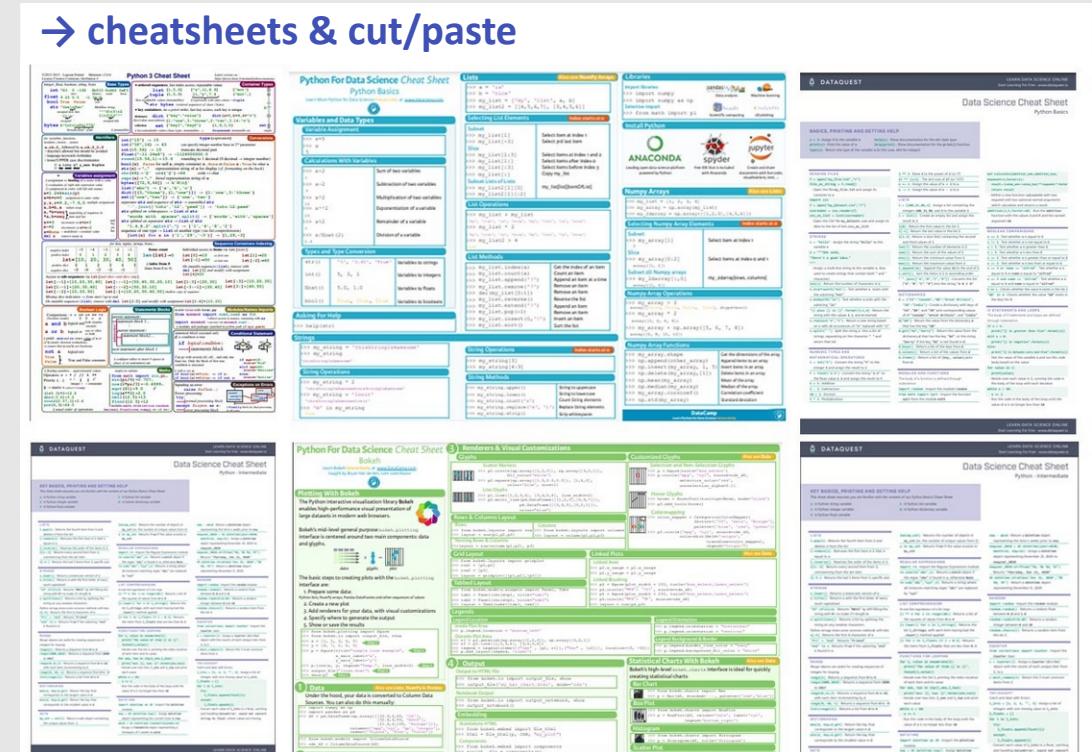


## Using Python as an everyday tool for scientific calculation and computation (it can even replace excel... by far)

- basic knowledge of programming languages (variables, arrays, loops, conditional tests...)
- enthusiastic and vast community



→ cheatsheets & cut/paste



The image displays four separate screenshots of Python cheat sheets from Dataquest, arranged in a 2x2 grid. Each screenshot is a dense, multi-colored reference guide containing numerous code snippets, tables, and explanatory text. The top-left sheet is titled 'Python 3 Cheat Sheet' and covers basic operations like arithmetic, comparison, and logical operators. The top-right sheet is titled 'Python For Data Science Cheat Sheet - Python Basics' and includes sections on variables and data types, array operations, and data structures. The bottom-left sheet is titled 'DATAQUEST Data Science Cheat Sheet - Beginner' and focuses on data manipulation with pandas and numpy. The bottom-right sheet is titled 'DATAQUEST Data Science Cheat Sheet - Intermediate' and covers more advanced topics like machine learning and data visualization with Matplotlib and Seaborn.

Uneasy?

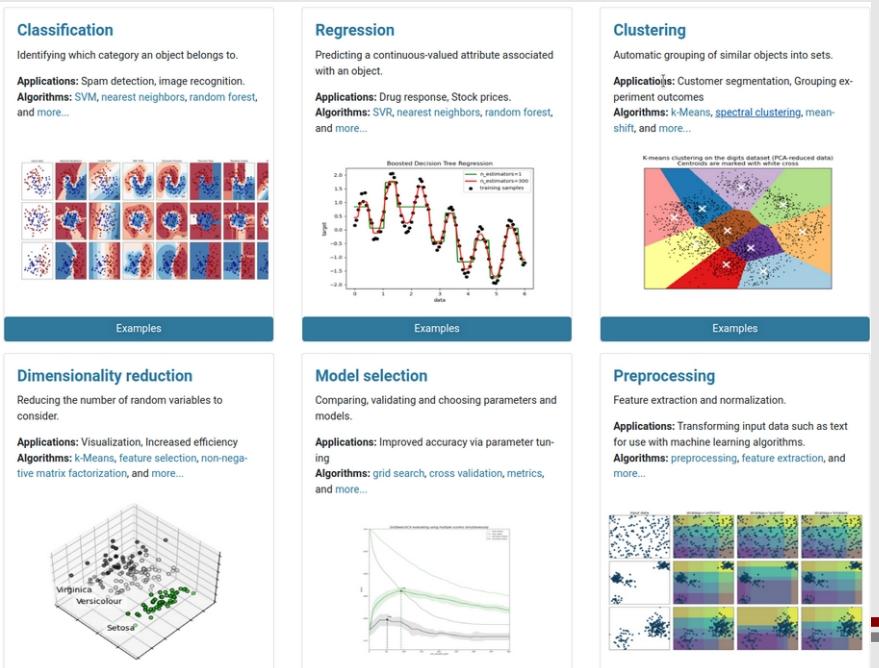
- yes and no
- important initial investment
- worth the effort



Machine learning in Python with scikit-learn

**Simple and efficient tools for predictive data analysis**  
**Accessible to everybody, and reusable in various contexts**  
**Built on NumPy, SciPy, and matplotlib**

INRIA took leadership of the project and made the first public release on February 2010  
3-clause BSD License (permissive free software license, compatible with the GNU GPL)



**Classification**  
Identifying which category an object belongs to.  
**Applications:** Spam detection, image recognition.  
**Algorithms:** SVM, nearest neighbors, random forest, and more...

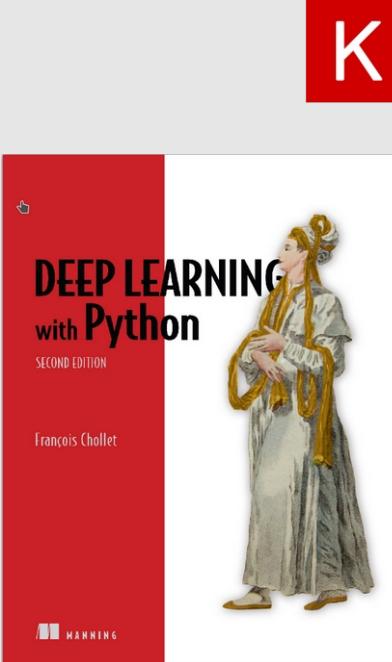
**Regression**  
Predicting a continuous-valued attribute associated with an object.  
**Applications:** Drug response, Stock prices.  
**Algorithms:** SVR, nearest neighbors, random forest, and more...

