Outline of this lecture

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 worflow – including xAI

6. Tutorials / Live demonstrations ← Jupyter notebooks

General context

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

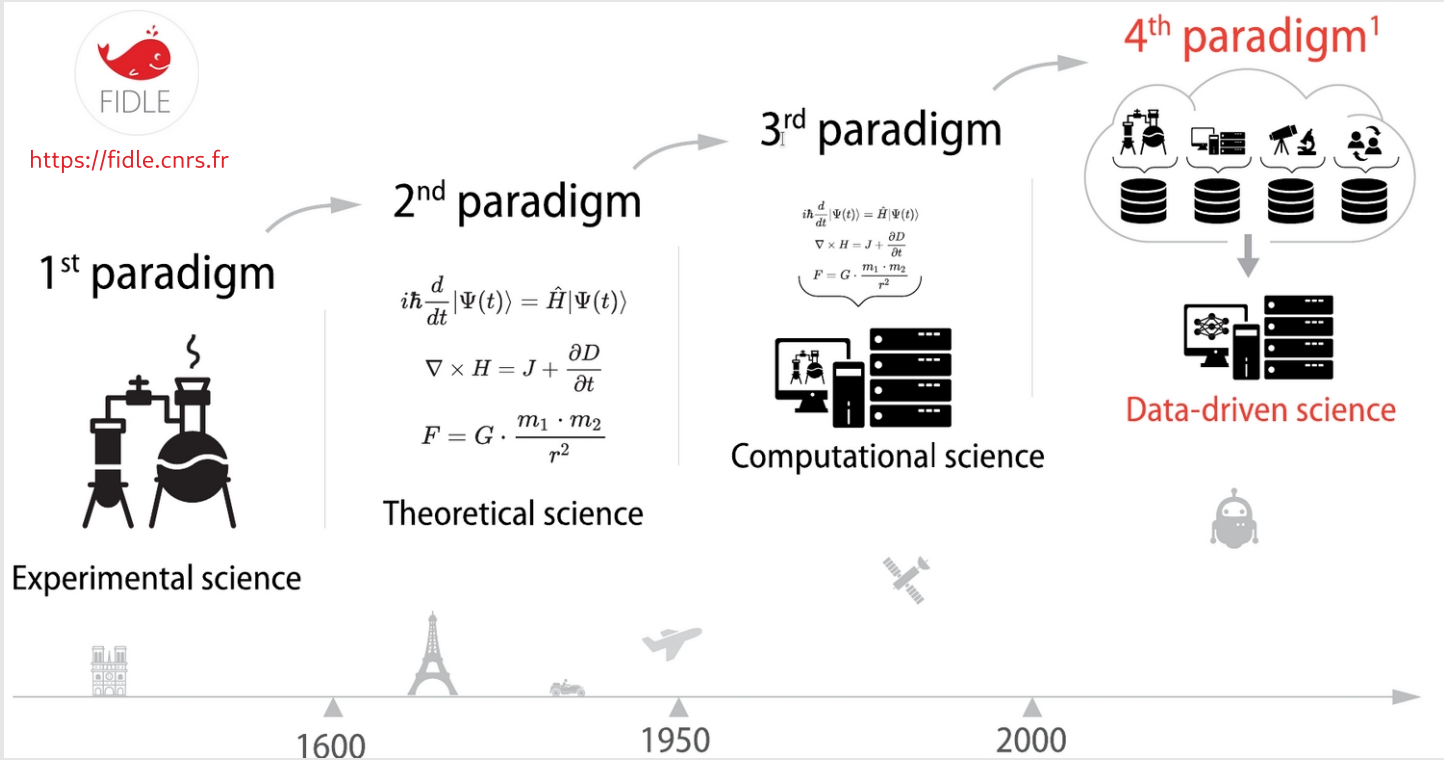
ability to move and manipulate objects

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



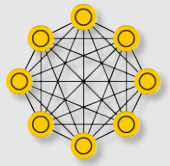
(1) Hey, T.; Tansley, S.; Tolle, K. The Fourth Paradigm: Data-Intensive Scientific Discovery; The Fourth Paradigm: Data-Intensive Scientific Discovery; Microsoft Research, 2009

Outlook: data science and machine learning

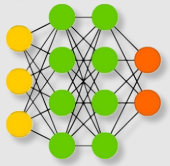
Artificial Intelligence and Artificial Neural Networks (ANN) timeline

Perceptron (P)

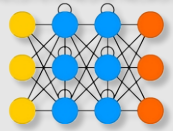
Hopfield Network (HN)



Deep Feed Forward (DFF)



Recurrent Neural Network (RNN)



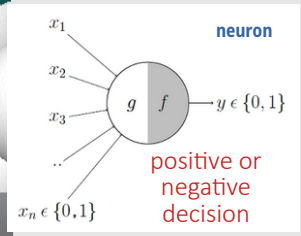
1950
Foundation of computation and intelligent algorithm
Turing

1986
Hidden layers
Rumelhart, Hinton & Williams

1986
Recurrent NN
Jordan

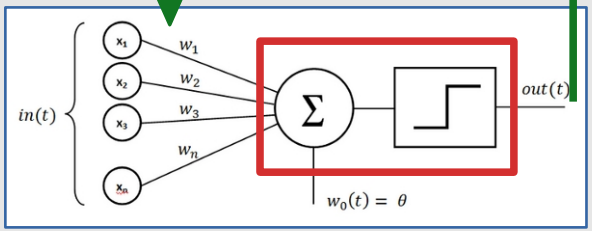
1943
Neural Nets
McCulloch & Pitt

were the first to suggest that neural activity is computational



1958
Perceptron model
Rosenblatt

1970
Backpropagation
Linnainmaa

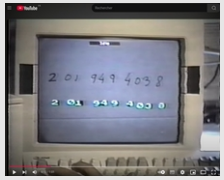


1982
Hopfield network
Hopfield

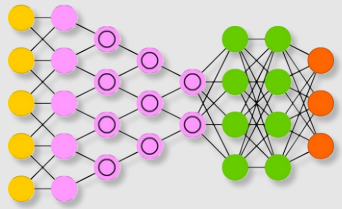
1985
Boltzmann machines
Ackley, Hinton & Sejnowski

Boltzmann Machine (BM)

1990
LeNet
LeCun



Deep Convolutional Network (DCN)



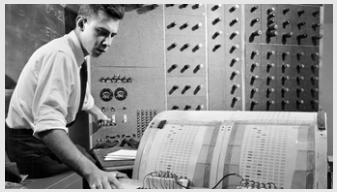


Image made by R.P. with ChatGPT 4o, November 2024

W. S. McCulloch & W. Pitts (**1943**)
A logical calculus of the ideas immanent in nervous activity
Bull. Math. Biophys. **5**: 115-133

F. Rosenblatt (**1958**)
The perceptron: A probabilistic model for information storage and organization in the brain
Psychol. Rev. **65**: 386-408

S. Linnainmaa (**1970**)
The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors (Masters thesis, in Finnish), University of Helsinki. pp. 6–7

J. J. Hopfield (**1982**)
Neural networks and physical systems with emergent collective computational abilities
Proc. Natl. Acad. Sci. U S A. **79**:2554

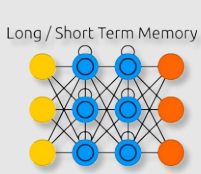
D. H. Ackley, G. E. Hinton, G. E., T. J. Sejnowski (**1985**)
A learning algorithm for Boltzmann Machines
Cog. Sci. **9**: 147

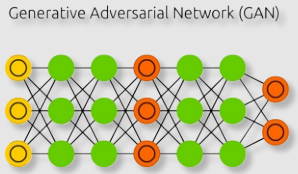
D. Rumelhart, G. Hinton & R. Williams (**1986**)
Learning representations by back-propagating errors
Nature **323**: 533

M. I. Jordan (**1986**)
Serial order: A parallel distributed processing approach
ICS Report 8604. Institute for Cognitive Science, UCSD, La Jolla

Outlook: data science and machine learning

Artificial Intelligence and Artificial Neural Networks (ANN) timeline





Levinthal's paradox solved!?

