Machine Learning in Computational Chemistry: A Python and Scikit-learn introduction

Marco Pezzella

marco.pezzella@unipg.it

26.11.2024

Table of Contents

Introduction

Python, Jupyter and Scikit-Learn

Supervised Learning: Regressions

Supervised Learning: decision trees

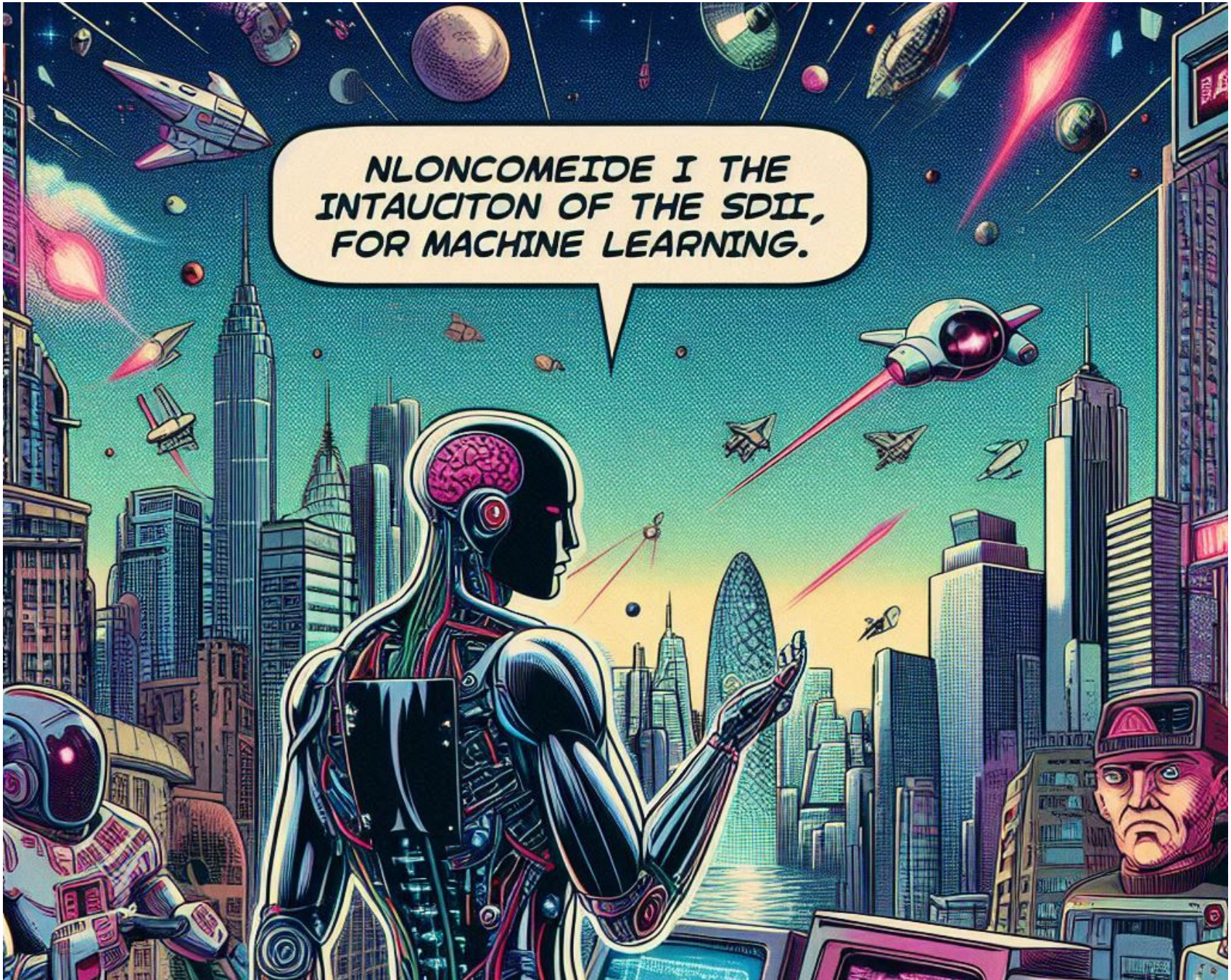
Unsupervised Learning: clustering and dimensionality reduction

Neural Networks

Evaluation Metrics

Few final words

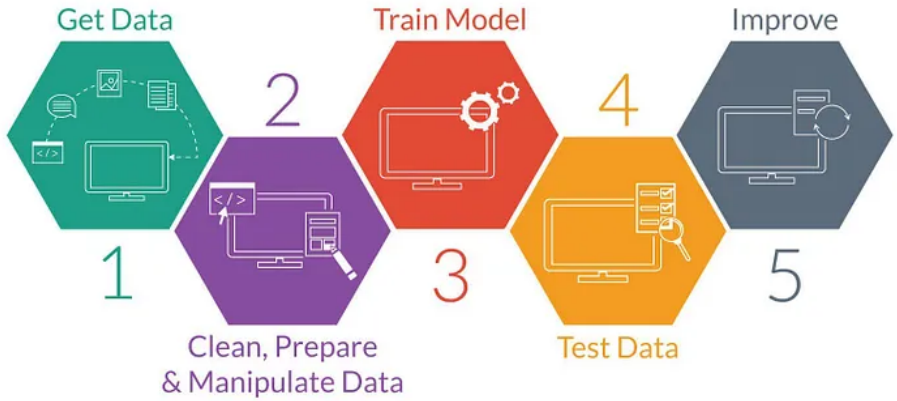
Introduction



Machine Learning

ML is a field of **artificial intelligence** that studies **statistical algorithms** that can learn from data and generalise unseen data.

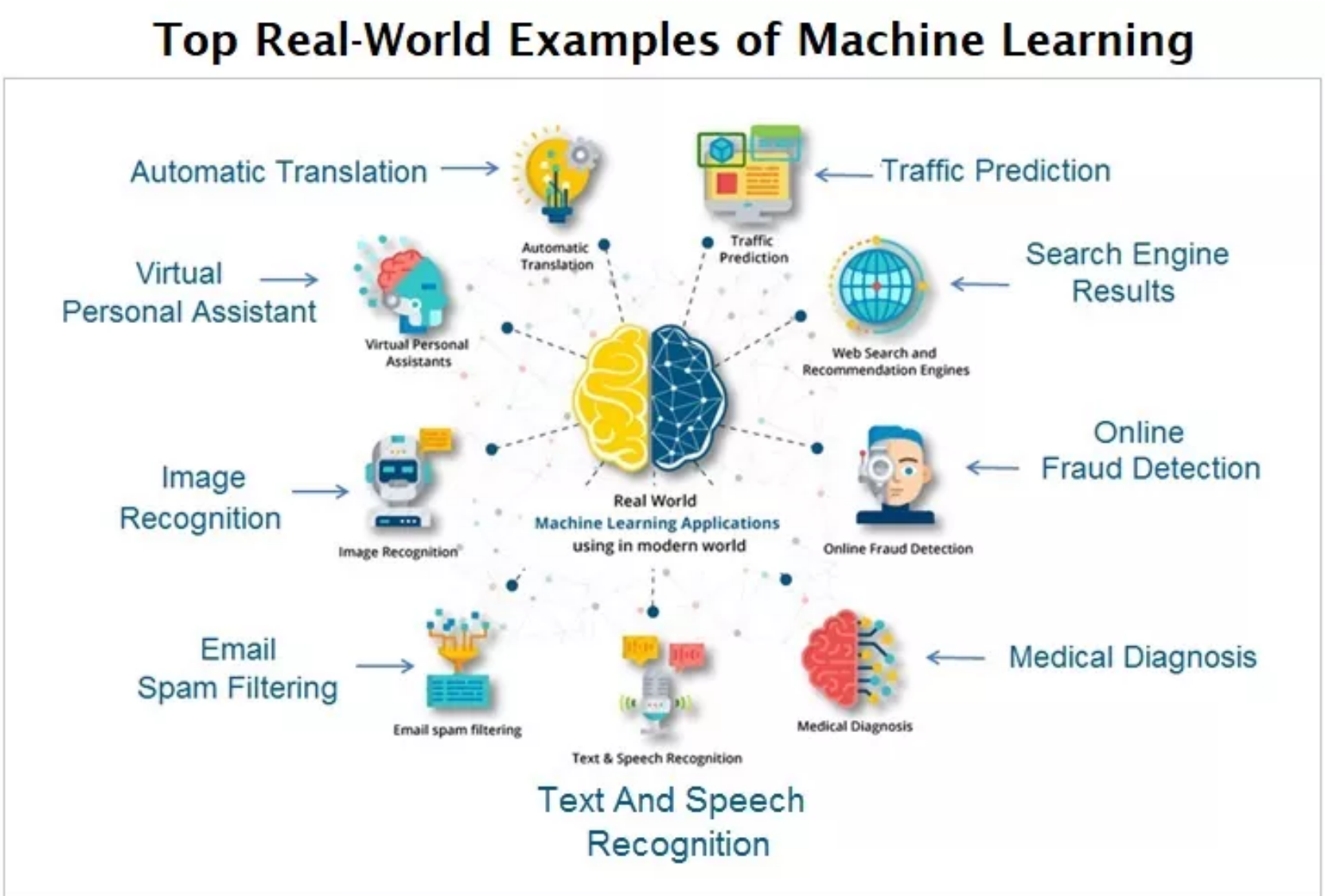
Generalisation is the ability to perform accurately on new, unseen examples/tasks after having experienced a learning data set.



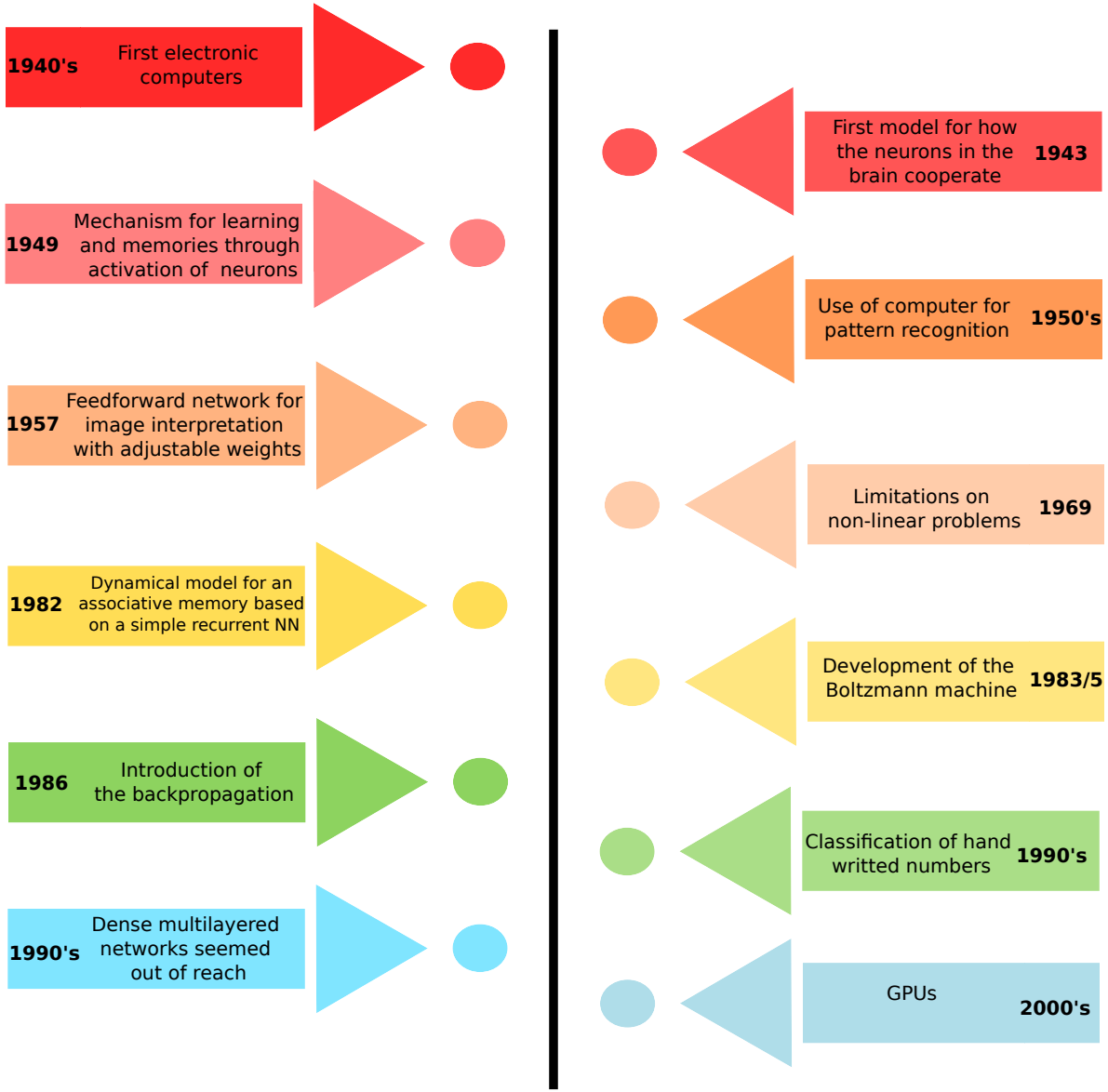
Advances in the field of **deep learning** have allowed **neural networks** to surpass many previous approaches in performance.