**Clustering**  
Automatic grouping of similar objects into sets.  
**Applications:** Customer segmentation, Grouping experiment outcomes  
**Algorithms:** k-Means, spectral clustering, mean-shift, and more...

**Dimensionality reduction**  
Reducing the number of random variables to consider.  
**Applications:** Visualization, Increased efficiency  
**Algorithms:** k-Means, feature selection, non-negative matrix factorization, and more...

**Model selection**  
Comparing, validating and choosing parameters and models.  
**Applications:** Improved accuracy via parameter tuning  
**Algorithms:** grid search, cross validation, metrics, and more...

**Preprocessing**  
Feature extraction and normalization.  
**Applications:** Transforming input data such as text for use with machine learning algorithms.  
**Algorithms:** preprocessing, feature extraction, and more...



**High Level Deep Learning Application Programming Interface (API)**

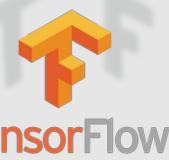
By François Chollet (Google)

Part on TensorFlow since 2017

MIT license (permissive free software license)

**how to start?**

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
```



<https://www.tensorflow.org/>

**Google Brain's second-generation system**

Supported by Google  
Low level API

Apache license (yet another permissive free software license)



<https://pytorch.org/>

From Torch library

Supported by Facebook

BSD licence

(permissive free software license)

**An open source machine learning framework that accelerates the path  
from research prototyping to production deployment**

**Widely used in the field of AI research**



The screenshot shows the Anaconda website homepage. At the top, there's a navigation bar with links for Products, Pricing, Solutions, Resources, Partners, Blog, and Company. A "Contact Sales" button is also present. Below the navigation, it says "Individual Edition is now ANACONDA DISTRIBUTION". The main heading is "ANACONDA DISTRIBUTION" in large green letters. Below that, it says "The world's most popular open-source Python distribution platform". To the right, there's a download section for "Anaconda Distribution" with a "Download" button and a link for "For Linux". It also mentions "Python 3.9 • 64-Bit (x86) Installer • 659 MB". Below this, there's a "Get Additional Installers" section with icons for Windows, Mac, and Linux.



Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment



JupyterLab is a web-based interactive development environment for notebooks, code, and data. Users can arrange workflows in **data science**, **scientific computing**, and **machine learning**.

**you can save your everyday data manipulation / visualization as you do in your chemistry laboratory notebooks**

## Not familiar with python... yet?

### "Python in the Physical Chemistry Lab (PPCL)" in a nutshell

This Python computer lab assumes a very basic knowledge of a programming language and algorithm development.

To run the content of a Python cell: click on a cell to select it. Then press SHIFT+ENTER on your keyboard, or press the play button in the top left corner of this window



If you click on a text cell by accident, you will see the so-called markdown coding of this cell (it is closely related to the HTML language). The corresponding formatted text/images/tables will be rendered by running the cell (SHIFT+ENTER or play button).

Ready? Put down your mobile phone 📱, please, and let's enter into the Python realm. 🎉🎉

### PPCL.ipynb

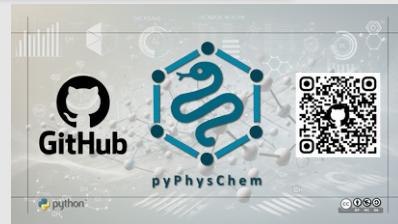
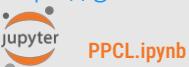
#### Simple calculations

#### Basic mathematical operations

```
# Every line that starts with a # character is a comment  
  
# addition  
3 + 2 # it is also possible to add a comment after a command  
  
#This is a new cell. It is possible to define several operations or commands in a cell  
# multiplication  
3*2  
  