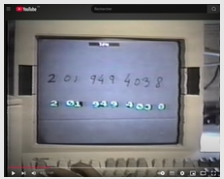
1996
LSTM RNN
Hochreiter & Schmidhuber

2014
GANs
Goodfellow

2018
AlphaFold 1
Google DeepMind

2024
AlphaFold 3
Google DeepMind

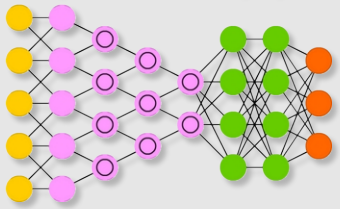
1990
LeNet
LeCun

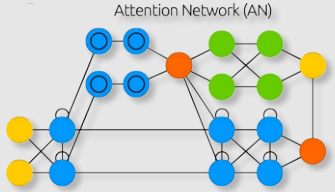


2017
AlphaGo
Google DeepMind

2020
AlphaFold 2
Google DeepMind

2017
Transformers, LLM
Google





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-Abade, 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.

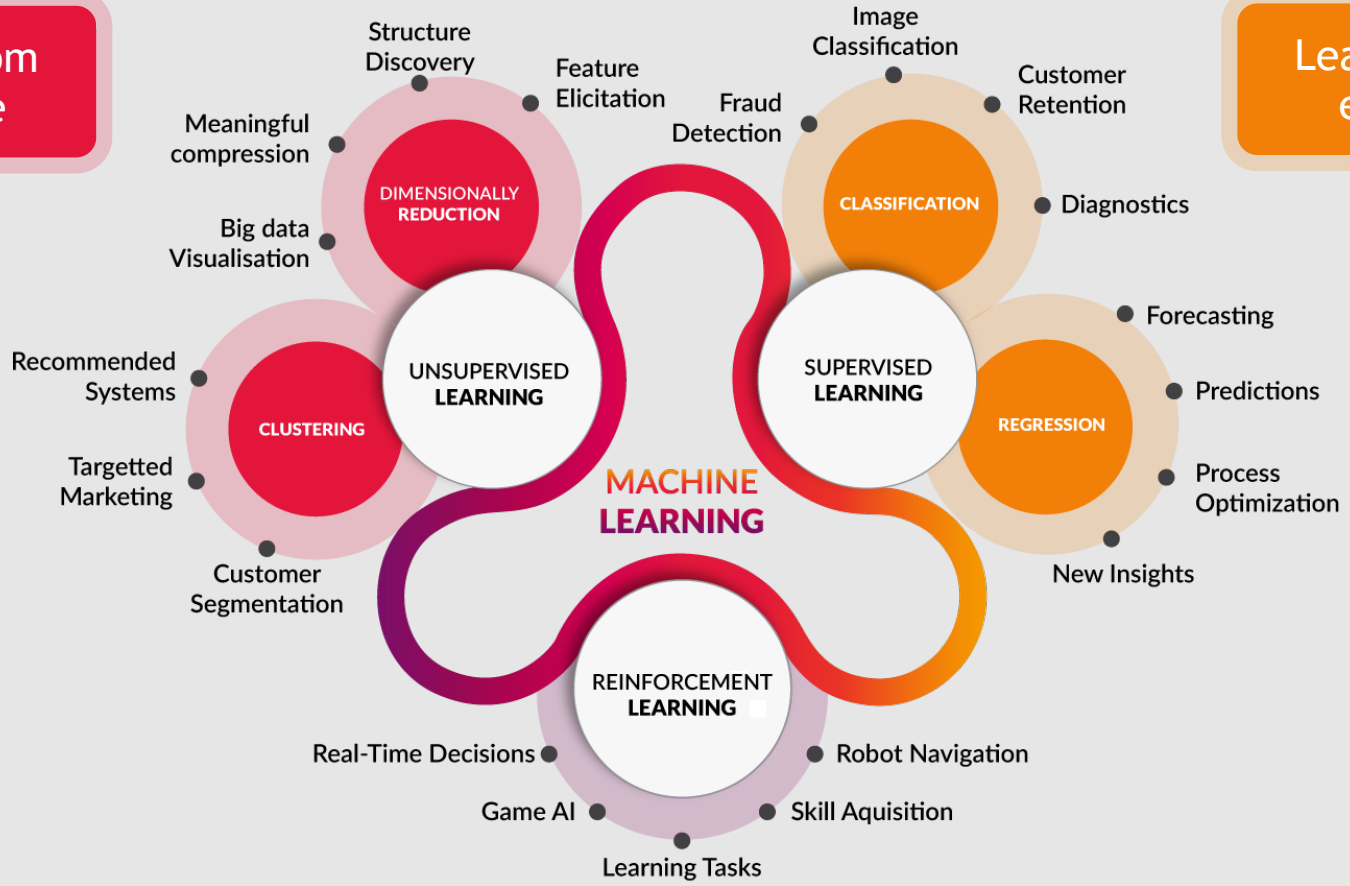
J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, S. W. Bodenstein, D. A. Evans, C.-C. Hung, M. O'Neill, D. Reiman, K. Tunyasuvunakool, Z. Wu, A. Žemgulytė, E. Arvaniti, C. Beattie, O. Bertolli, A. Bridgland, A. Cherepanov, M. Congreve, 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

Learning from data alone

Learning from examples



https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d

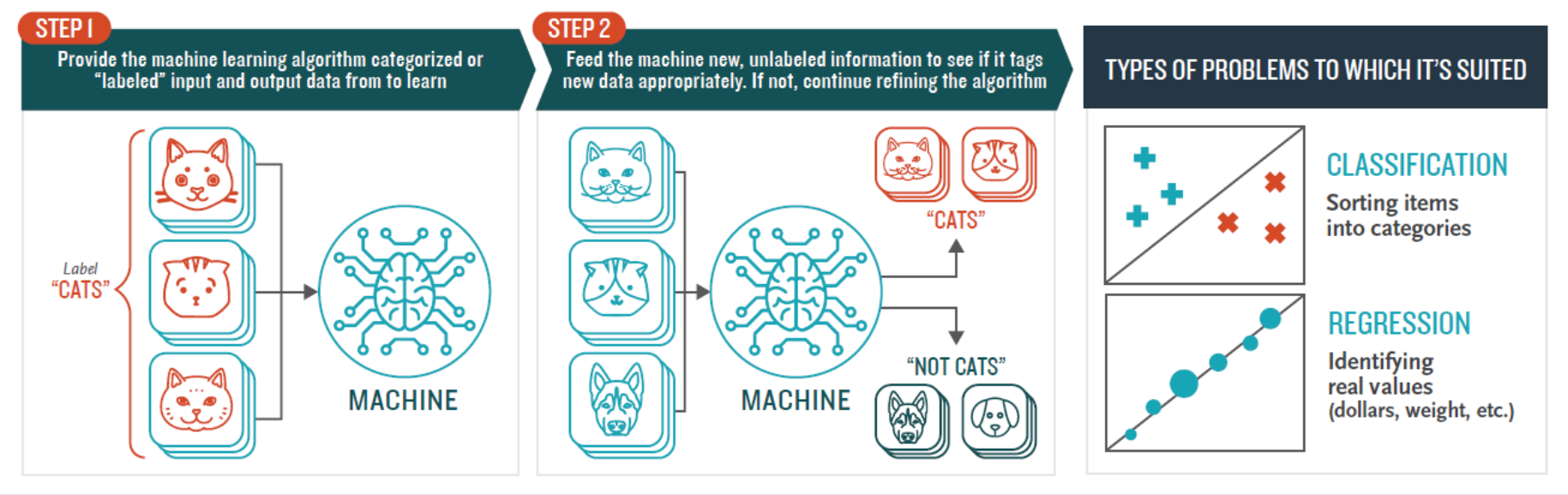
Supervised learning

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

Learning from examples

<u>Example of cats detection</u>:

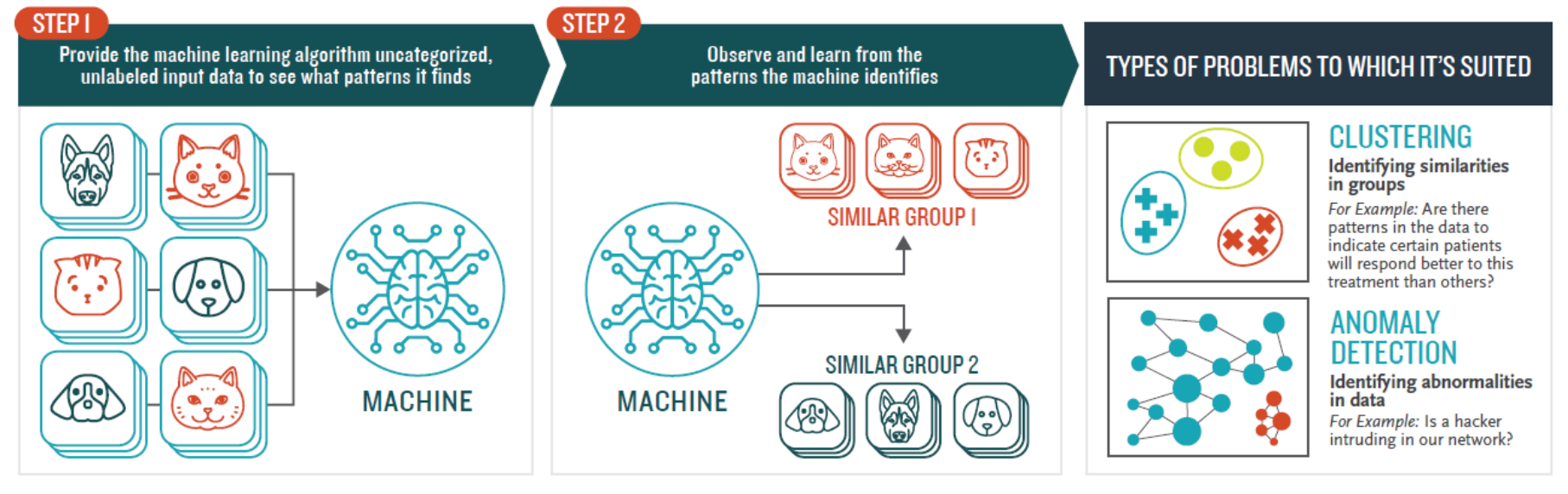
- **collect thousands of images of cats**
- **draw a bounding box around each cat**
- **feed the entire dataset to the machine so it can learn all by itself**



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

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)

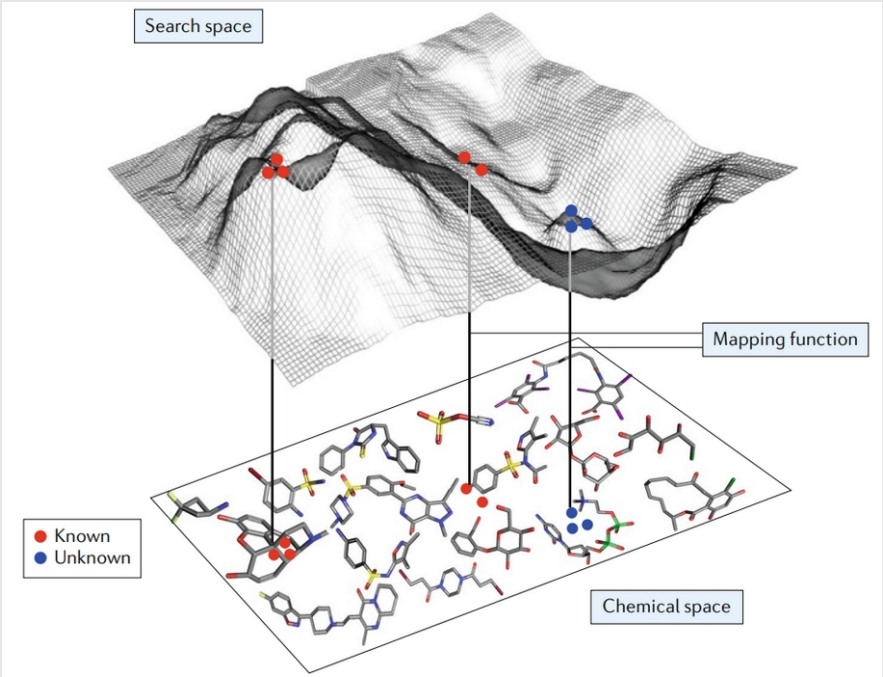
Data Exploration

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

One of the numerous motivations: from data-rich chemistry to data-driven discovery

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

Gromski, P.S., Henson, A.B., Granda, J.M. et al. (**2019**) *Nat Rev Chem* **3**: 119–128

Development of AI tools with python

How to develop home-made ML tools?

mathematica

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://keras.io/

https://www.tensorflow.org/

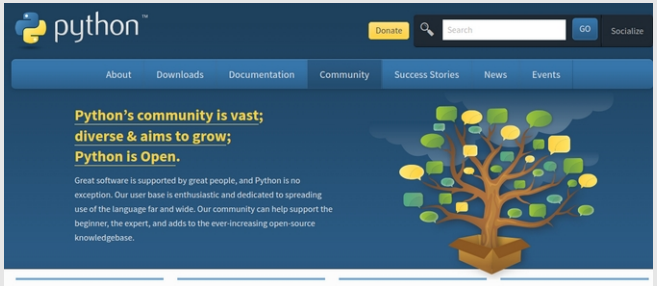
pymatgen

Using Python as an everyday tool for scientific calculation and computation

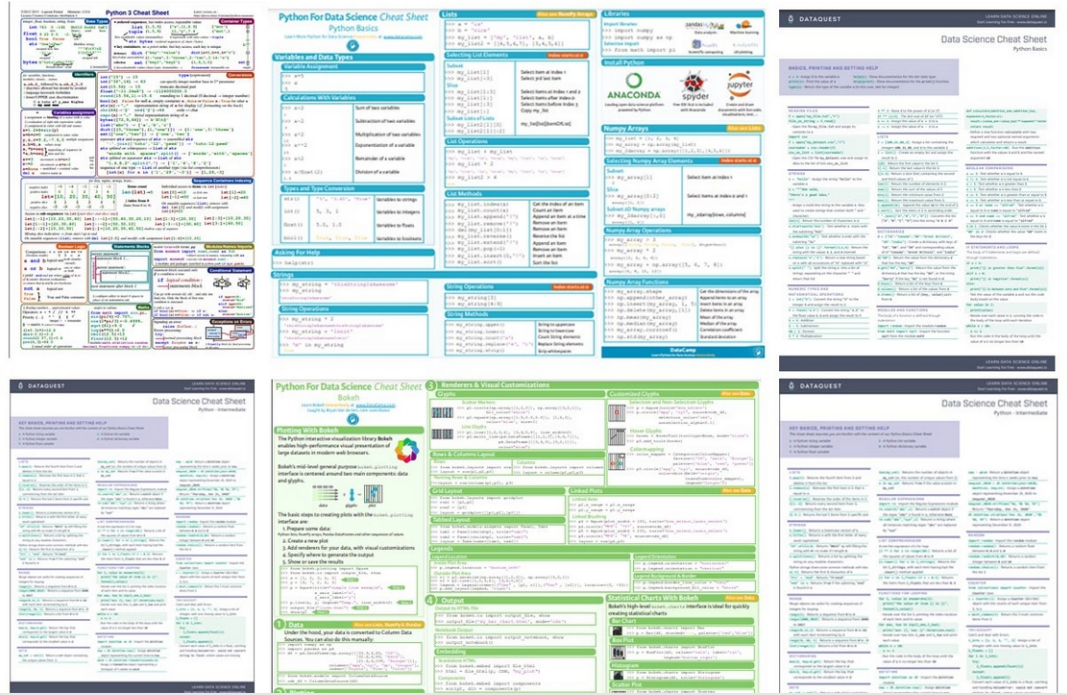
(it can even replace excel… by far)

→ basic knowledge of programming languages (variables, arrays, loops, conditional tests...)