Image from Medium

The importance of machine learning



Road to a Nobel prize: Physics



Road to a Nobel prize: Chemistry

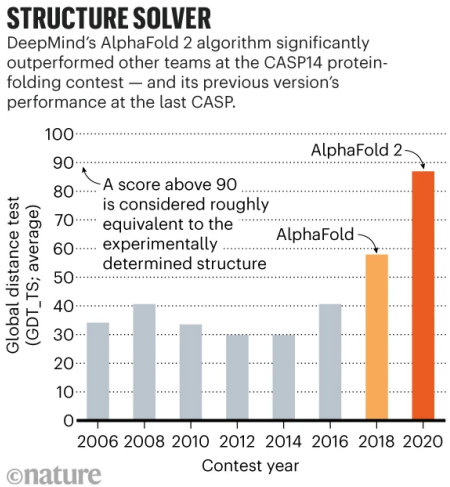
There are 200M sequences in public databases. What are the folding paths?

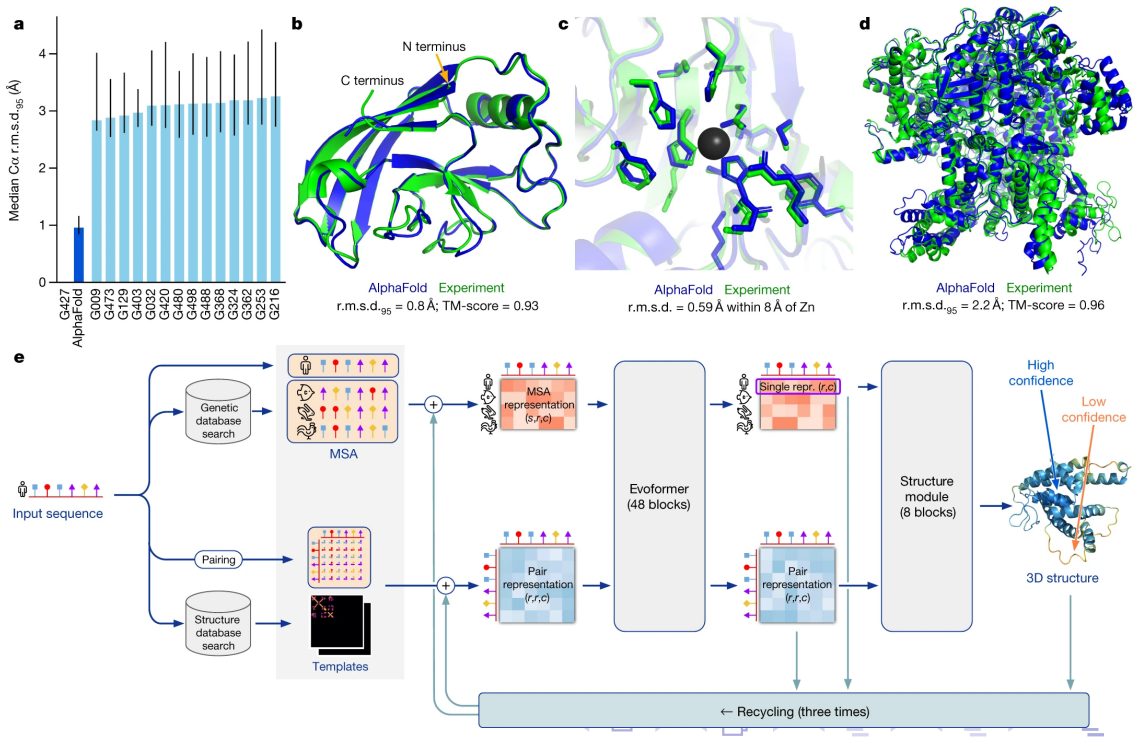
2018: DeepMind constructed AlphaFold1, based on convolutional neural networks. It was trained on Protein Data Bank . GDT score of about 60%

Af2 target is the FES: There is a direct connection to the physical principles.

2020: AlphaFold2 presents an accuracy competitive with experimental structures. The GDT score of about 90%. Backbone accuracy for monomers of about 1 Å.

AF2 source code was made public. A similar structure is adopted in the RoseTTAFold program.

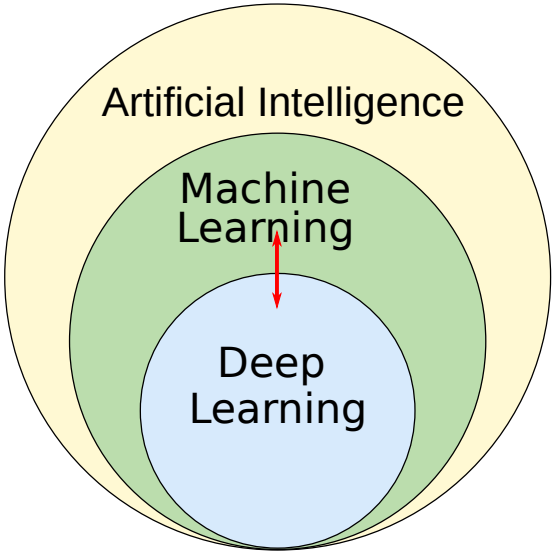


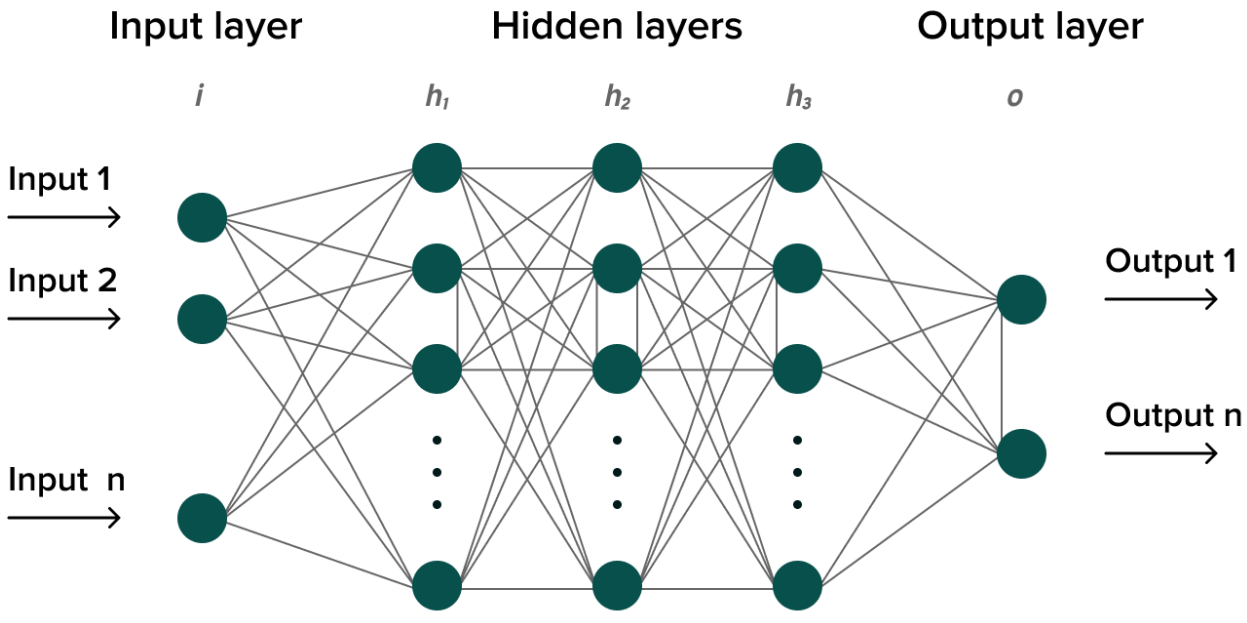


Some terminology I

Artificial intelligence refers to the development of computer systems that can perform tasks designed to analyse data, recognize patterns, and make decisions based on their inputs.

Deep learning is a type of ML that uses multi-layered neural networks to process data in ways that **mimic human cognitive functions**. It enables the learning from vast amounts of **unstructured data**, allowing them to perform tasks **without explicit programming** for each specific task.





Some terminology II

Learning

Learning refers to the process by which algorithms improve their performance on a specific task through experience. It involves adjusting model parameters to minimize prediction errors.

Classification

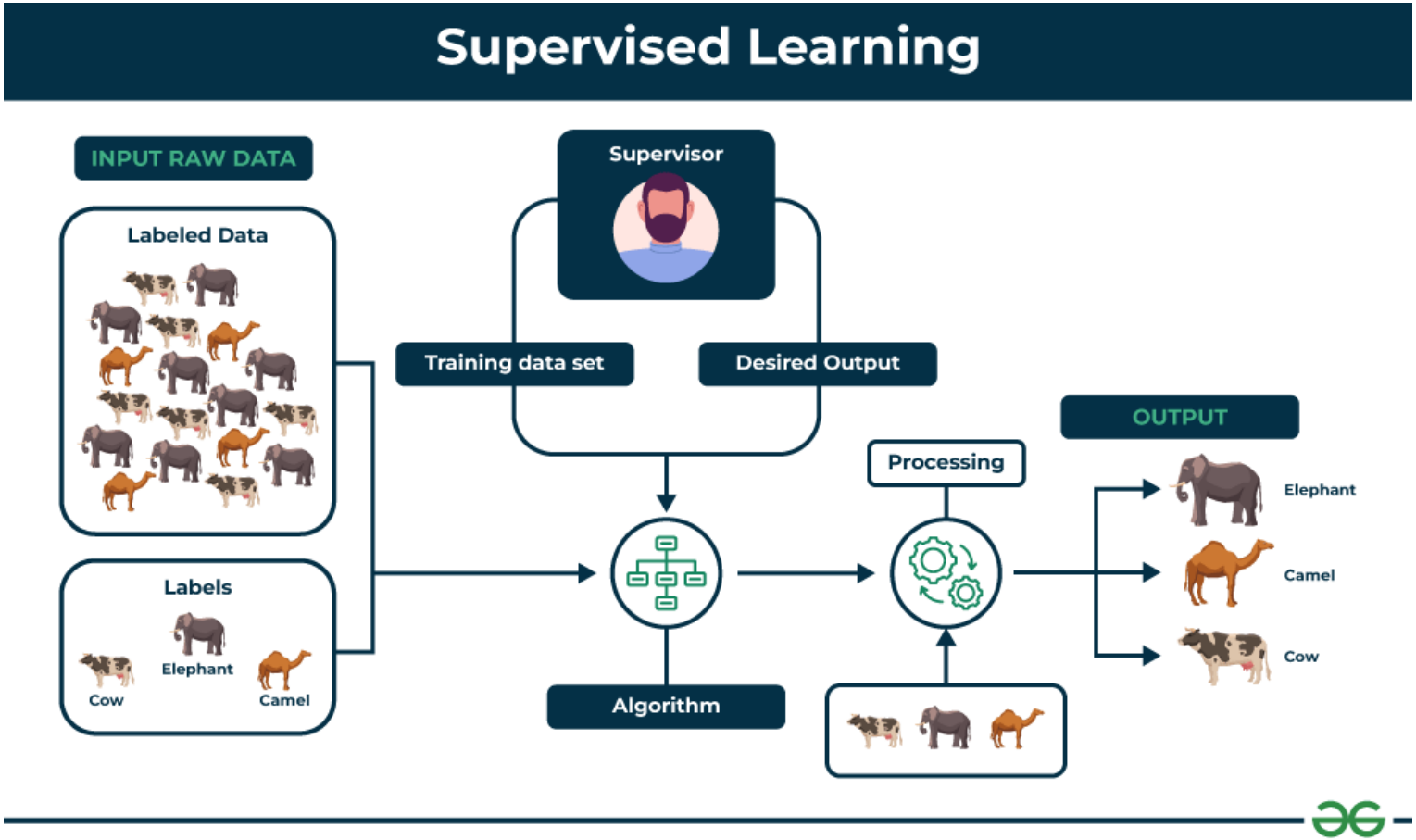
Classification is a task where the model predicts discrete labels or categories for given input data.