#division  
7/2
```

github repository

<https://github.com/rpoteau/pyPhysChem>



README

   Université de Toulouse [ pyPhysChem ] 

Ce dépôt GitHub propose une collection de notebooks Jupyter conçus pour intégrer la programmation en Python dans l'enseignement de la chimie physique. Ces notebooks fournissent des exemples commentés et illustrés, couvrant des sujets tels que les dérivées et les intégrales, l'atome d'hydrogène et les représentations moléculaires. Ce dépôt inclut également des ressources pour des applications d'apprentissage automatique en chimie, comme les réseaux de neurones artificiels et les autoencoder. Pour utiliser ces outils, les utilisateurs sont invités à installer Jupyter ainsi qu'une distribution Python, Anaconda étant recommandée. Des instructions détaillées pour cloner le dépôt et exécuter les notebooks sont disponibles dans ce fichier README.md

This GitHub repository offers a collection of Jupyter Notebooks designed to integrate Python programming into physical chemistry education. These notebooks provide commented and illustrated examples, covering topics such as derivatives and integrals, the hydrogen atom, and molecular representations. The repository also includes resources for machine learning applications in chemistry, like artificial neural networks and autoencoders. To utilize these materials, users are advised to install Jupyter and a Python distribution, with Anaconda being a recommended option. Detailed instructions for cloning the repository and running the notebooks are provided in the present README.md document

Table des Matières

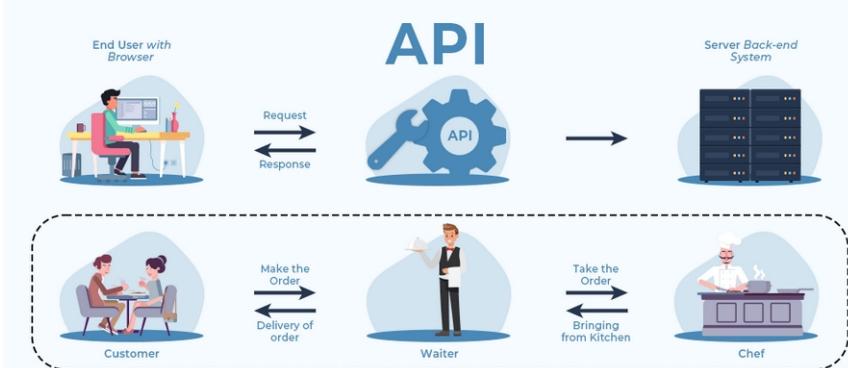
- Document principal et pré-requis
- Installation et activation d'une distribution python
  - Introduction
  - Installation de miniconda
  - Activation d'un environnement conda
  - Installation des bibliothèques Python et des outils additionnels nécessaire
- Clonage du dépôt (repository) pyPhysChem et installation des bibliothèques Python nécessaires
- Utiliser ces notebooks à l'aide de JupyterLab
- Liste des changements
- Comment citer ce travail ?

Table of Contents

- Main document and prerequisites
- Installation and activation of a Python distribution
  - Introduction
  - Installing miniconda

Most Python libraries expose powerful APIs that give access to advanced scientific tools

## Analogy: an API is like a restaurant menu



- You (the user) pick a dish → the request