→ enthusiastic and vast community



→ cheatsheets & cut/paste



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

ML using the Python programming language

https://scikit-learn.org/

Machine learning in Python with scikit-learn

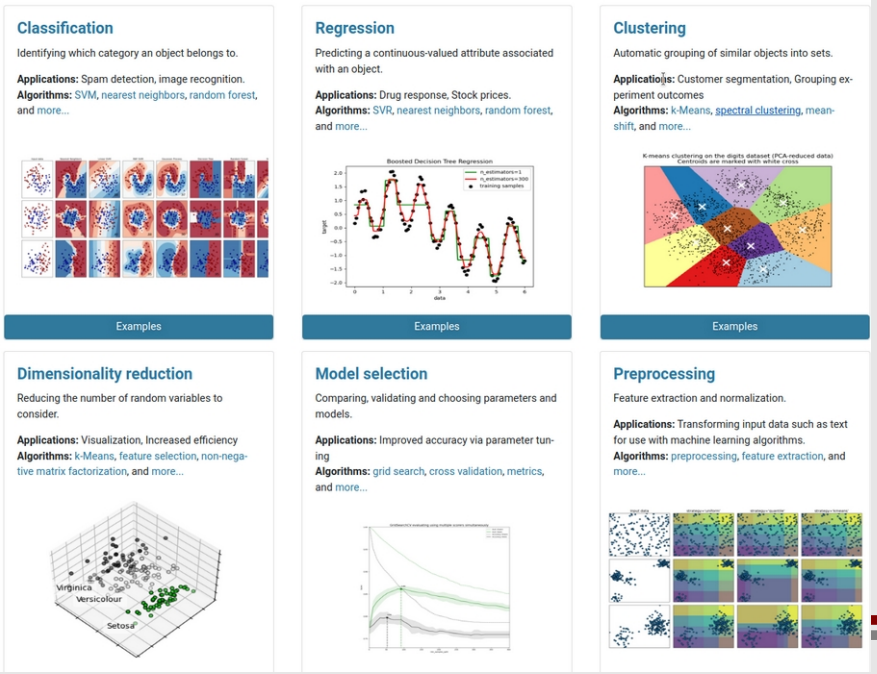
Simple and efficient tools for predictive data analysis

Accessible to everybody, and reusable in various contexts

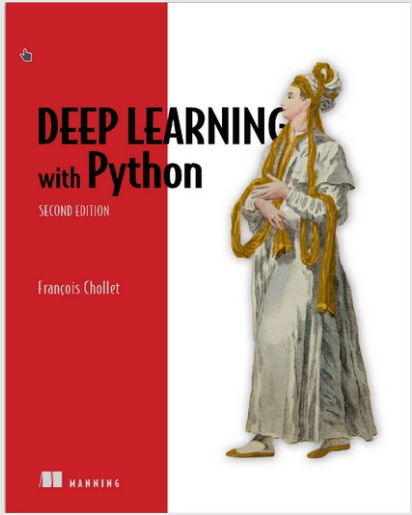
Built on NumPy, SciPy, and matplotlib

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)



https://keras.io/



High Level Deep Learning Application Programming Interface (API)

By François Cholet (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)

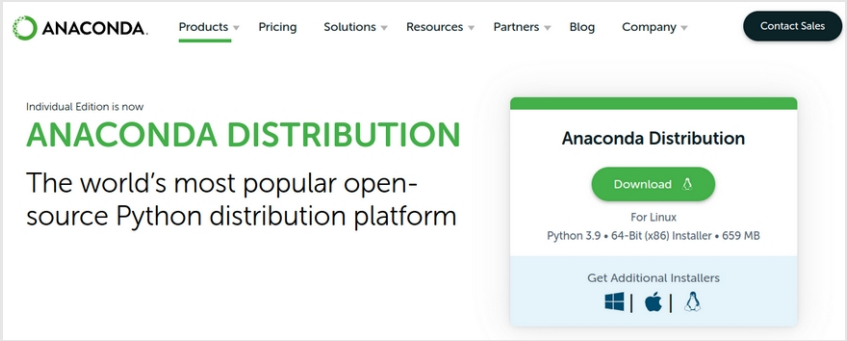
https://pytorch.org/

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

ANACONDA



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

jupyter

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 (SHIT-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 a 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

Python in the Physical Chemistry Lab

[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 autoencodeurs. 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

🇬🇧 Table of Contents

Python programming language

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

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

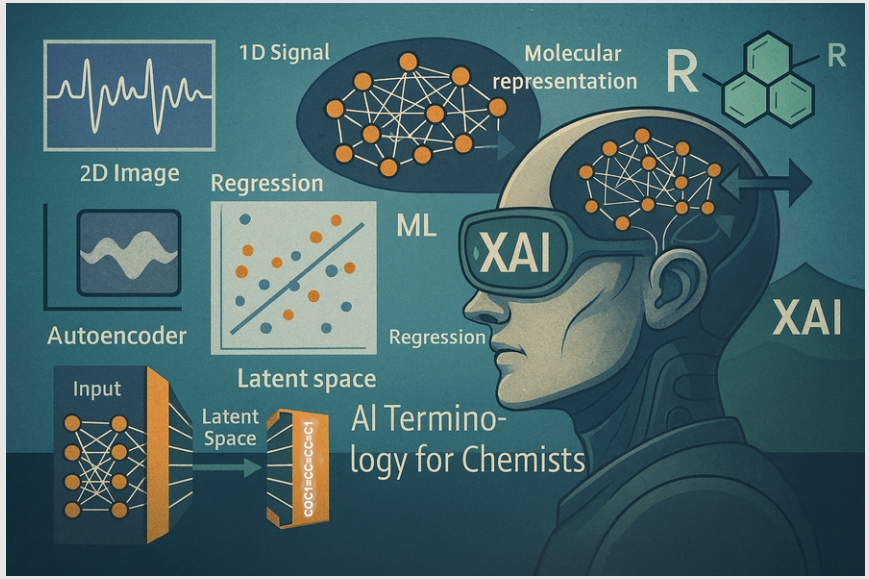
Data leakage / confidentiality concerns
(if code or sensitive data are pasted in public LLMs)

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

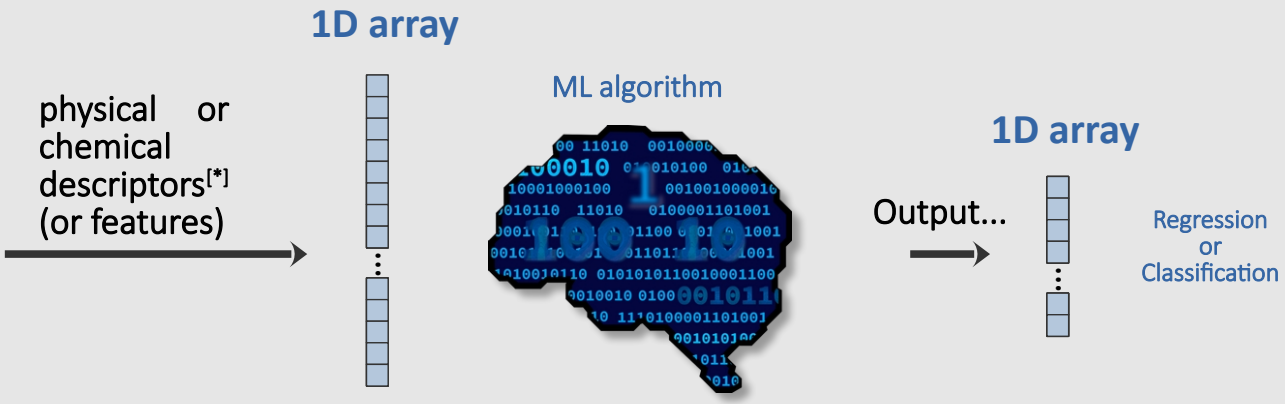
Loss of skill development
If over-used as a substitute for critical thinking instead of a tool

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

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

Which data ?