Regression

Regression is task where the model predicts continuous values based on input data.

Supervised Learning

Paradigm where **inputs** and a **desired output** values **train a model**. The algorithm generalises from the training data.

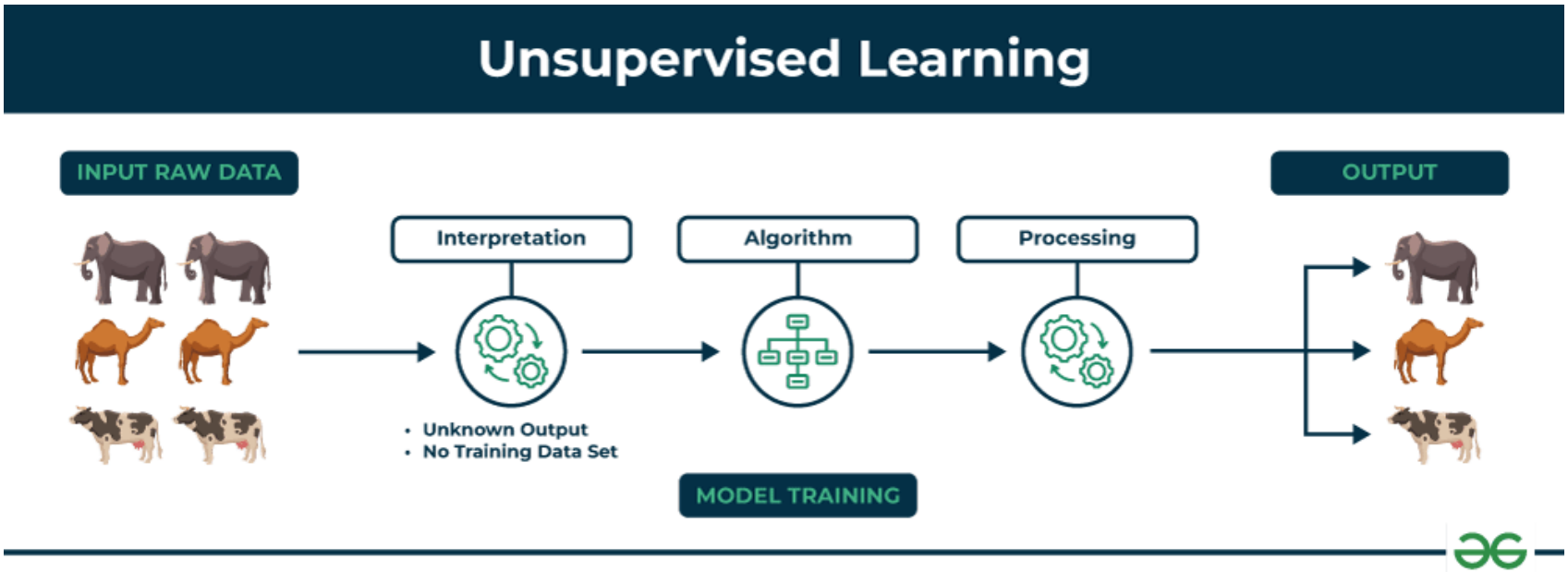


- Classification problems.
- Energy predictions.

Figure from geeksforgeeks.org

Unsupervised Learning

Framework in machine learning where algorithms learn patterns exclusively from **unlabelled** data.

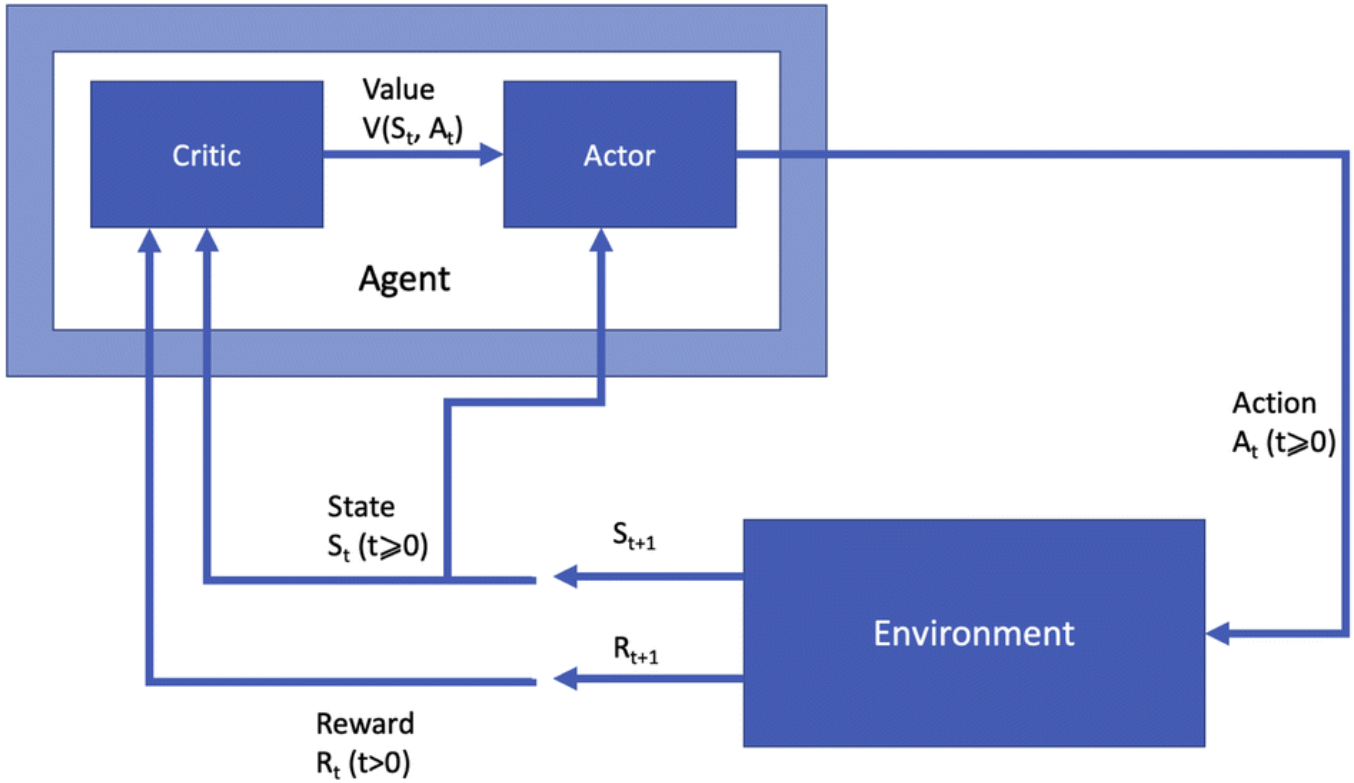


- Clustering of data
- Dimensionality reduction

Figure from geeksforgeeks.org

Reinforcement Learning

Area of machine learning and optimal control concerned with how an **intelligent agent** should take actions in a dynamic environment in order to **maximize a reward signal**.



- Sequencing.
- Reactivity.

Figure from Gow *et al.*

Python, Jupyter and Scikit-Learn

Python

Python is a high-level, general-purpose programming language. It emphasizes code readability with the use of significant indentation. It supports multiple programming paradigms.

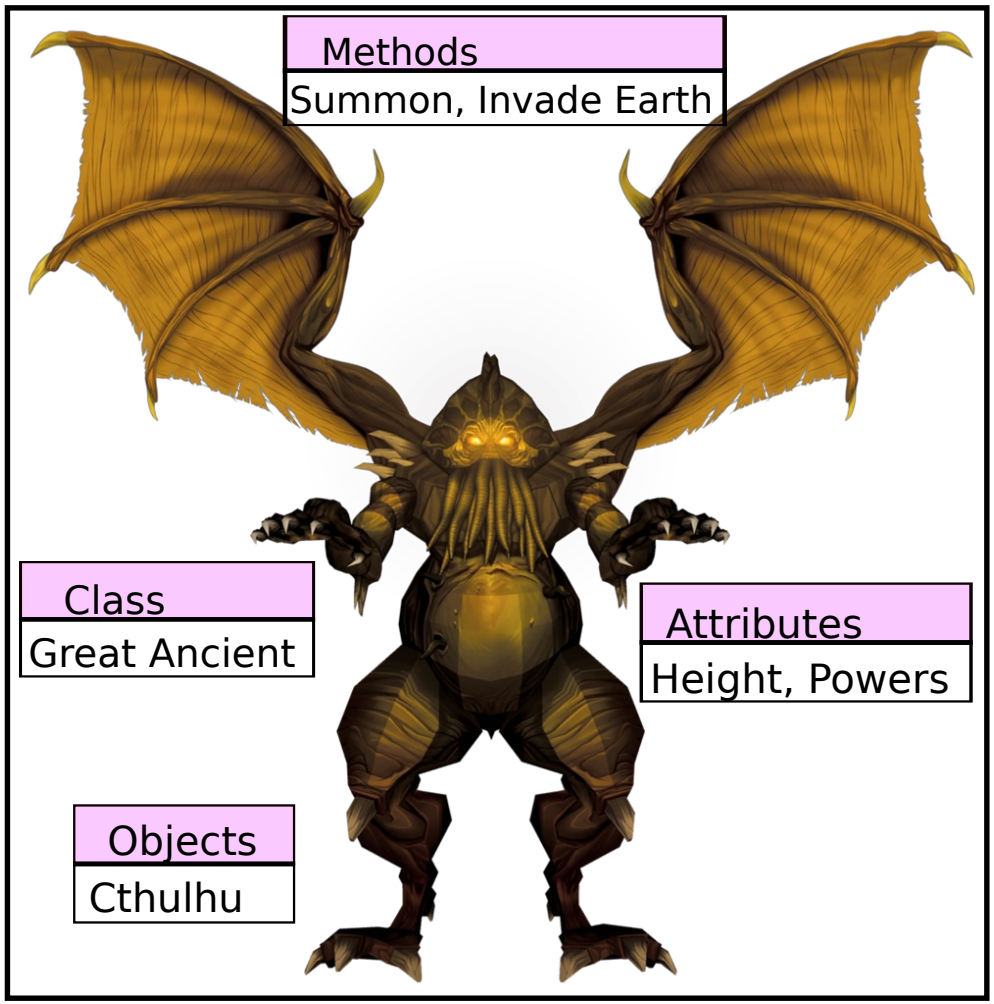
Python 3 is the standard, substituting Python 2.7 (migration started in 2015 and finished in 2020). Updates are released for version 3.9 and onward.

Python

```
class Calculator:
    def add(self, a, b):
        return a + b

    def subtract(self, a, b):
        return a - b


calc = Calculator()
x = 10
y = 5
print("Add:", calc.add(x, y))
print("Subtract:", calc.subtract(x, y))
```

Object-oriented programming

Object-oriented programming (OOP) is a programming paradigm based on the concept of objects, which can contain data and code.



Copyright: South Park, Comedy Central

An object is a data structure or abstract data type containing **fields** and **methods**. Objects are typically stored as contiguous regions of memory. They are accessed as variables with **complex internal structures**.

Classes

It is a **blueprint** for creating objects.