- The kitchen (the server) cooks it → the processing
- The waiter (the API) brings it to your table → the response

You don't need to know how the kitchen works or how to "flamber" a "crème brûlée" - you just need to read the menu (the API documentation)

Of course, having a bit of cooking knowledge helps: if you don't know what "*confit*", "*al dente*", or "*umami*" means, your ordering experience may become... experimental. Same with APIs: a few basic programming, applied mathematics and [physical] chemistry notions prevent unexpected outcomes

## Why APIs matter (for students and researchers)

### 1. Don't reinvent the wheel

→ No need to re-implement algorithms or re-create datasets. Just use the API provided

### 2. Save time and reduce errors

→ APIs offer reliable, tested functionality

Example: solving a first- or second-order differential equation with `scipy (integrate.solve_ivp)` instead of coding your own numerical solver from scratch

### 3. Access complex resources

→ APIs connect you to huge databases (Materials Project, PubChem,...), remote servers, or specialised software (rdkit,...) that would be impossible to handle directly

### 4. Standardized communication

→ An API defines a clear, documented way to interact with a service. Everyone uses the same rules

### 5. Interoperability

→ APIs allow Python to interact with code in C, Fortran, Java, or with remote web services, without worrying about the underlying complexity

### 6. Rapid prototyping

→ APIs work like Lego bricks: you can quickly assemble advanced tools and test scientific ideas or proof-of-concepts with only "a few" lines of code

gAIs provide assistance to the development of simple or even advanced python codes

## Opportunities / Useful Applications

### Rapid prototyping of scientific workflows

Data parsing and mining, visualization, automation, small ML pipelines

### Assistance for complex coding tasks

DFT parsing, spectroscopy analysis, HPC workflows, error-handling...

### Automatic documentation & code explanations

Docstrings, comments, restructuring legacy scripts

### Debugging support

Identifying logical errors, improving robustness, optimizing performance

### Bridging multi-disciplinary knowledge

Linking chemistry concepts with algorithmic implementations