How much?

Collect data

Analyze and format data

Hidden patterns?

Train and optimize models

Which algorithm?

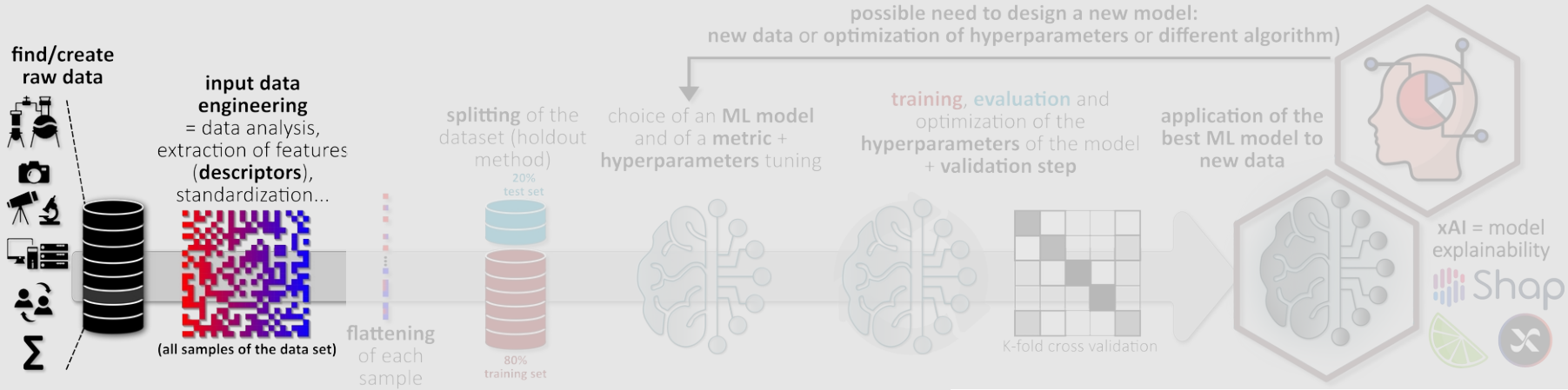
Validate model performance on new data

What did the model learn?

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

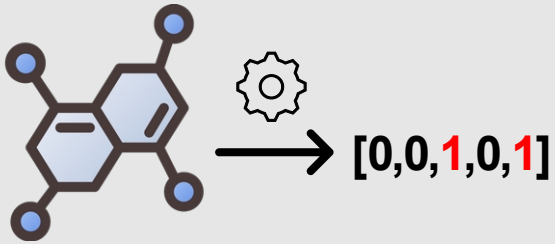
Simplified – but recommended – workflow

Data analysis & feature engineering



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

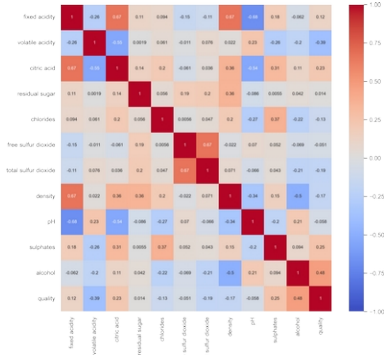
Molecular case: conversion of molecular structures into descriptors that numerically represent their features, e.g. vectors