- **Class Definition**.
- **Attributes:** variables.
- **Methods:** functions.
- **Instances:** specific objects created from a class.
- **Inheritance:** attributes and methods taken from other classes.
- **Encapsulation:** restriction of access to certain attributions.
- **Polymorphism:** Different classes can implement methods with the same name.

Functions

It is a reusable block of code that performs a **specific task**.

- **Definition:** This begins with the def keyword, followed by the function name and parentheses.
- **Parameters:** they allow the function to accept input.
- **Docstring:** string that describes what the function does.
- **Body:** actual content of a function.
- **Return Statement:** used to send a value back to the caller.

Libraries

It is a **collection** of precompiled codes, documentations, data, and classes that can be used later on in a program.

It contains bundles of code that can be used repeatedly in different programs, **guaranteeing the reproducibility** of the code.

When linking a library with our code the linker automatically searches for that library. It extracts the functionalities of that library.

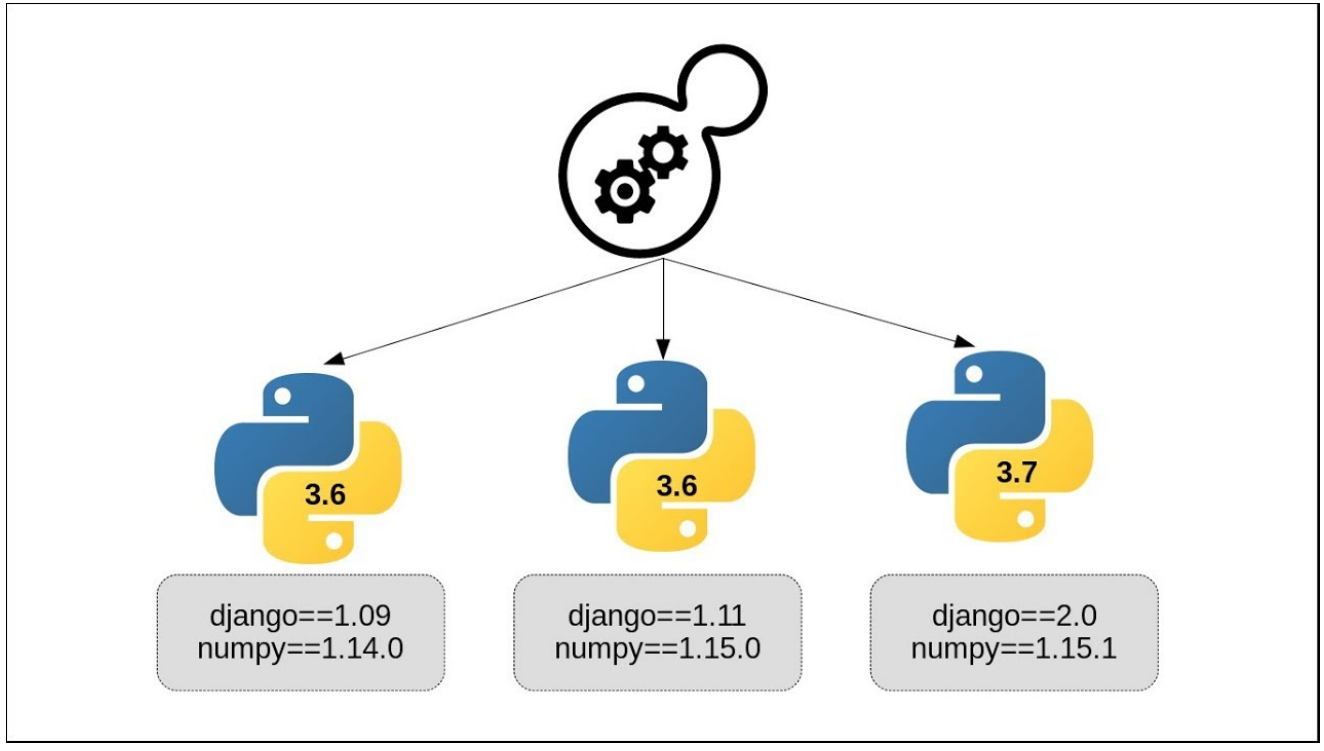
Python - more complex

```python
# mathlib.py

class Calculator:
    def add(self, a, b):
        """Return the sum of a and b."""
        result = a + b  # Store the sum of a and b in result
        return result  # Return the result

    def divide(self, a, b):
        """Return the quotient of a and b. Raises an error if b is
            0."""
        if b == 0:
            raise ValueError("Cannot divide by zero.")  # Handle
                division by zero
        result = a / b  # Store the quotient of a and b in result
        return result  # Return the result
```

```python
# main.py
from mathlib import Calculator  # Importing the Calculator class
    from mathlib

calc = Calculator()  # Creating an instance of the Calculator class

# Variables for calculation
num1 = 5  # First number for calculations
num2 = 3  # Second number for calculations

# Performing calculations using variables
addition_result = calc.add(num1, num2)  # Store the result of
    addition
division_result = calc.divide(num1, num2)  # Store the result of
    division
```

Virtual Environments

It is a directory with a particular file structure. It has a *bin* subdirectory as well as subdirectories that hold packages installed in the specific *venv*.



Virtual environments let you have a **stable**, **reproducible**, and **portable environment**. You are in control of which packages versions are installed and when they are upgraded.

Conda

Virtual environments can be created with *venv* (default in Python) or *conda* (external manager).

venv: lightwight solution but issues with external compatibilities

conda: Robust and cross-languange manager, heavier.

Conda can be installed with **Anaconda** or **Miniconda**.

Anaconda is heavier, on the order of few GB, and comes with GUI and some packages. **Miniconda** is 10 times lighter, with no packages or GUI included. The latter is more customisable, suitable for experienced users.

Jupyter Notebooks and IDE

An **Integrated Development Environment** is a comprehensive software application that provides tools for software development in a single interface.

Jupyter Notebook is one of the most used. It is an open-source web application that allows the creation documents containing live code, equations, visualizations, and text. **Google Colab** is a cloud version of jupyter Notebook.

Alternatives are **Microsoft Visual Studio Code**, Pycharm, Spyder...

Why Machine Learning with Python?

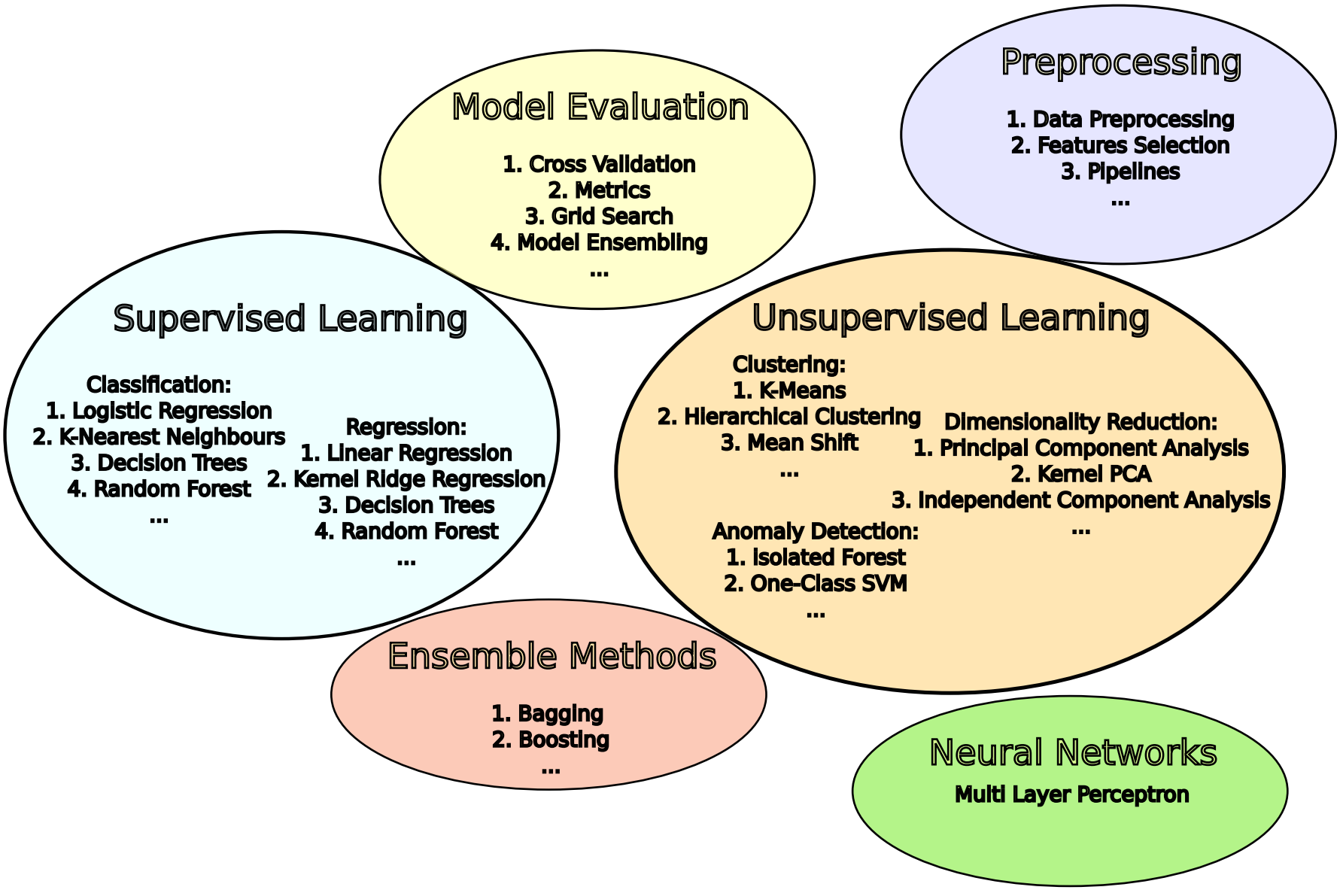
I. Simple and readable syntax.
II. A lot of pre-existing libraries.
III. Flexibility (in tools and in way of programming).
IV. Platform independence (OS, architecture...).
V. (whispering) Google

Sckit-Learn

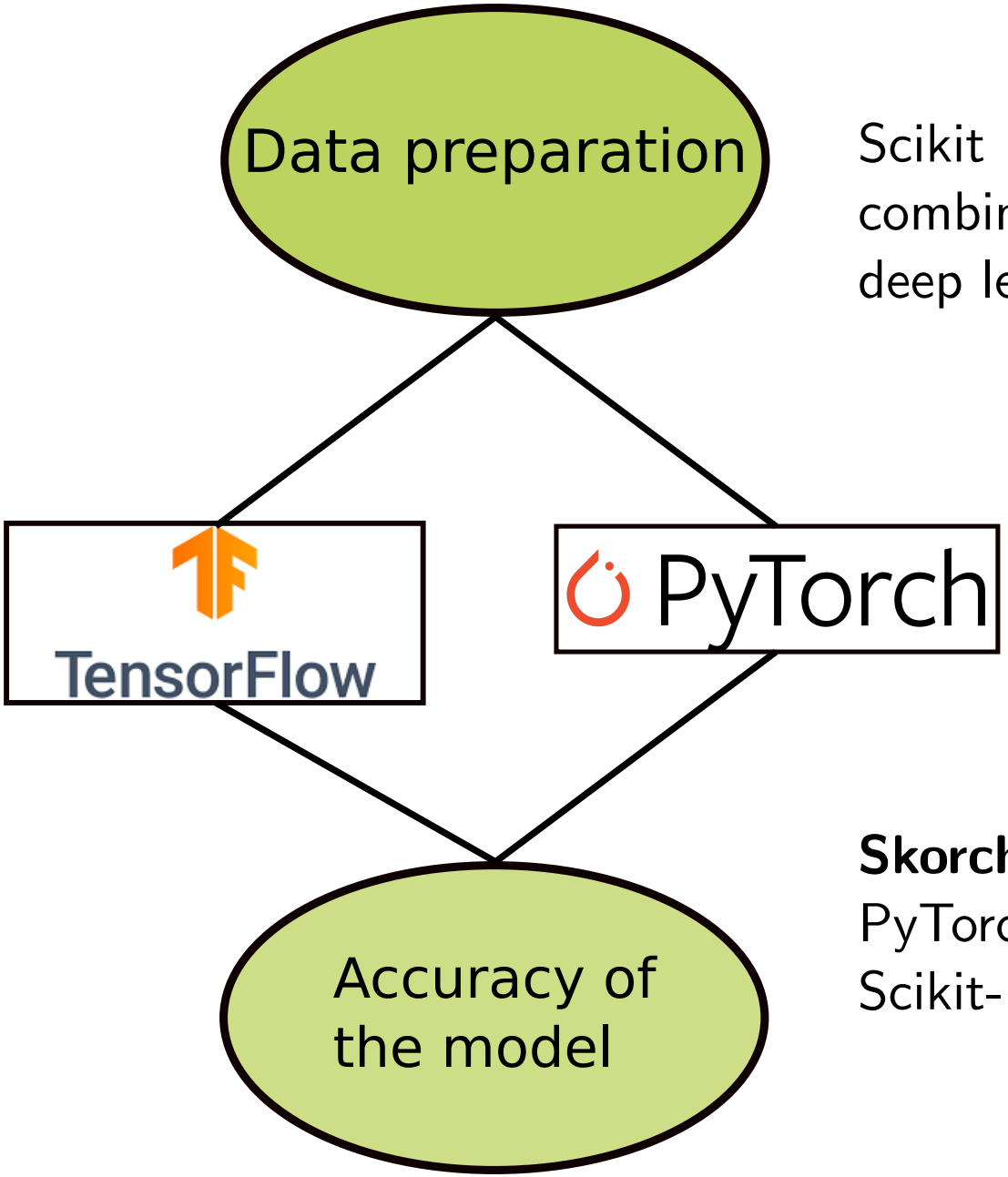
It is a free and open-source machine learning library for Python. The beta version was released in 2010, with the v1.5.2 available now.