**gAIs can accelerate code development, but the scientist remains responsible for validation, interpretation, and critical checking and thinking**

## Risks / Limitations

### Hallucinations and subtle errors

Syntactically correct but chemically or physically wrong code

### Overtrust in outputs

Risk of using unverified code in production or scientific publications

### Data leakage / confidentiality concerns

(if code or sensitive data are pasted in public LLMs)

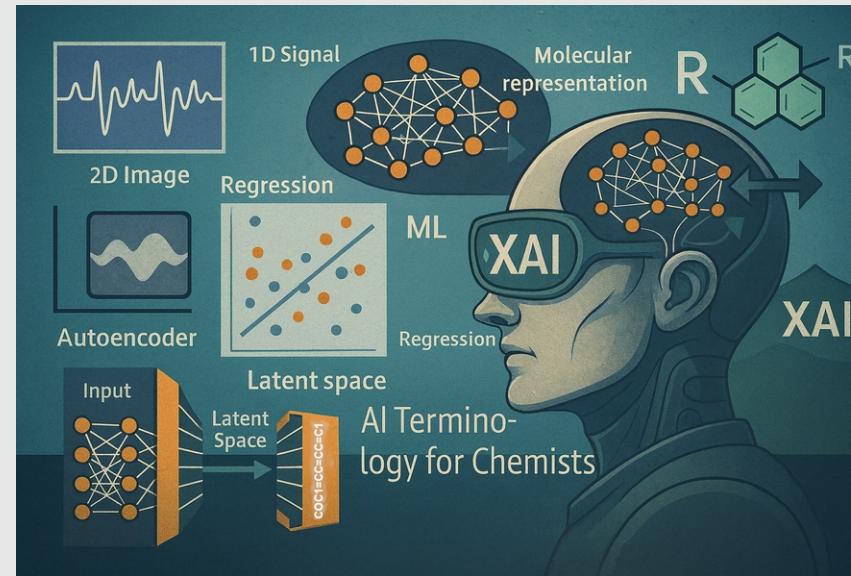
### Environmental impact

Training and inference of large models have a non-negligible carbon footprint

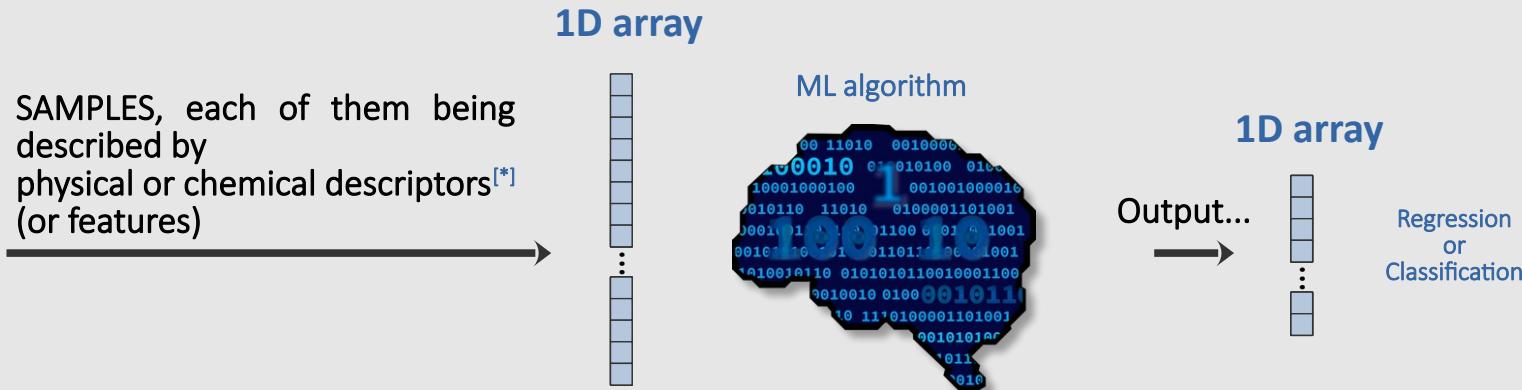
### Loss of skill development

If over-used as a substitute for critical thinking instead of a tool

# Typical workflow – with a bit of xAI



# General context



[\*] electron-counting rules, local aromaticity, HOMO-LUMO gaps, d-band centers, Fermi energy, global or local structural parameters, atomic charges, electron density, BDE, experimental spectrum, kinetic curve, thermal conductivity, heat capacity, resistivity, ductility,....

# Strategy



Collect data

Which data ?



How much?



Explainable AI tools  
to see through the  
*black box* models



Validate model  
performance  
on new data



Analyze and format  
data



Which  
algorithm?

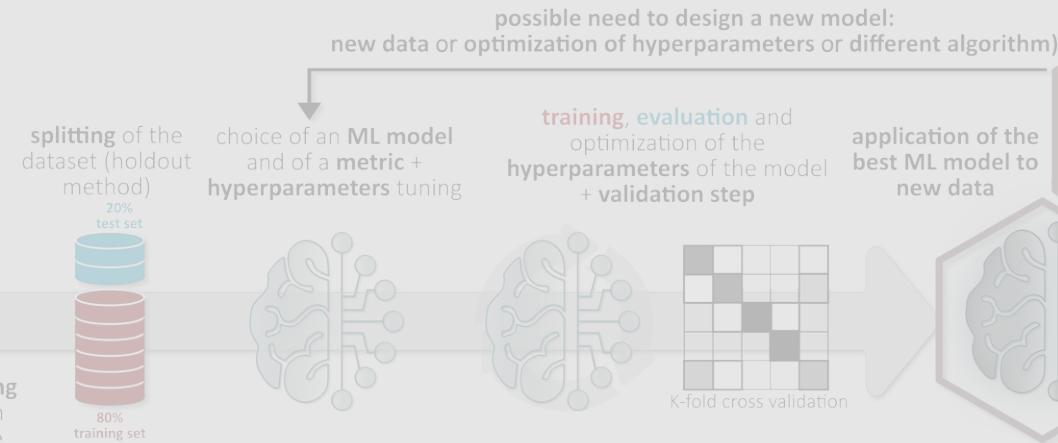
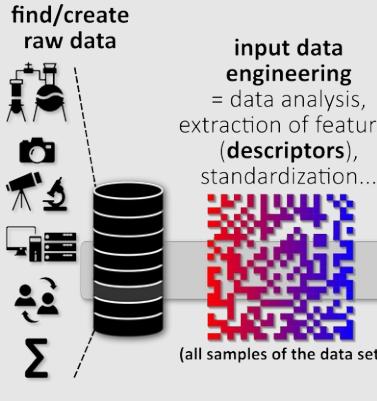


Train and optimize  
models

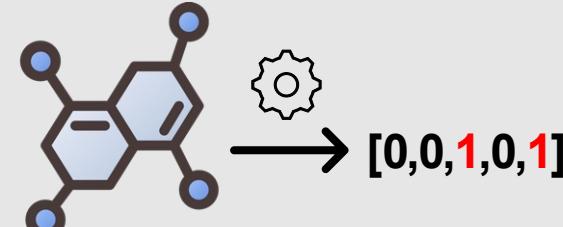


# Simplified – but recommended – workflow

## Data analysis & feature engineering

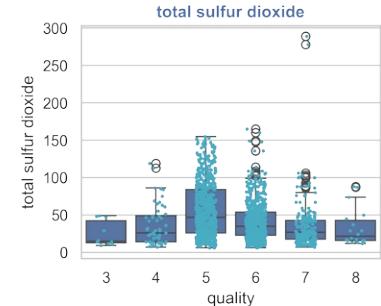
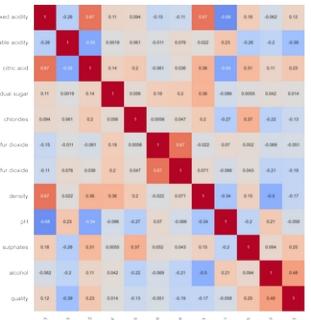


format and transform the raw data into a machine-readable form



Molecular case: conversion of molecular structures into descriptors that numerically represent their features, e.g. vectors

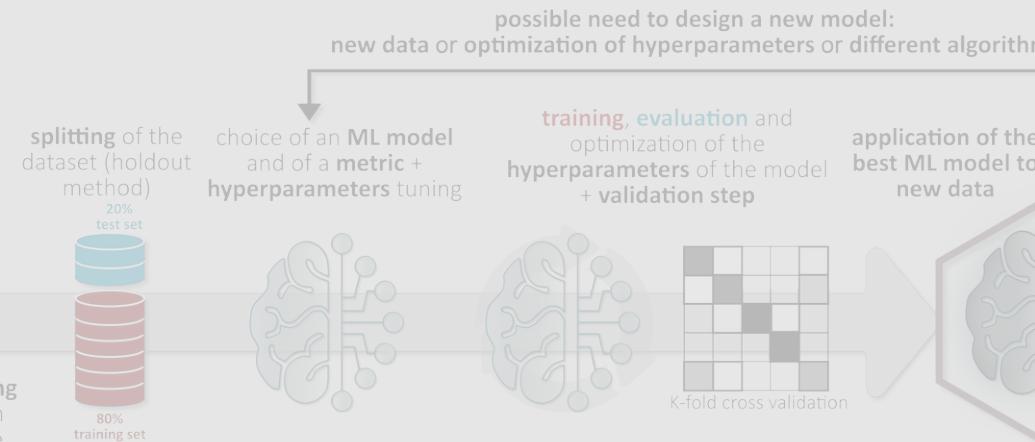
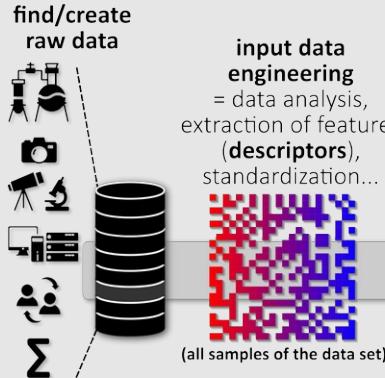
### Exploratory Data Analysis Understanding the Dataset Data Quality Assessment



# Simplified – but recommended – workflow

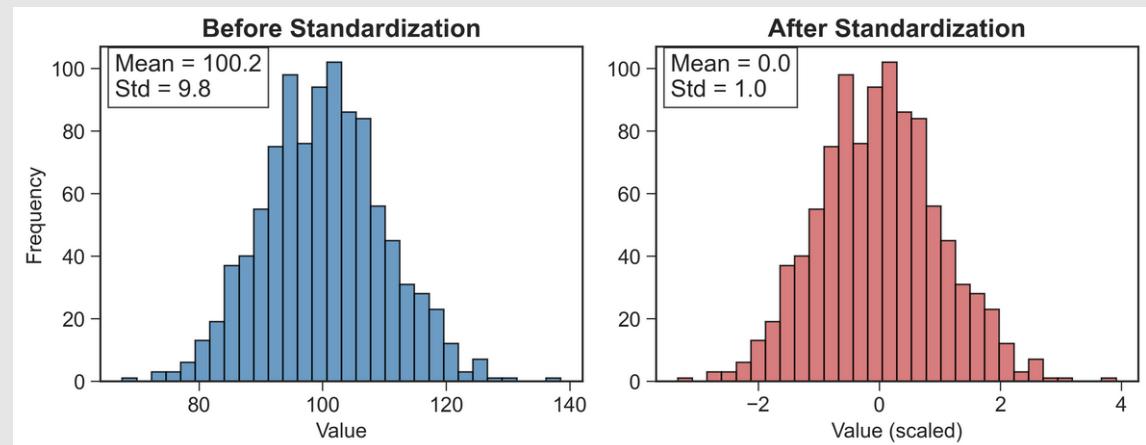
## Data analysis & feature engineering

Version: Sunday, December 7, 2025



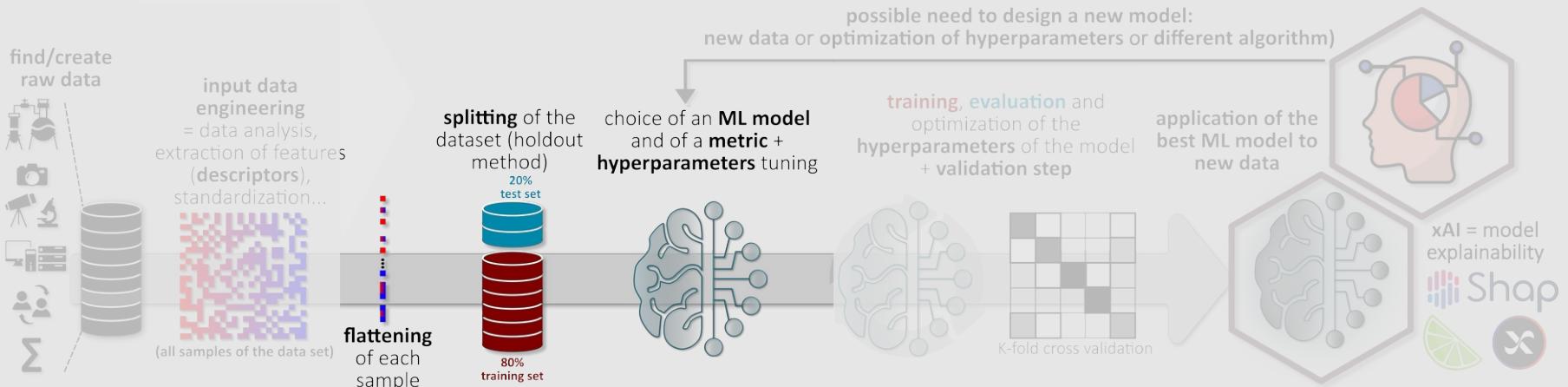
format and transform the raw data into a machine-readable form

data standardization



# Simplified – but recommended – workflow

## Choice of an algorithm & of its hyperparameters



### Supervised learning

we want the model to learn the relationship between input features/descriptors (X) and outputs - target (Y)

We split the data:

- 80 % for training → to teach the model
- 20 % for test → to check how well it learned on new data

Hyperparameters? e.g. number of neurons, number of neuron layers...

Usual metrics  
(=loss function)

$$\mathcal{L}_{\text{MAE}} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

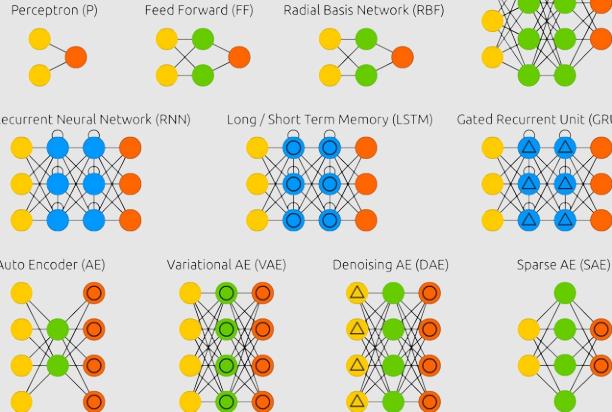
# Simplified – but recommended – workflow

## Choice of an algorithm: zoo of neural networks (NN)

- Input Cell
- Backfed Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Gated Memory Cell
- Kernel
- Convolution or Pool

### A mostly complete chart of Neural Networks

©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org



# Simplified – but recommended – workflow

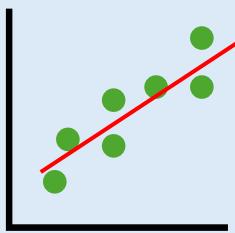
## Choice of an algorithm: neuron-free algorithms

### Linear methods

Fit a straight line through the data

Simple, fast, interpretable

Only for linear relationships



#### Examples:

- Linear regression
- Ridge regression

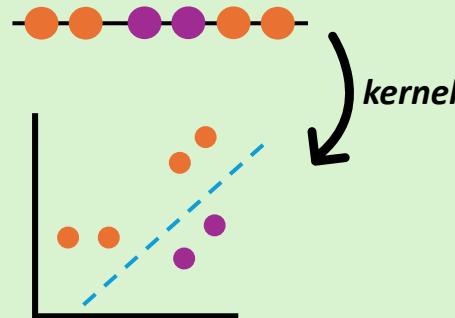


### Kernel-based methods

Use a mathematical trick to capture non-linear patterns

Flexible, good for small datasets

Computationally expensive for large data



#### Examples:

- Support Vector Machines
- Support Vector Regressors

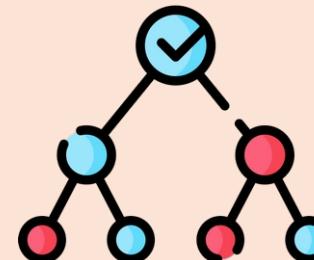


### Tree-based methods

Make rule-based “if–then” decisions

Power + interpretability, good for small datasets

Cannot extrapolate, instability



#### Examples:

- Decision Trees
- Random Forests

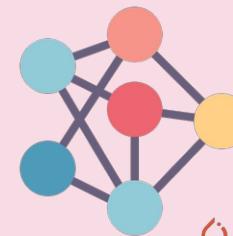


### Neural networks

Stack layers of “neurons” to learn complex patterns

Very powerful for large datasets

Need a lot of data, overfitting risk



PyTorch



#### Examples:

- Deep Neural Networks
- Autoencoders...

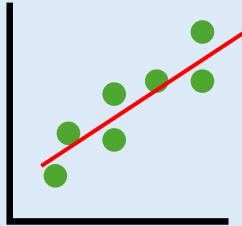
Keras

# Simplified – but recommended – workflow

Version: Sunday, December 7, 2025

## Linear methods

Fit a straight line through the data



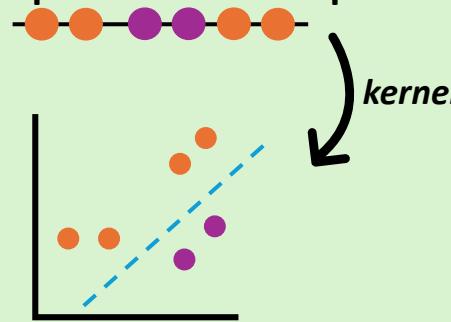
Examples:

- Linear regression
- Ridge regression



## Kernel-based methods

Use a mathematical trick to capture non-linear patterns



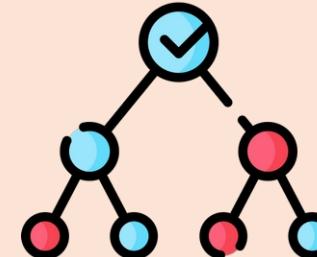
Examples:

- Support Vector Machines
- Support Vector Regressors



## Tree-based methods

Make rule-based “if–then” decisions



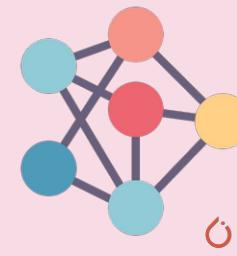
Examples:

- Decision Tree
- Random Forest



## Neural networks

Stack layers of “neurons” to learn complex patterns