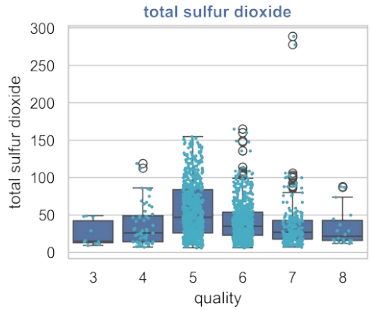


Exploratory Data Analysis

Understanding the Dataset

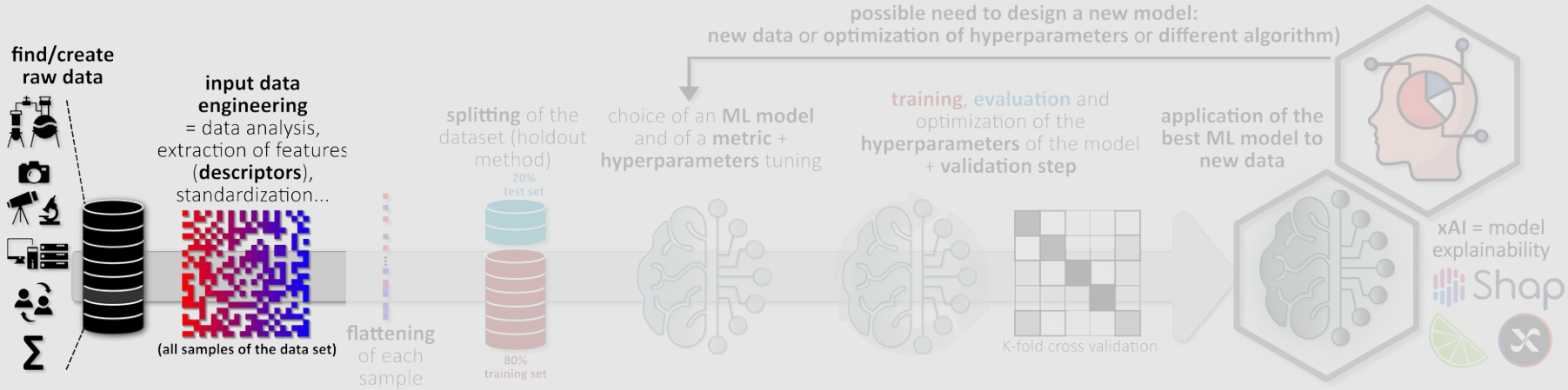
Data Quality Assessment





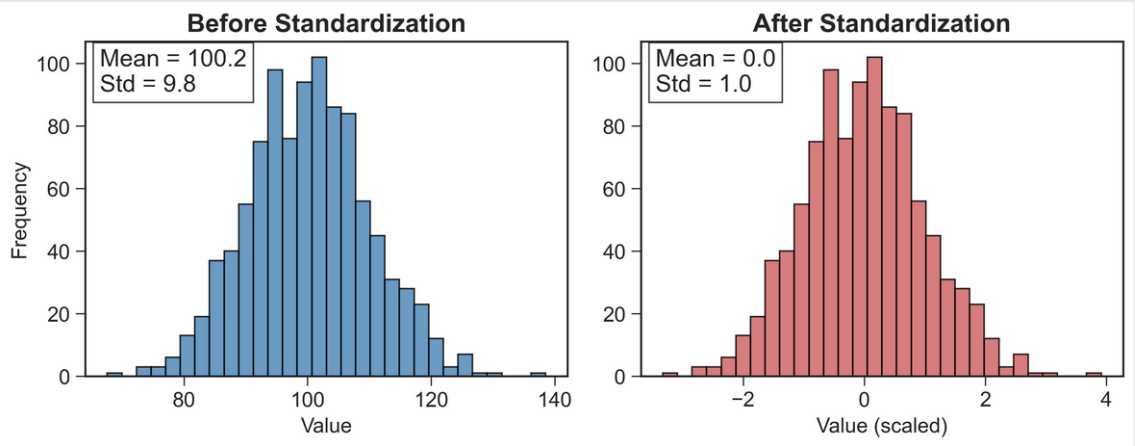
Simplified – but recommended – workflow

Data analysis & feature engineering



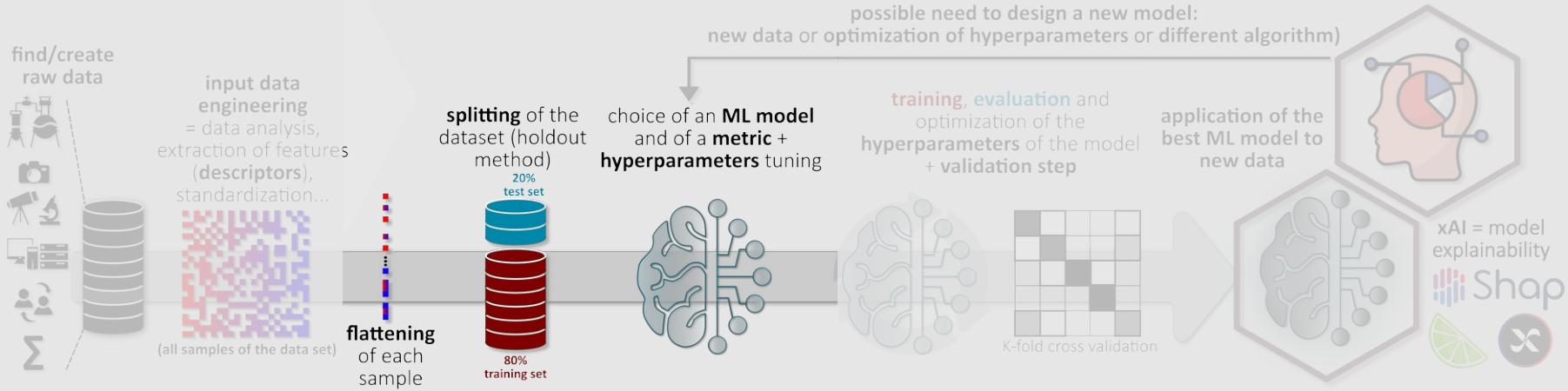
format and transform the raw data into a machine-readable form

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...

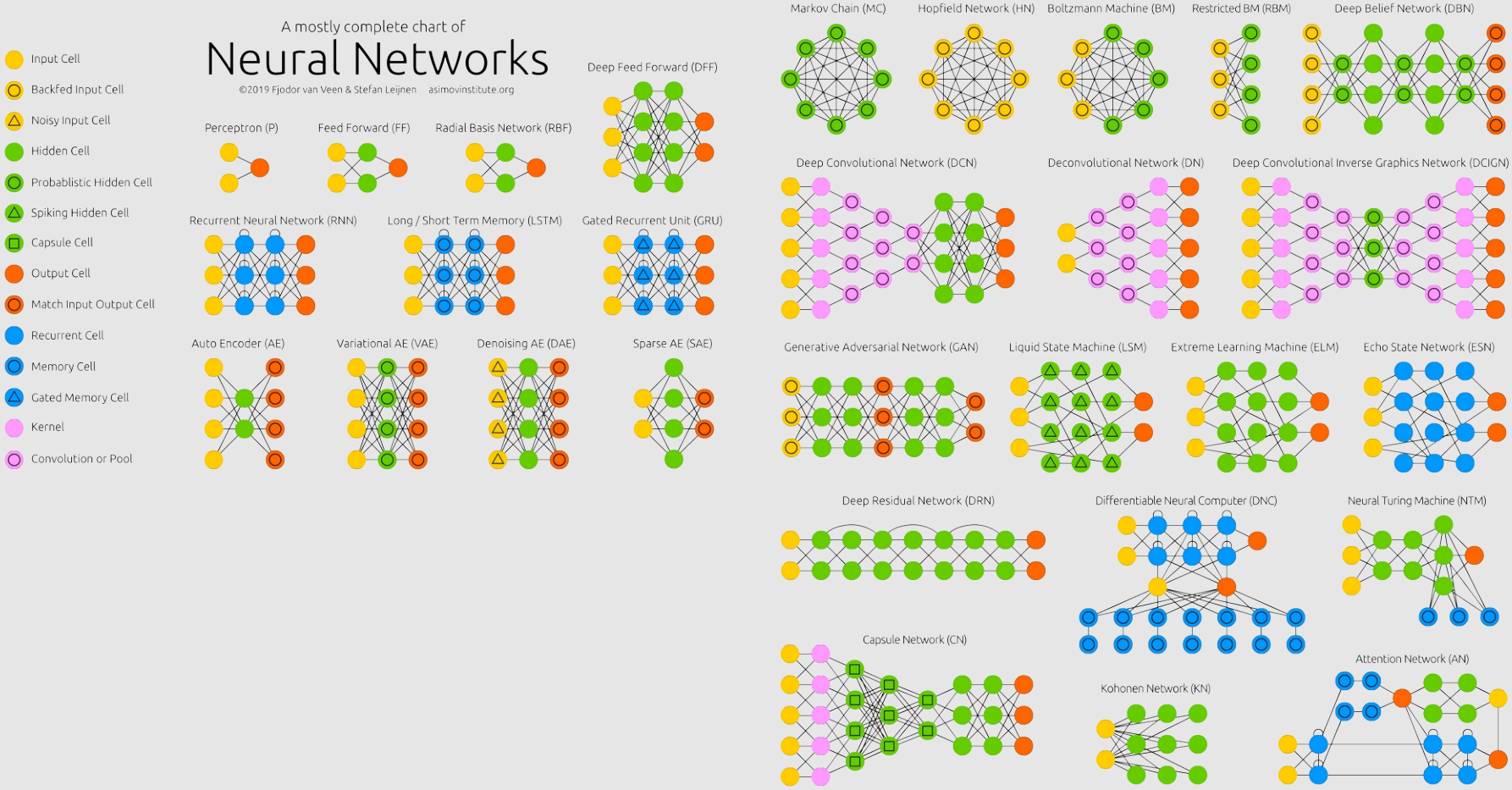
Usual metrics (=loss function)

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

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

Simplified – but recommended – workflow

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



Simplified – but recommended – workflow

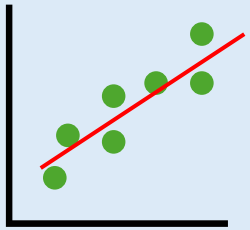
Choice of an algorithm: neuron-free algorithms

🟦 Linear methods

Fit a straight line through the data

Simple, fast, interpretable

Only for linear relationships



Examples:

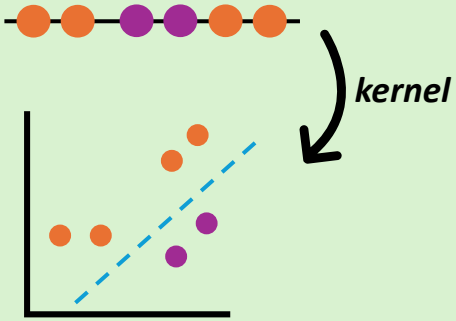
- Linear regression
- Ridge regression

🟩 Kernel-based methods

Use a mathematical trick to capture non-linear patterns

Flexible, good for small datasets

Computationally expensive for large data



Examples:

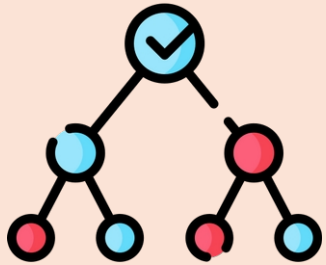
- Support Vector Machines
- Support Vector Regressors

🟧 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:

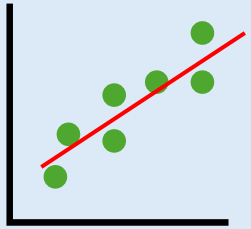
- Deep Neural Networks
- Autoencoders...