It is largely written in Python, and uses NumPy for linear algebra and array operations; some core algorithms are written in Cython to improve performances. Other libraries (Matplotlib, Plotly, Pandas, Scipy...) are integrated in it as well.

Scikit-Learn capabilities



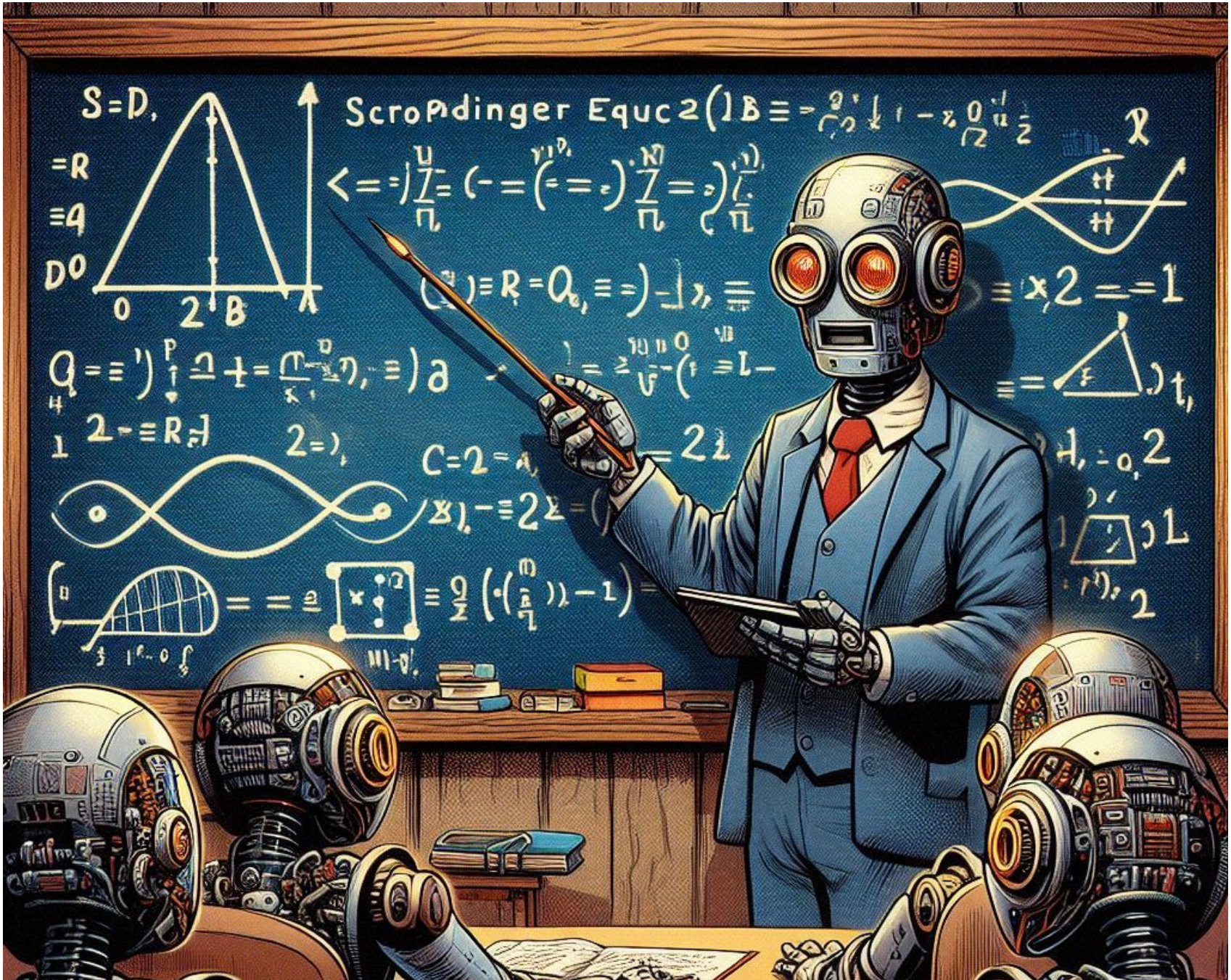
Scikit-Learn with other libraries



Scikit Learn can be used in combination with other libraries for deep learning tasks.

Skorch is the bridge between PyTorch's neural networks and Scikit-Learn's API.

Supervised Learning: Regressions



Linear Regression

It is a model that estimates the linear relationship between a scalar response and one or more independent variable).

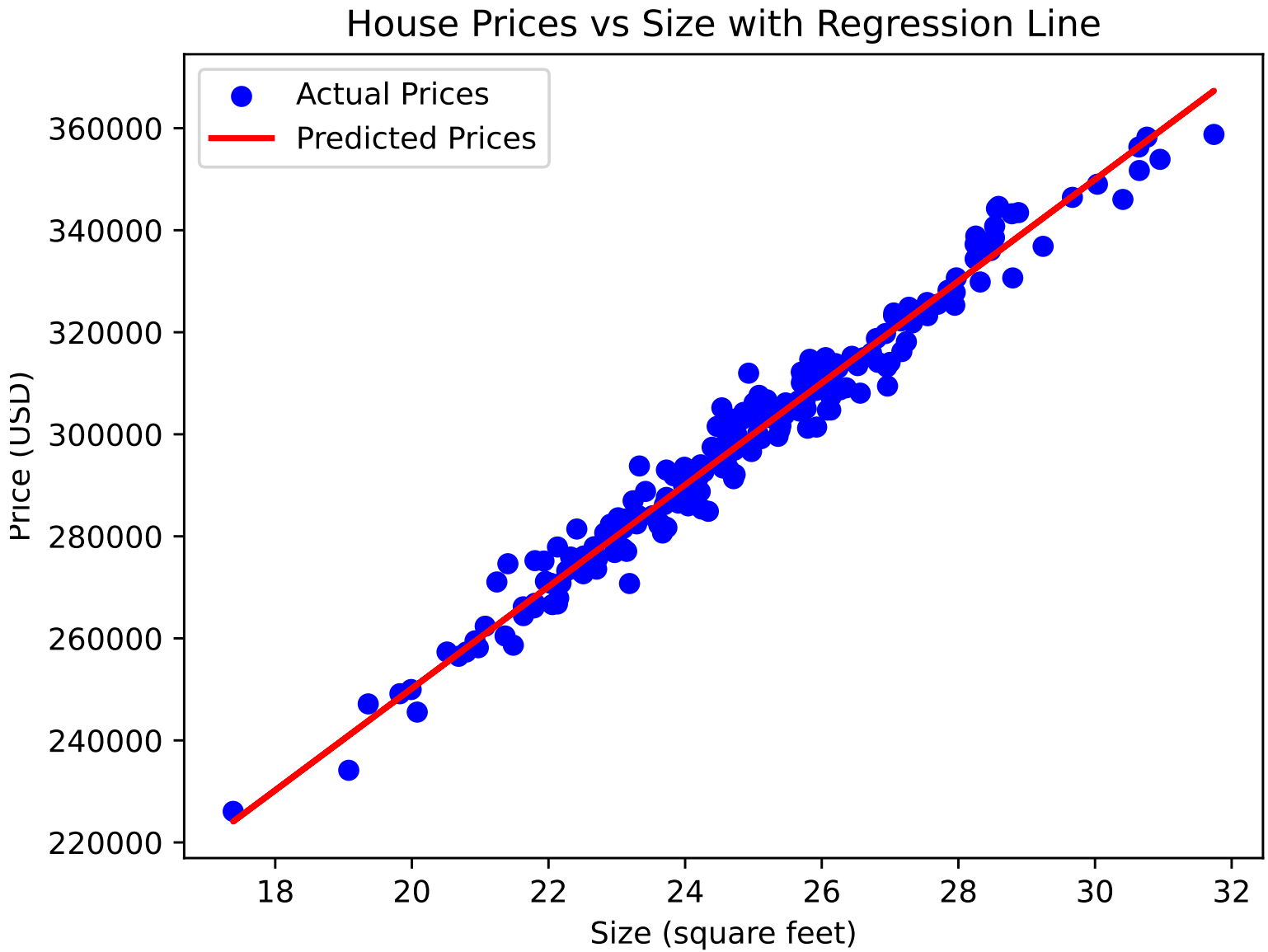
One dimensional: $Y = a + bX + \epsilon$

$$b = \frac{n\sum_i x_i y_i - (\sum_i x_i)(\sum_i y_i)}{n\sum_i x_i^2 - (\sum x_i)^2} \qquad a = \frac{\sum_i y_i - b(\sum_i x_i)}{n}$$

Multiple dimension: $Y = a + \sum_i^n b_i X_i + \epsilon$ or $\mathbf{Y} = \mathbf{Xb} + \epsilon$

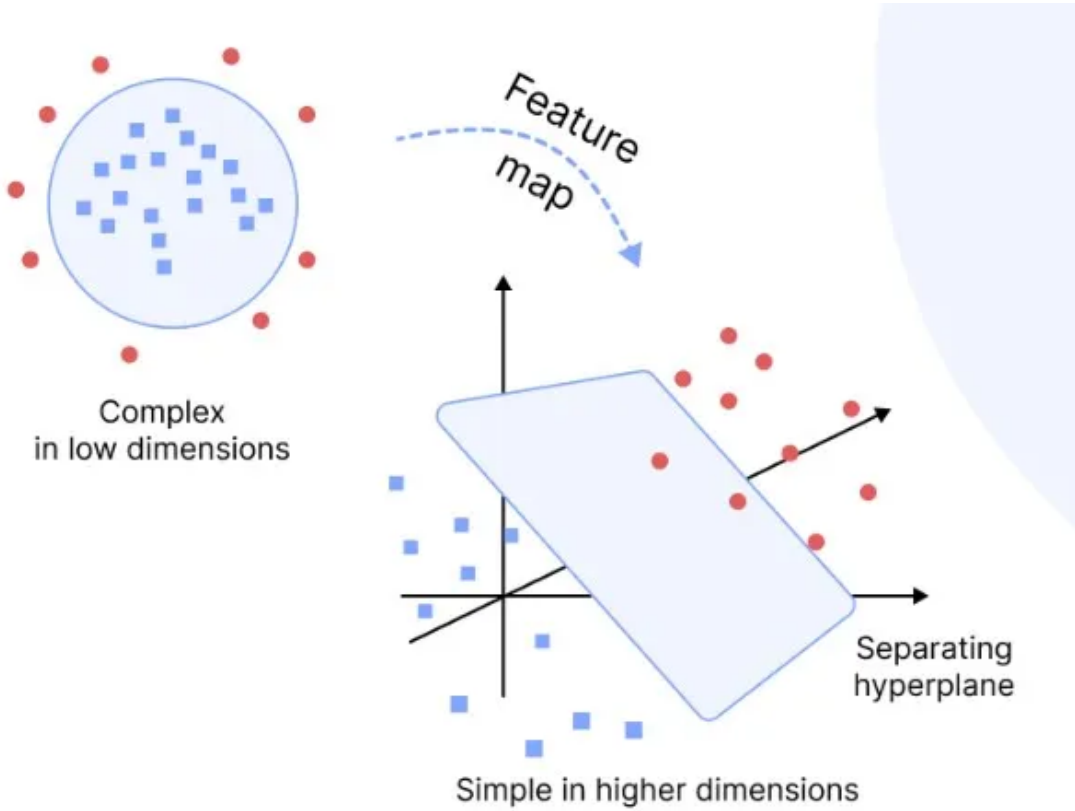
- Y: scalar response.
- X: independent variable.
- a: y-intercept.
- b: slope.
- ϵ: error term.
- n: is the number of data points.
- x, y: individual points.