Examples:

- Deep Neural Networks
- Autoencoders



## Bagging (Bootstrap Aggregating)

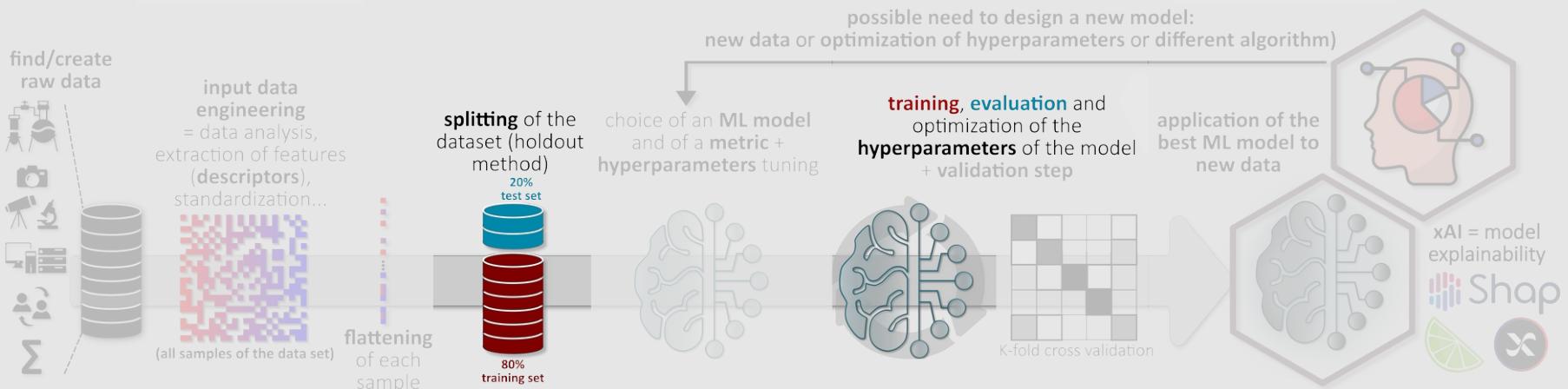
Train several models on random subsets of the data.  
Then average their predictions.

Example: SVR-bagging = multiple SVRs trained on different samples, predictions averaged.



# Simplified – but recommended – workflow

## Model training – Supervised learning



We want the model to learn the relationship between input features/descriptors ( $X$ ) and outputs - target ( $Y$ )

The model makes predictions  $\hat{Y}$

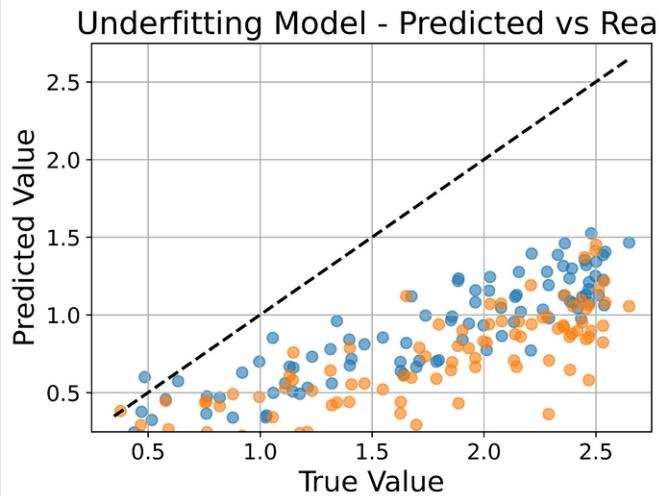
Learning process: minimizing the error between actual values ( $Y$ ) and predicted values ( $\hat{Y}$ ) by adjusting the model's internal parameters

It can be iterative (NN case) or analytical (linear least square fit case)

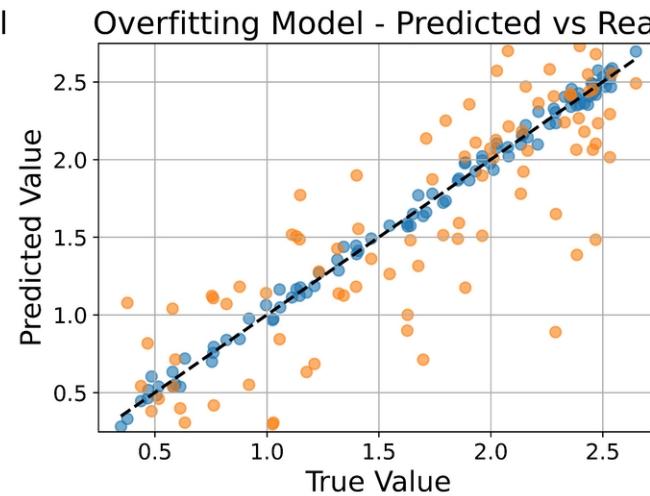
# Simplified – but recommended – workflow

## Model evaluation – at the end of the training process. Regression case

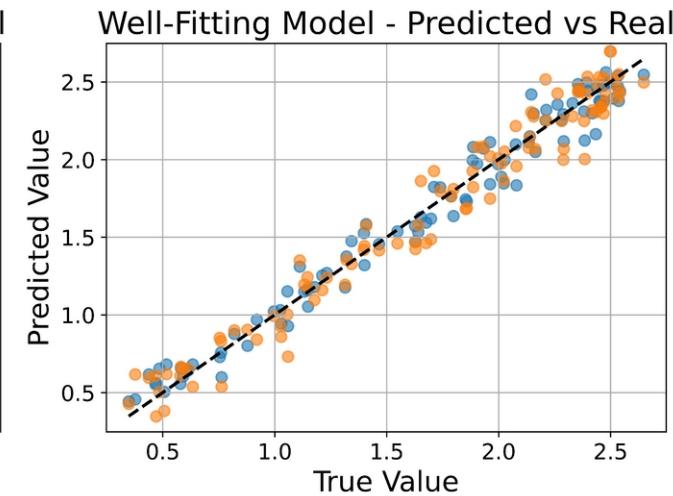
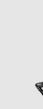
The model is too simple, fails to capture patterns in the training data → poor on both training and test sets



The model is too complex, memorizes the training data → good on training but poor on new data



The model captures the main trends in the data → good performance and generalization



Performance can possibly be improved by changing the hyperparameters of the model

# Simplified – but recommended – workflow

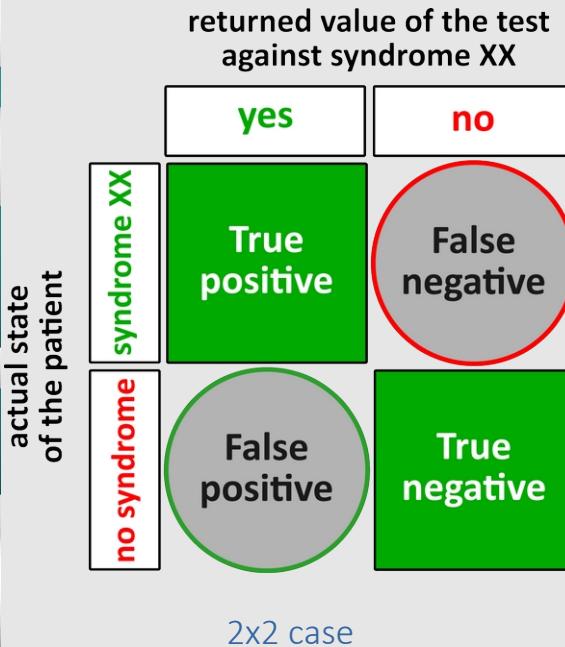
## Model evaluation – at the end of the training process. Classification case

consists in plotting confusion matrices (CMs)

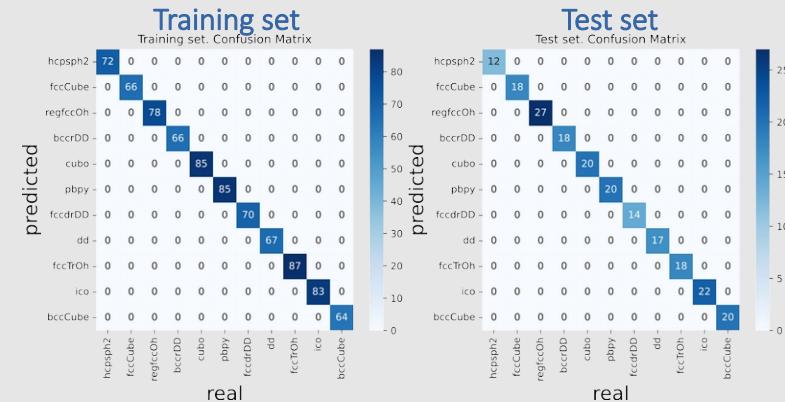
A confusion matrix is a  $2 \times 2$  table (for binary classification) comparing the model's predictions to the true labels

Same logic as a biological/medical diagnostic test for a syndrome

- A **sensitive** test minimizes false negatives → good for dangerous syndromes
- A **specific** test minimizes false positives → good when false alarms are costly
- ML models behave exactly like diagnostic tests: threshold tuning shifts balance between sensitivity and specificity



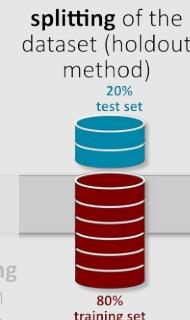
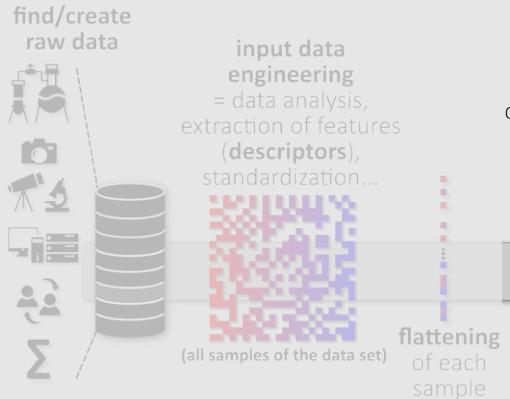
CM generalizes naturally to multi-class classification



# Simplified – but recommended – workflow

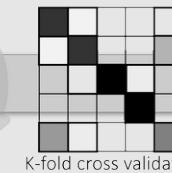