Keras

Simplified – but recommended – workflow

🟦 Linear methods

Fit a straight line through the data

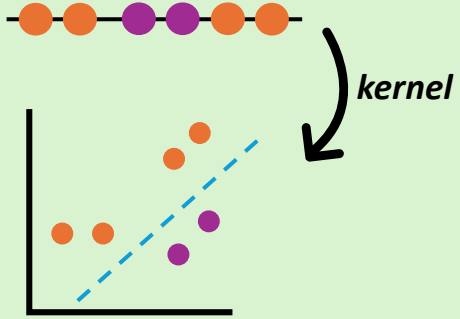


Examples:

- **Linear regression**
- **Ridge regression**

🟩 Kernel-based methods

Use a mathematical trick to capture non-linear patterns

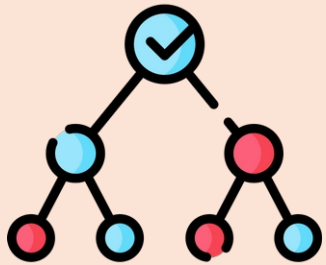


Examples:

- **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

PyTorch

Examples:

- **Deep Neural Networks**
- **Autoencoders**

Keras

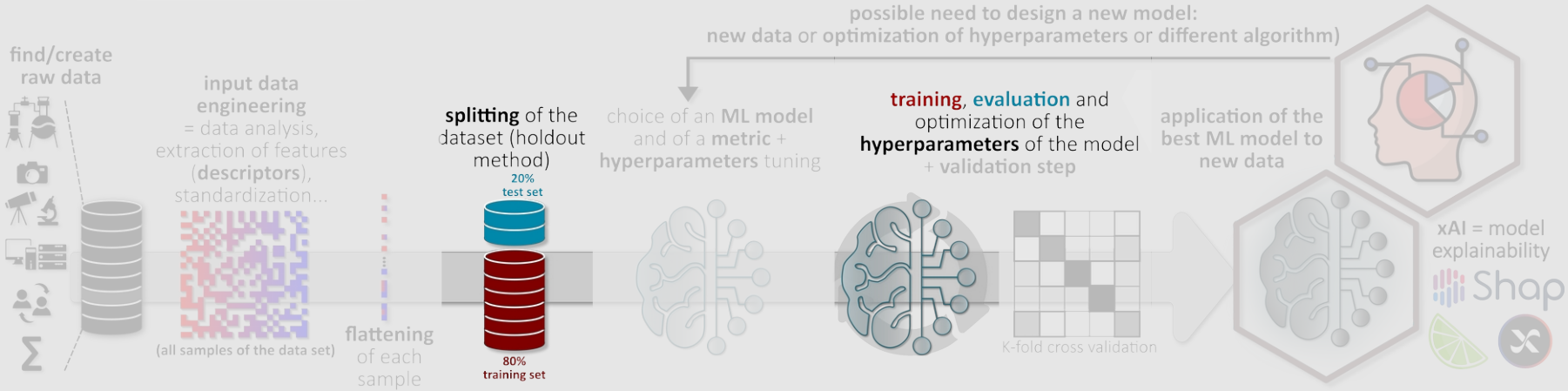
⬛ Bagging (Bootstrap Aggregating)

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 pedicted values ($\hat{Y}$) by adjusting the model's internal parameters

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

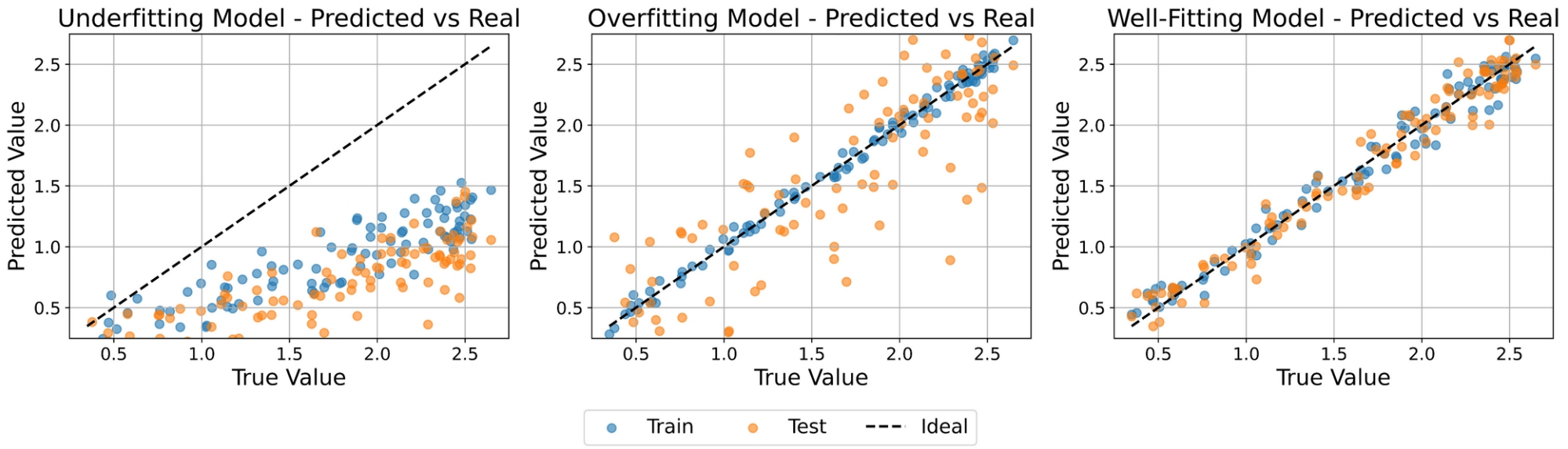
Simplified – but recommended – workflow

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

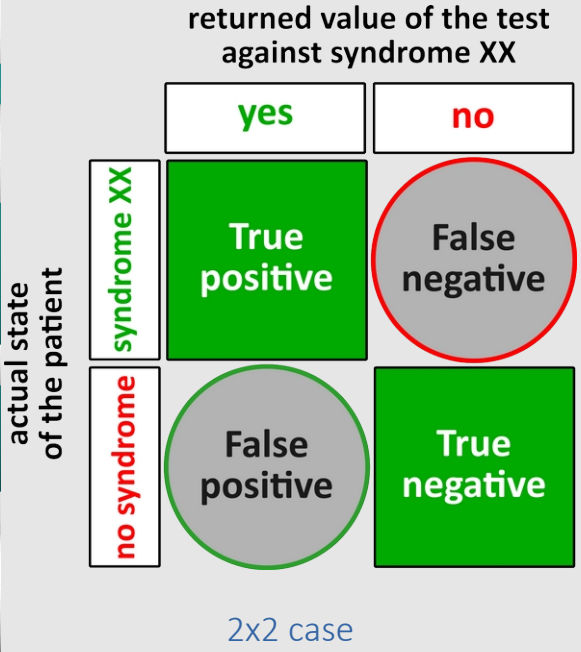
The model captures the main trends in the data → good performance and generalization



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



2x2 case

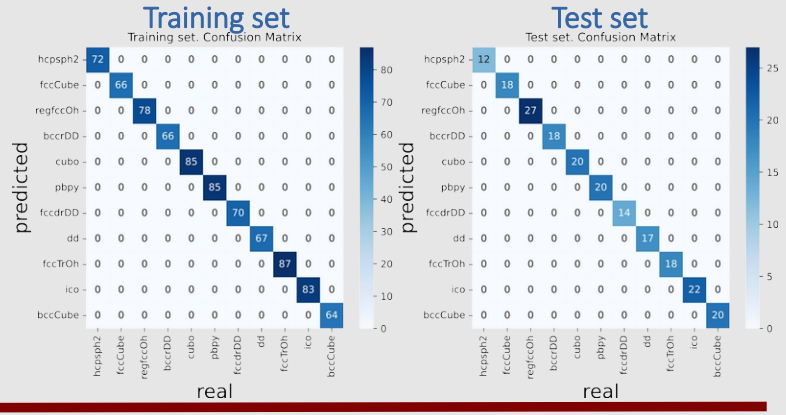
consists in plotting confusion matrices (CMs)