Example: Linear Regression



Kernel methods

Kernel methods are a class of algorithms in ML that enable the transformation of data into higher-dimensional spaces, allowing for the handling of nonlinear relationships.



The kernel trick

This technique allows algorithms to operate in high-dimensional feature spaces without explicitly computing the coordinates in those spaces.

The kernel functions compute the inner products between pairs of data points.

$$K(x, y) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle_V$$

$$\varphi : V \to H$$

- V: input space of $\mathbf{x}$ and $\mathbf{y}$.
- H: high-dimensional Hilbert space.
- φ: feature map.
- K: kernel function.

The Kernel

For a kernel function to be valid, it must satisfy Mercer's theorem: for any finite subset of points $x_1, x_2, \ldots, x_n$ in V, and any coefficients $c_1, c_2, \ldots, c_n$:

$$\sum_{i,j} K(x_i, x_j) c_i c_j \geq 0$$

This condition ensures that the kernel function is positive semi-definite:

I Convergence and accuracy.

II Validity of the representer theorem.

III Expansion of the the kernel as a linar combination of eigenafunctions and eigenweights.

IV Kernel symmetry: $K(x, y) = K(y, x)$

V Non-negative definite kernel functions.

Representer theorem: the solution to kernel-based optimisation problem can be represented as a linear combination of the training data points

Reproducing Kernel Hilbert Space

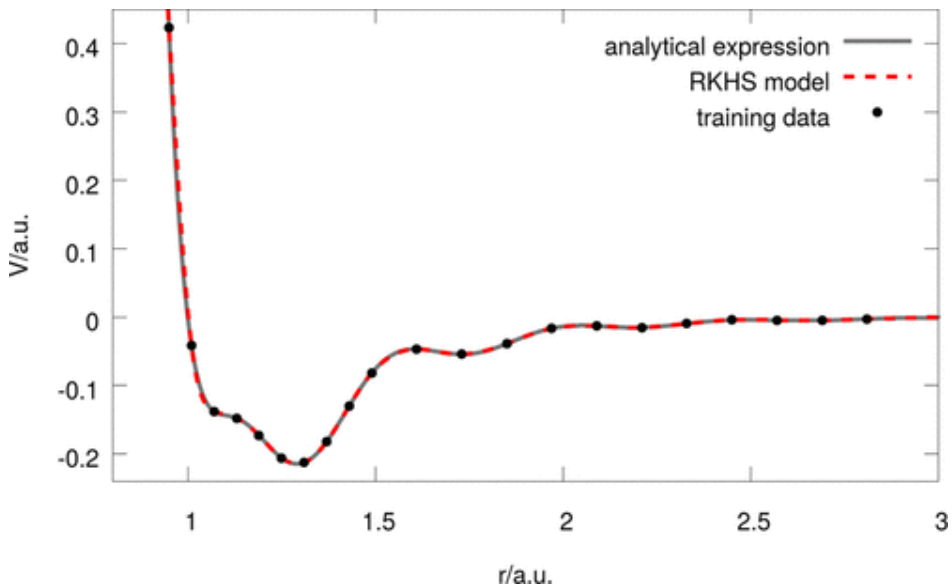
Kernel methods often assume that learning occurs in an RKHS.

It is an high-dimensional Hilbert space of functions in which point evaluation is a continuous linear functional.

For each $x \in V$, there exists a $K_x \in H$ such that for all $\varphi \in H$,

$$\langle \varphi, K_x \rangle_V = \varphi(x)$$

This space is suitable for non-parametric statistics and ML.



$$V(r) = (r^{-9} - r^{-6})(cos^2(7r) + 1)$$

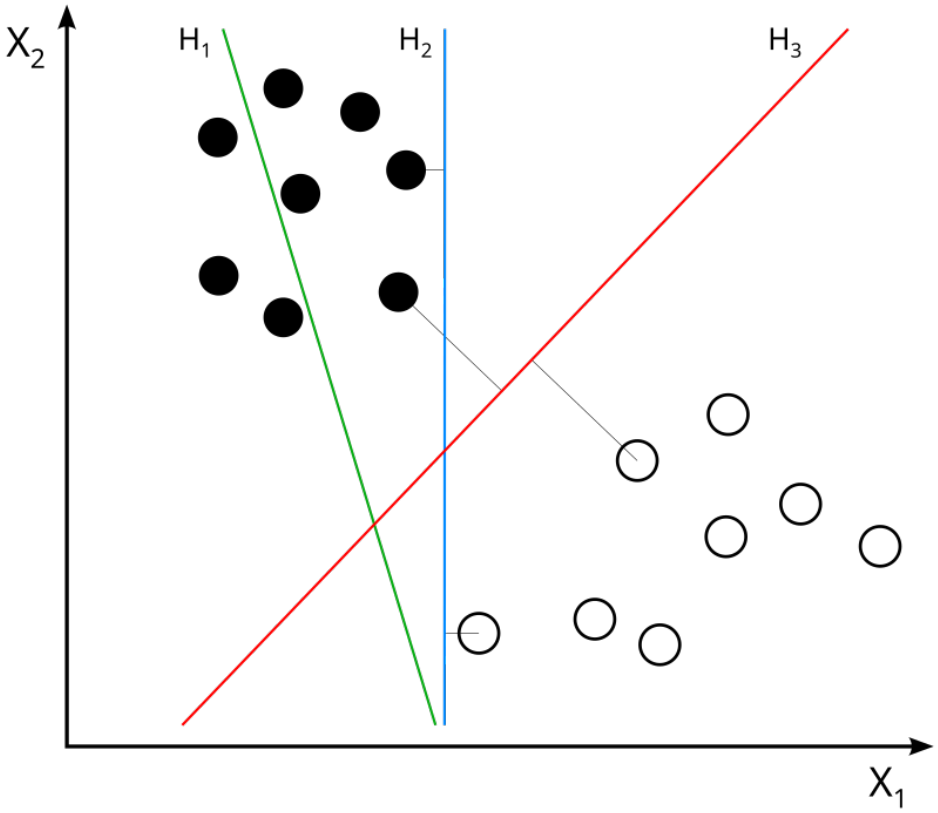
Figure from Unke et al, *J. Chem. Inf. Model.* 2017, 57, 8, 1923–1931

Support Vector Machines

They are supervised models used for classification and regression analysis.

Support vectors are the data points that lie closest to the decision boundary: they define the position and orientation of the hyperplane.

The **hyperplanes** in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space remains constant.

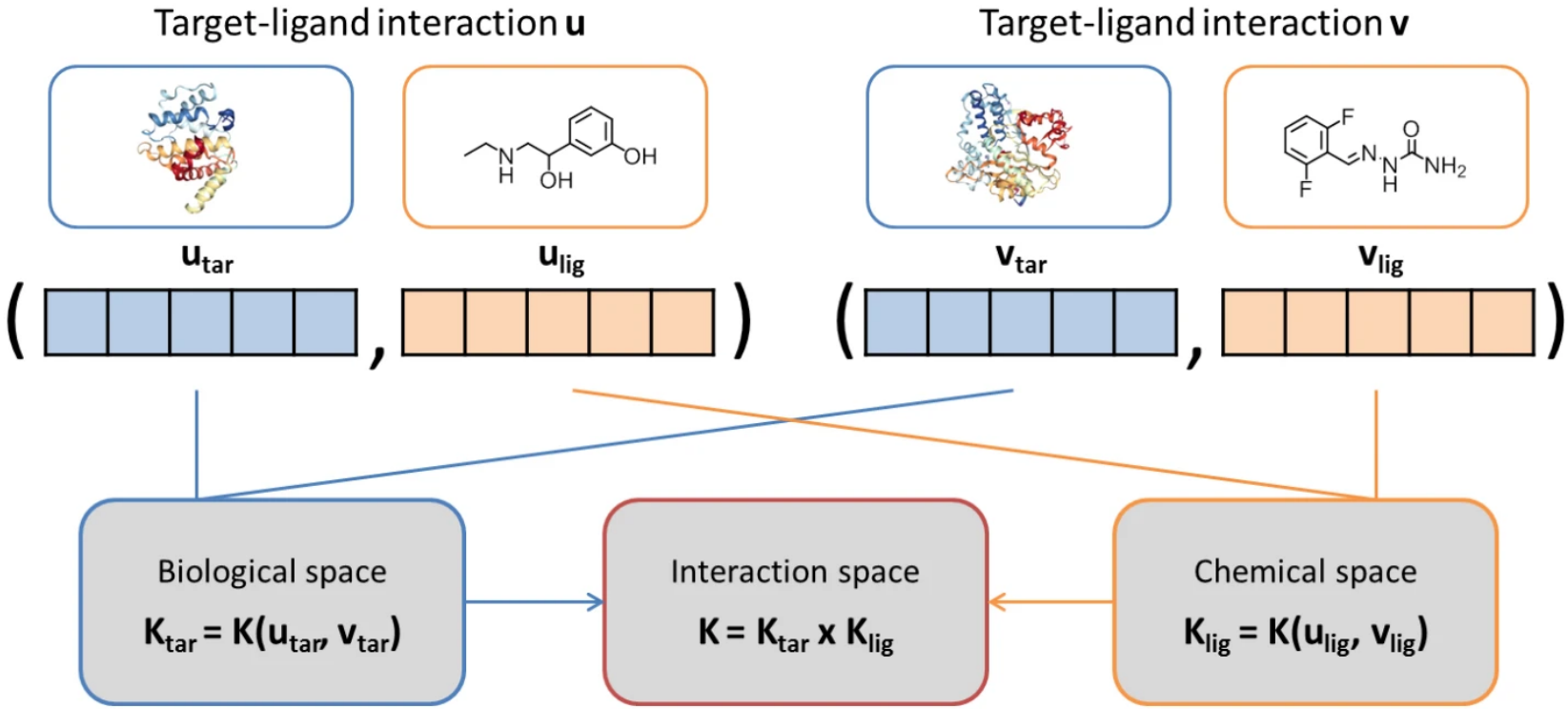


They are chosen so that the distance to the nearest data point on each side is maximised (**maximum-margin hyperplanes**): the larger the margin, the lower the generalisation error of the classifier.

SVM in chemistry

Materials from J. Comput. Aided Mol. Des. (2022) 36:355–362.

SVM and derived methods achieve high accuracy in compound classification, which is essential for virtual compound screening.



Target-ligand interaction: the product kernel of two separate kernels is used for target-ligand pairings yielding a combined similarity score.

Kernel Ridge Regression

KRR combines ridge regression with kernel methods to handle non-linear relationships in data.

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x) + b$$

- K: kernel function evaluated at points x_i.
- x_i: training data points.
- b: bias term.
- α_i: coefficients to be determined.

α_i are found solving the equation:

$$\hat{\alpha} = \arg\min_{\alpha} \left(\sum_{i=1}^{N} (\hat{f}(x_i) - f(x_i))^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \right)$$

KRR in chemistry I

Materials from J. Chem. Phys. 150, 204121 (2019).