Version: Sunday, December 7, 2025

## Model validation



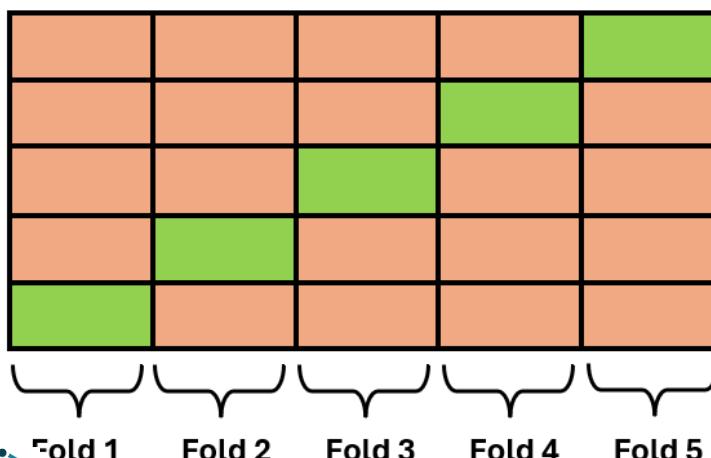
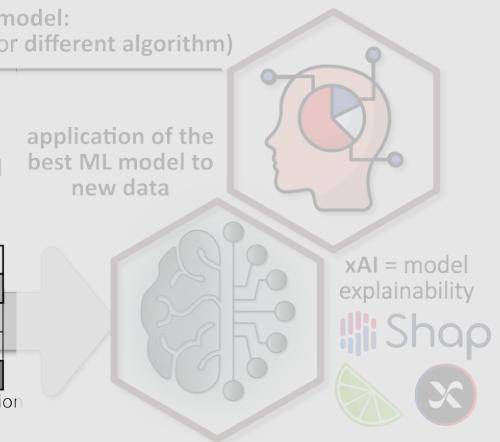
choice of an ML model and of a metric + hyperparameters tuning

training, evaluation and optimization of the hyperparameters of the model + validation step

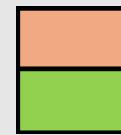


K-fold cross validation

possible need to design a new model:  
new data or optimization of hyperparameters or different algorithm)



### Cross-Validation



Training  
Testing



Generalizability?

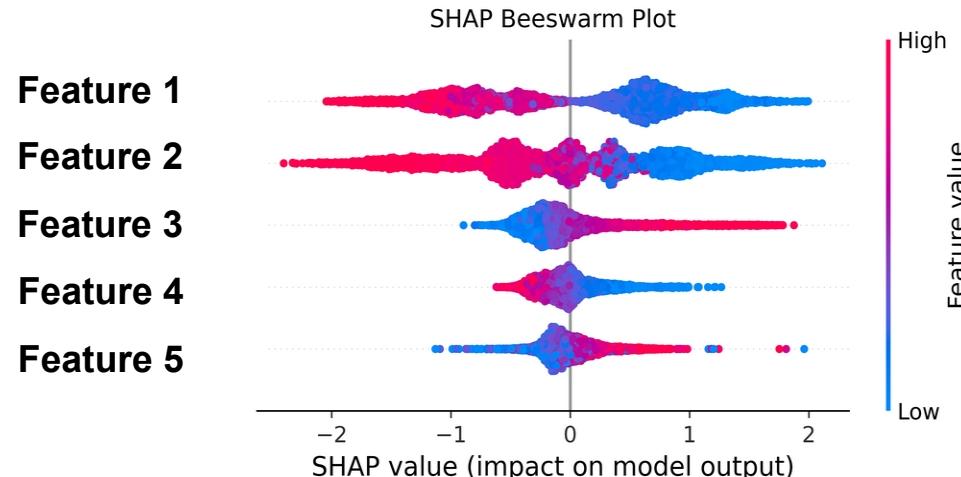
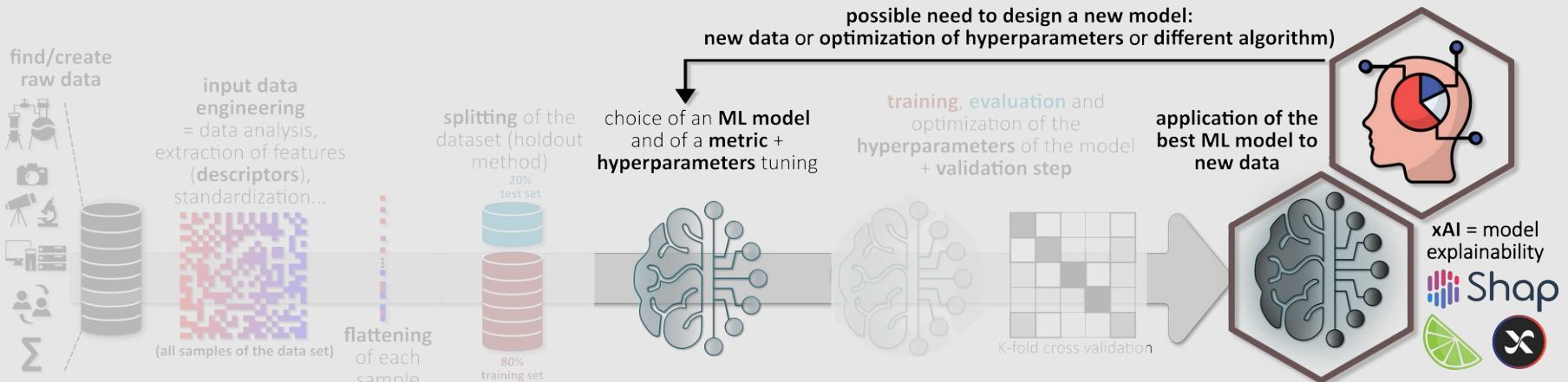
### Extrapolation



New “challenging” sets of data

# Simplified – but recommended – workflow

## Model explainability = eXplainable AI (xAI)



- **Feature importance**
- Ensures that predictions are chemically grounded and trustworthy
- Prediction variation with varying descriptor values
- Might encourage to consider another model



# How many data?

There is no single magic number

Data requirements vary by several orders of magnitude depending on whether you are building a generalist “chemical brain” or a specialized expert tool

Typically

- **Generalist AI:  $10^9$ .** Example = LLMs that can understand the language and concepts of chemistry
- **Specialist AI:  $10^6$ .** Example = Proprietary databases of chemical reactions
- **Expert AI:  $10^3$ .** Example = Structure-Activity Relationships (SAR)

Don't forget: quality is better than quantity

# Conclusion

## Key Insights & Perspectives for AI in Chemistry

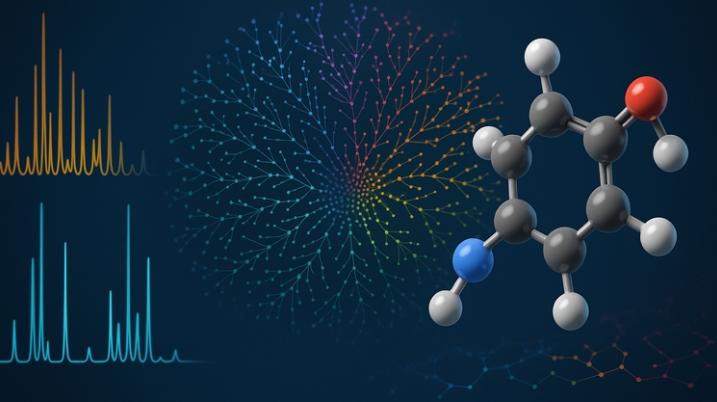


Image generated with ChatGPT 5.0, December 2025

Version: Sunday, December 2025

### 4. Models must be validated outside their comfort zone

Applicability domain, extrapolation limits, and chemical plausibility checks are essential, especially in materials design, catalysis, and solvent engineering.



### 5. "Assistive AI" can accelerate research

From designing new descriptors to writing code, general AIs, or – even better, both in terms of energy cost and efficiency ? – specialized AIs can be a powerful assistant, but requires human expertise to avoid biases, hallucinations, and misleading outputs.

### 6. AI is a tool for creativity in chemistry

When combined with theory, computation, and experiment, AI opens new ways to explore chemical space, test hypotheses, and design better molecules/materials.