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

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

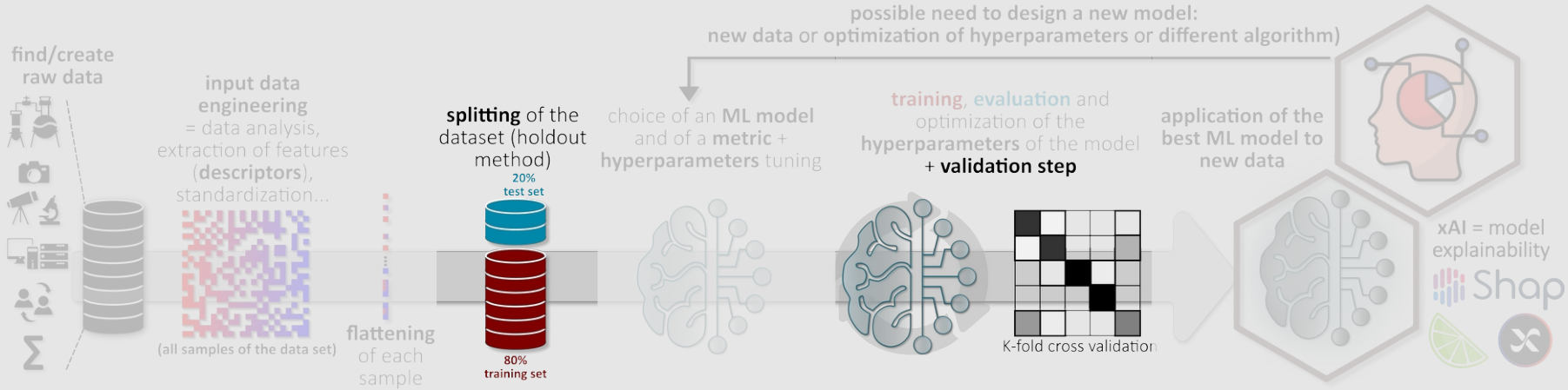
- **A sensitive test minimizes false negatives → good for dangerous syndromes**
- **A specific test minimizes false positives → good when false alarms are costly**
- **ML models behave exactly like diagnostic tests: threshold tuning shifts balance between sensitivity and specificity**

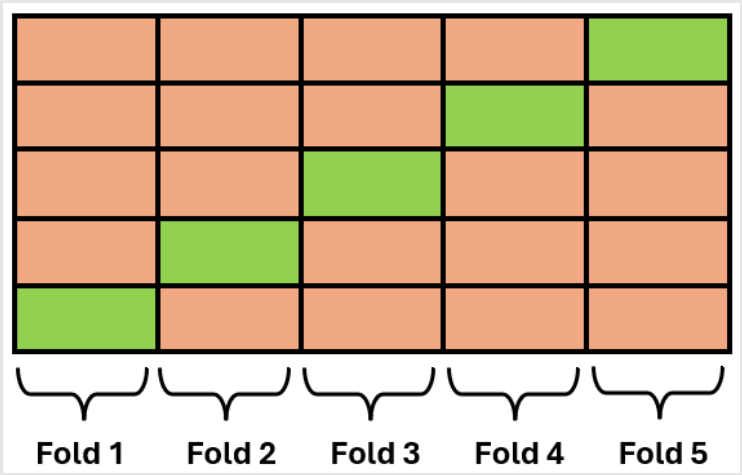
CM generalizes naturally to multi-class classification



Simplified – but recommended – workflow

Model validation





Cross-Validation

Training
Testing

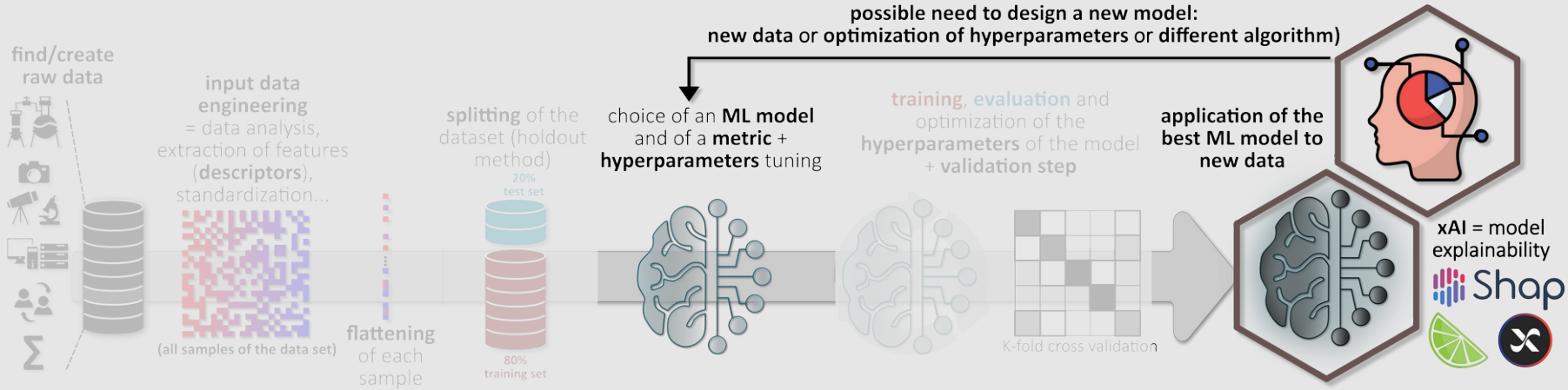
Generalizability?

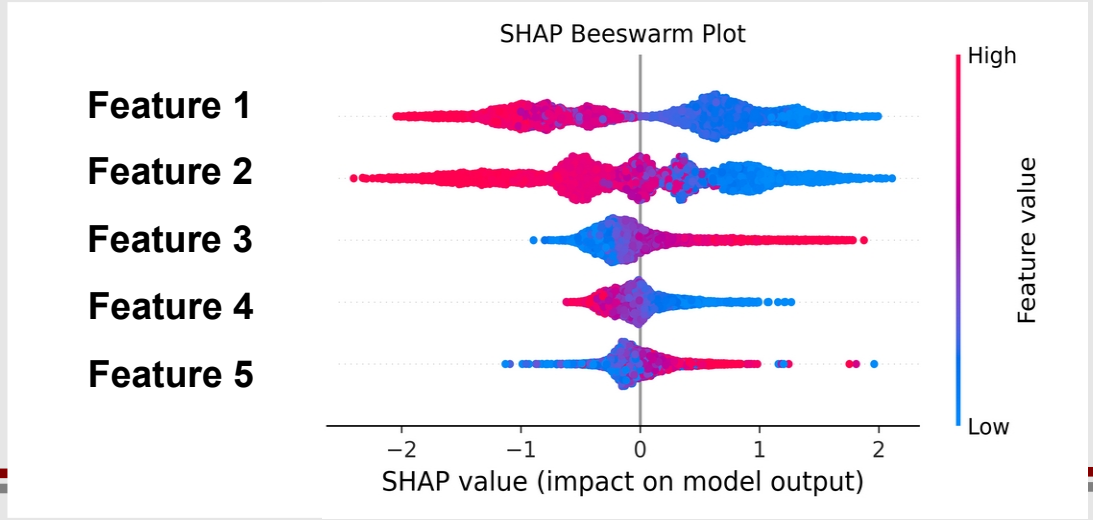
Extrapolation

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**

github repository

Python in the
Physical Chemistry Lab

[pyPhysChem]

https://github.com/rpoteau/pyPhysChem