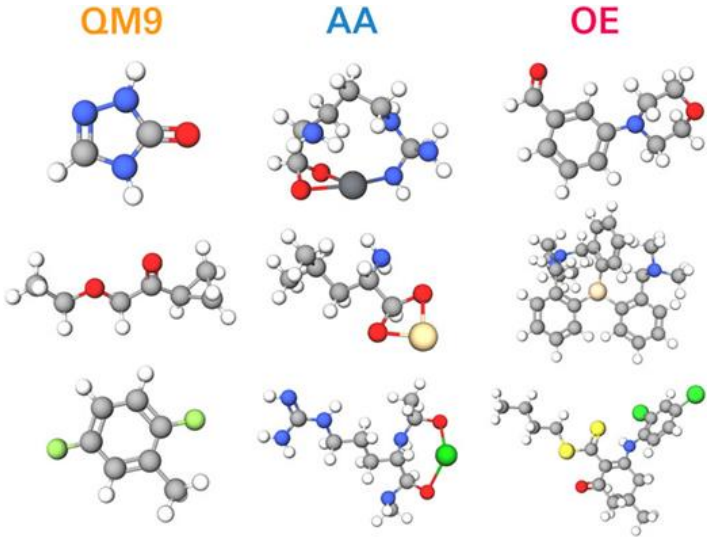
HOMO energy prediction with KRR at DFT level performances are estimated with three datasets.

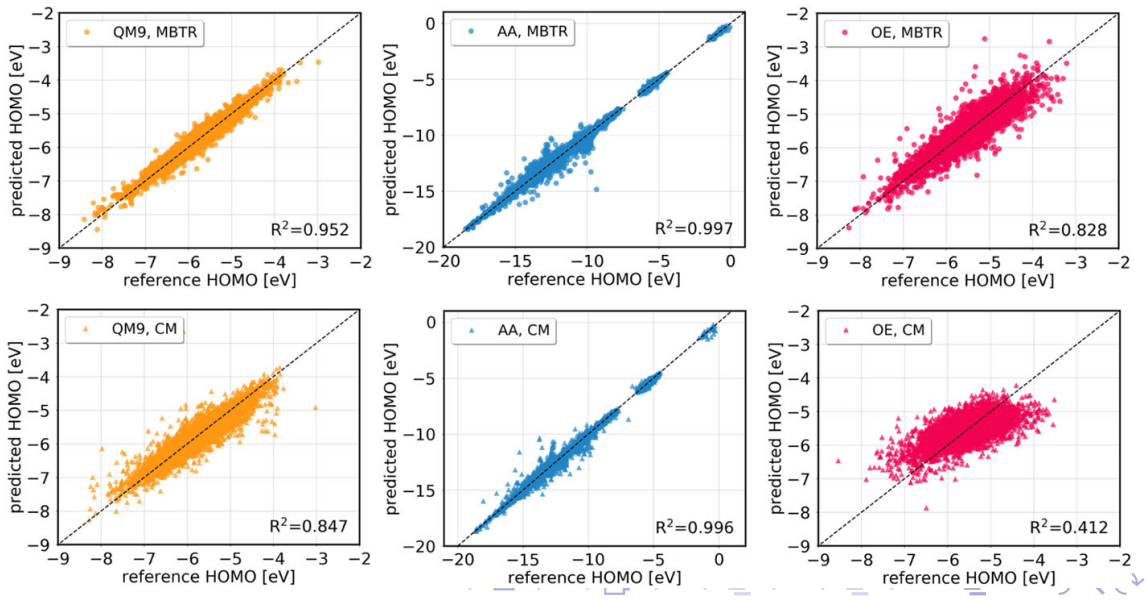
Guassian or **Laplacian** Kernels.

Molecular Representation:

$$C_{ij} = \begin{cases} Z_i Z_j / R_{ij} & \text{if } i \neq j \\ \frac{1}{2} Z_i^2 & \text{if } i = j \end{cases} \qquad g_k(\mathbf{r}) = \begin{cases} f_1(Z_i) \\ f_2(Z_i, Z_j, R_{ij}) \\ f_3(Z_i, Z_j, Z_k, R_{ij}, R_{ik}, \theta_{ijk}) \end{cases}$$

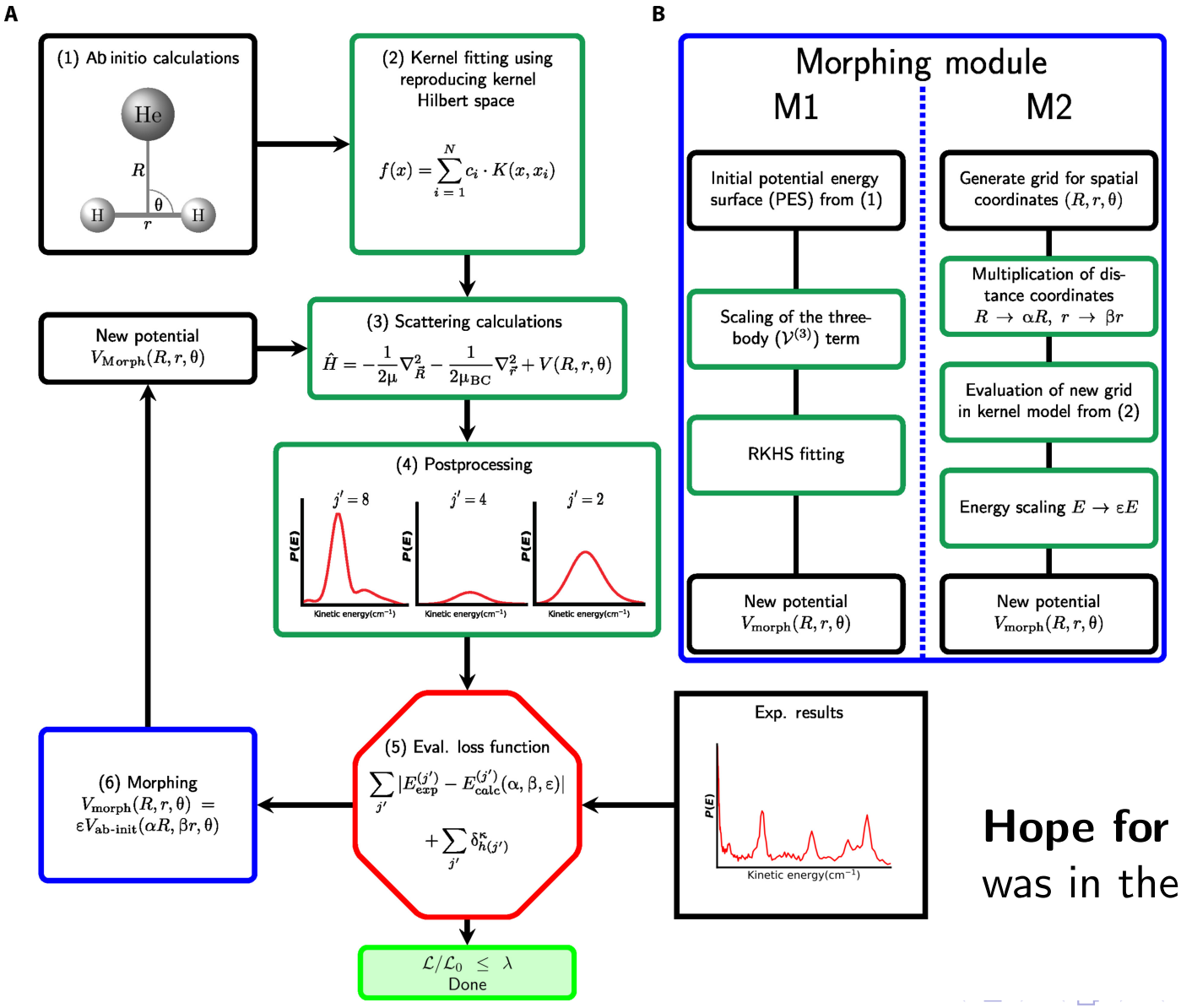
Coulomb Matrix **Many-Body Tensor Representation**





KRR in chemistry II

Materials from Science Advances, Vol 10, Issue 9.



Hope for the future: Luis was in the same your class !

Kernel Density Estimation

KDE is a non-parametric method to estimate the probability density function of a random variable based on kernels as weights. It is used for **smoothing problems of finite size problems**.

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

n: number of points in the distribution.

x_i: sample from some univariate distribution $(x_1, ..., x_n)$.

$\hat{f}_h$: unknown density function to reproduce.

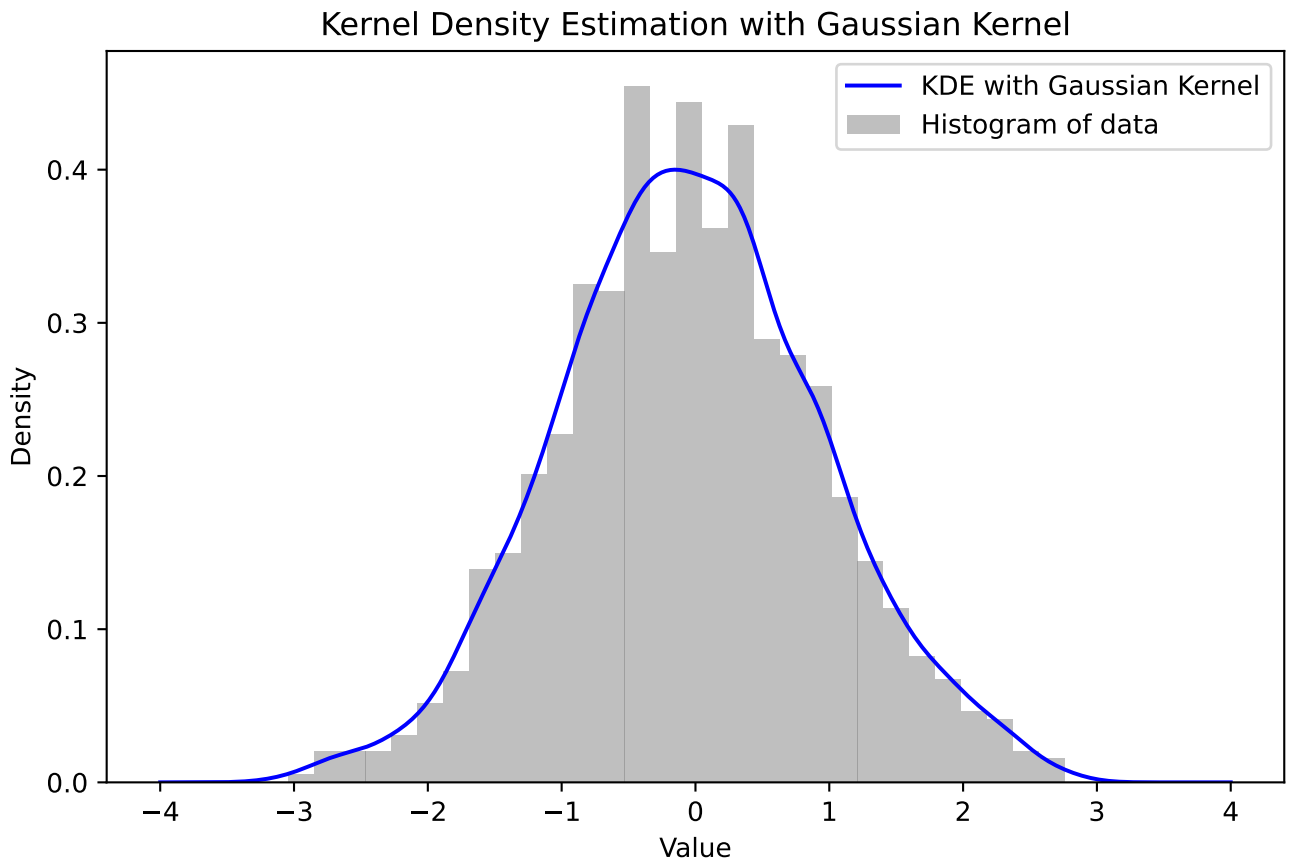
x: point in the original distribution $f(x)$.

h: Smoothing parameter, or *bandwidth*.

K: Kernel function.

Example: Normal Kernel Distribution

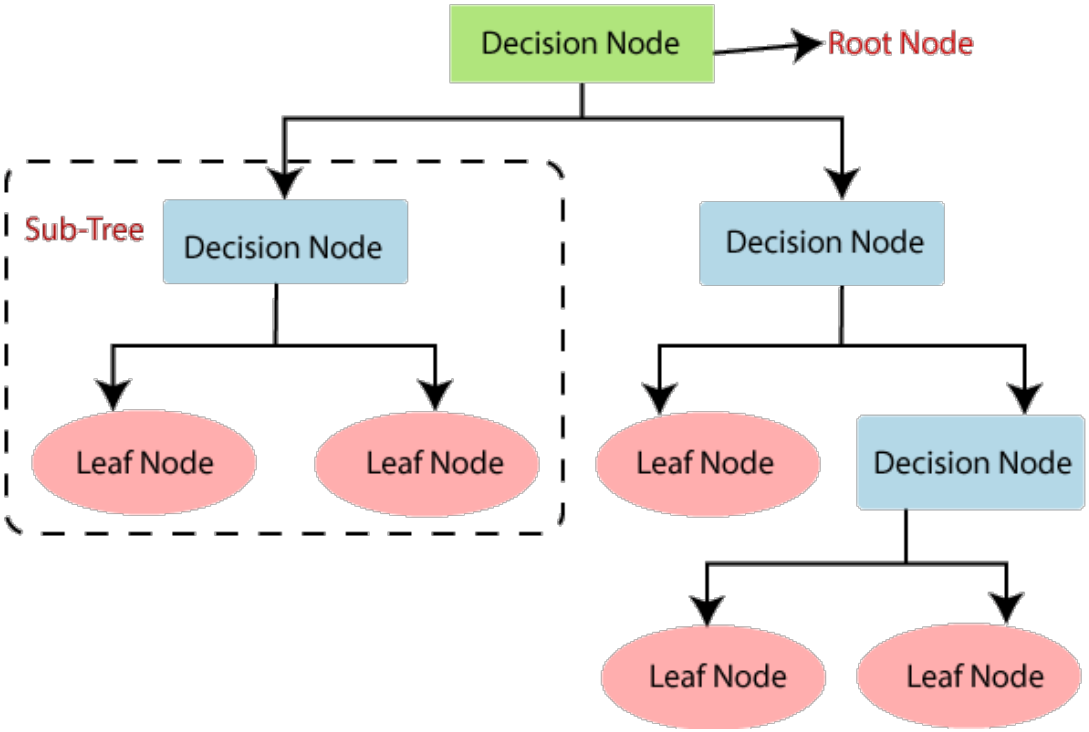
$$\hat{f}_h(x) = \frac{1}{nh\sigma}\frac{1}{\sqrt{2\pi}}\sum_{i=1}^{n}\exp\left(\frac{-(x-x_i)^2}{2h^2\sigma^2}\right),$$



Supervised Learning: decision trees

Decision Tree

They are a non-parametric supervised learning method. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.



Root Node: The starting node, which stands for the complete dataset.

Branch Nodes: Internal nodes that represent decision points.

Leaf Nodes: Final categorization or prediction-representing.

Decision Rules: Rules that govern the splitting of data at each branch node.

Decision Tree

Attribute Selection: The process of choosing the most informative attribute for each split.

Splitting Criteria: Metrics like entropy, or the Gini Index are used to calculate the optimal split.

Entropy: Entropy is a measure of the uncertainty or disorder in a dataset.

$$H(S) = -\sum_{i}^{n} p_i \log_2 p_i$$

S: current dataset.

n: number of classes.

p: number of elements in a class in S.

Gini index: how often a randomly chosen element would be incorrectly labeled if it was randomly labeled.

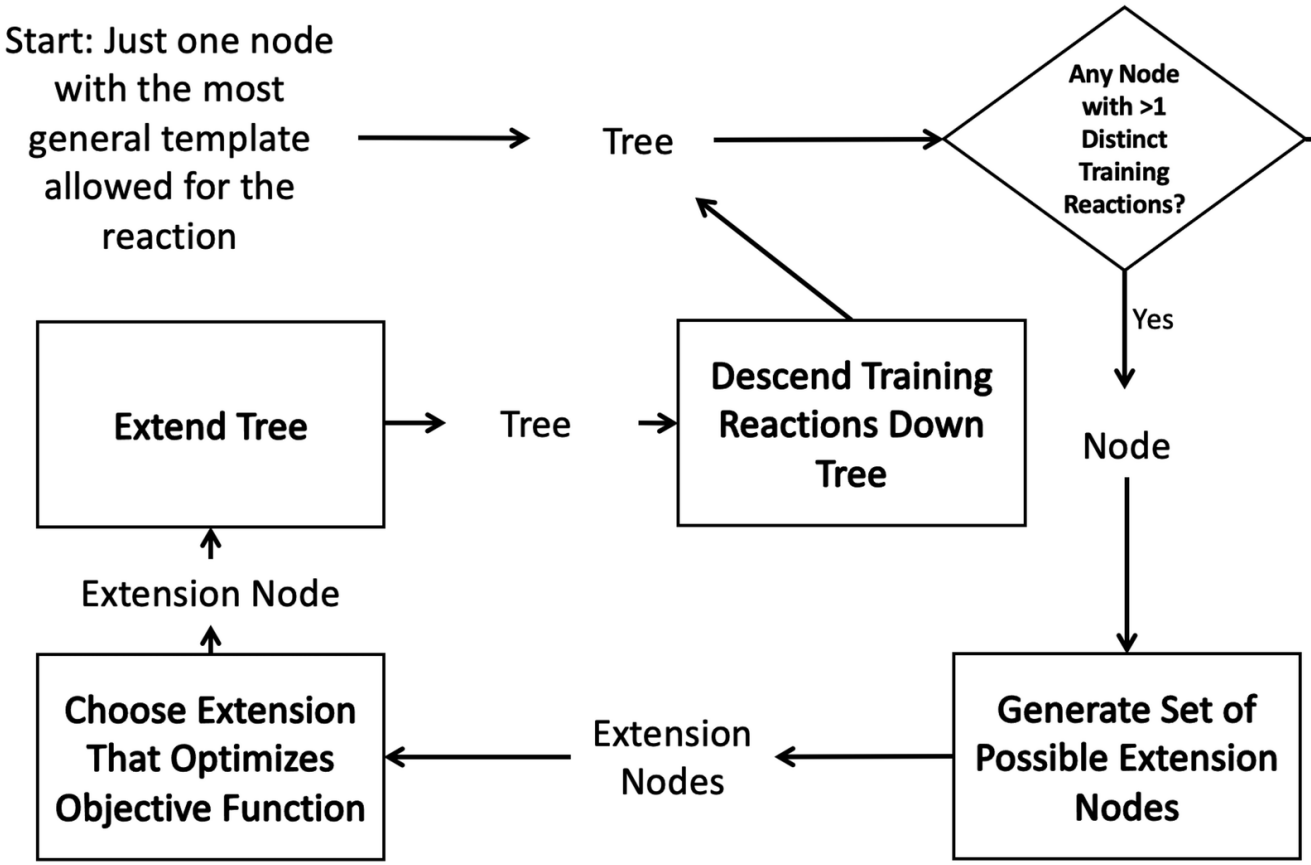
$$G(S) = 1 - \sum_{i}^{n} p_i^2$$

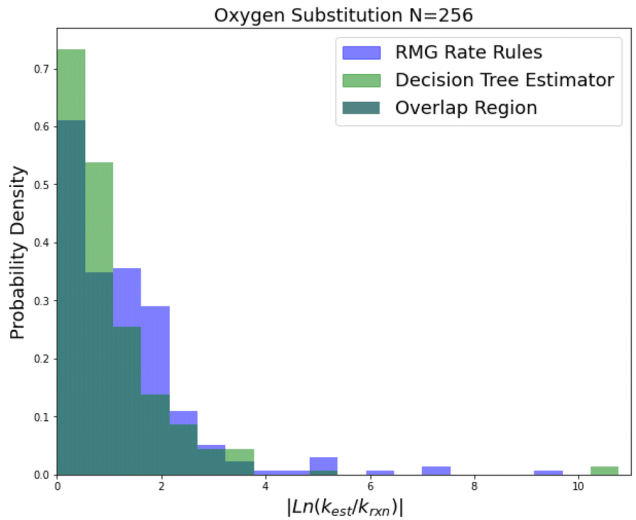
Pro and cons

+ Simple understanding and interpretation.
+ Robust against co-linearity.
+ Easy feature selection.
+ Possibility to deal with large data.
- Small change in the training can result in a large change in the tree.
- Overfitting: mechanisms such as pruning are necessary to avoid this issue.
- Bias in favour of attributes with more levels.

Example of decision trees in chemistry

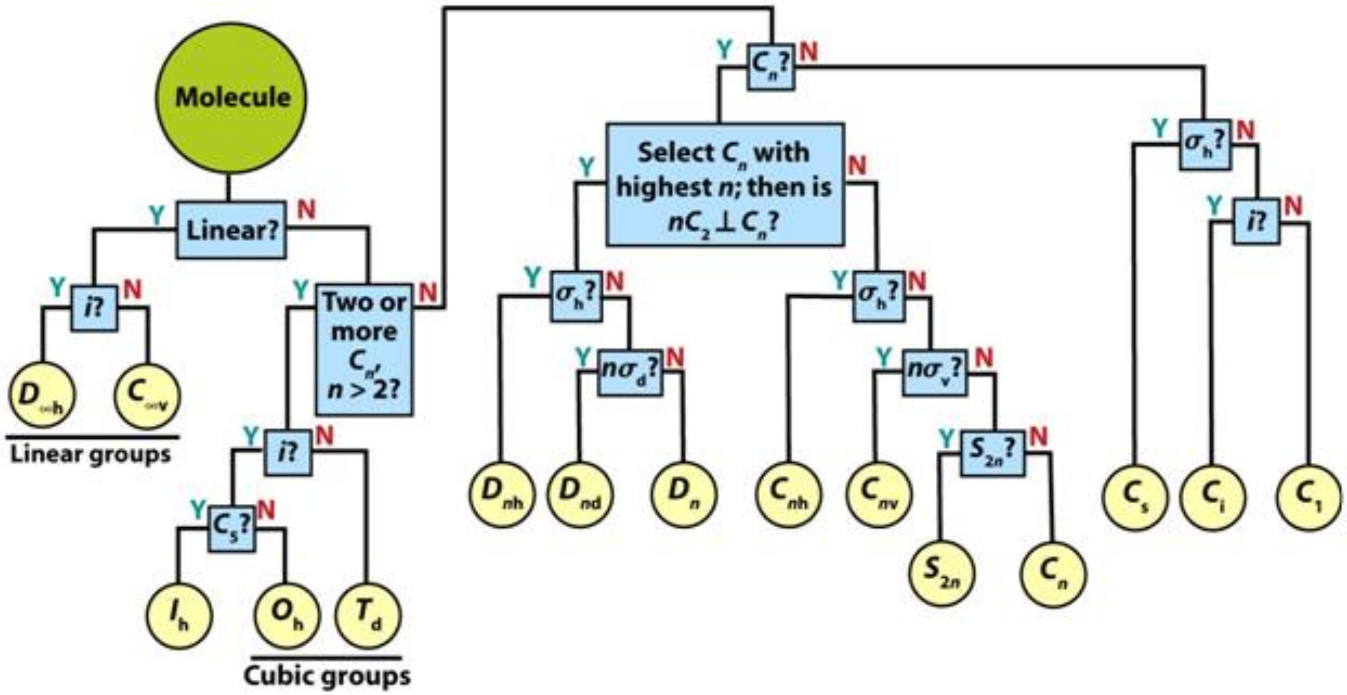
Materials from React. Chem. Eng., 2024, 9, 1364-1380





Sophisticated decision tree models can be used to model site specific reaction rates for molecules!

The most infamous example of decision tree



Whoever did this figure: I hate/love you!

Random Forest

It is an estimator that fits a number of DT on various random sub-samples of the dataset and uses **voting** (classification) or **averaging** (regression) to improve the predictive accuracy and control over-fitting.

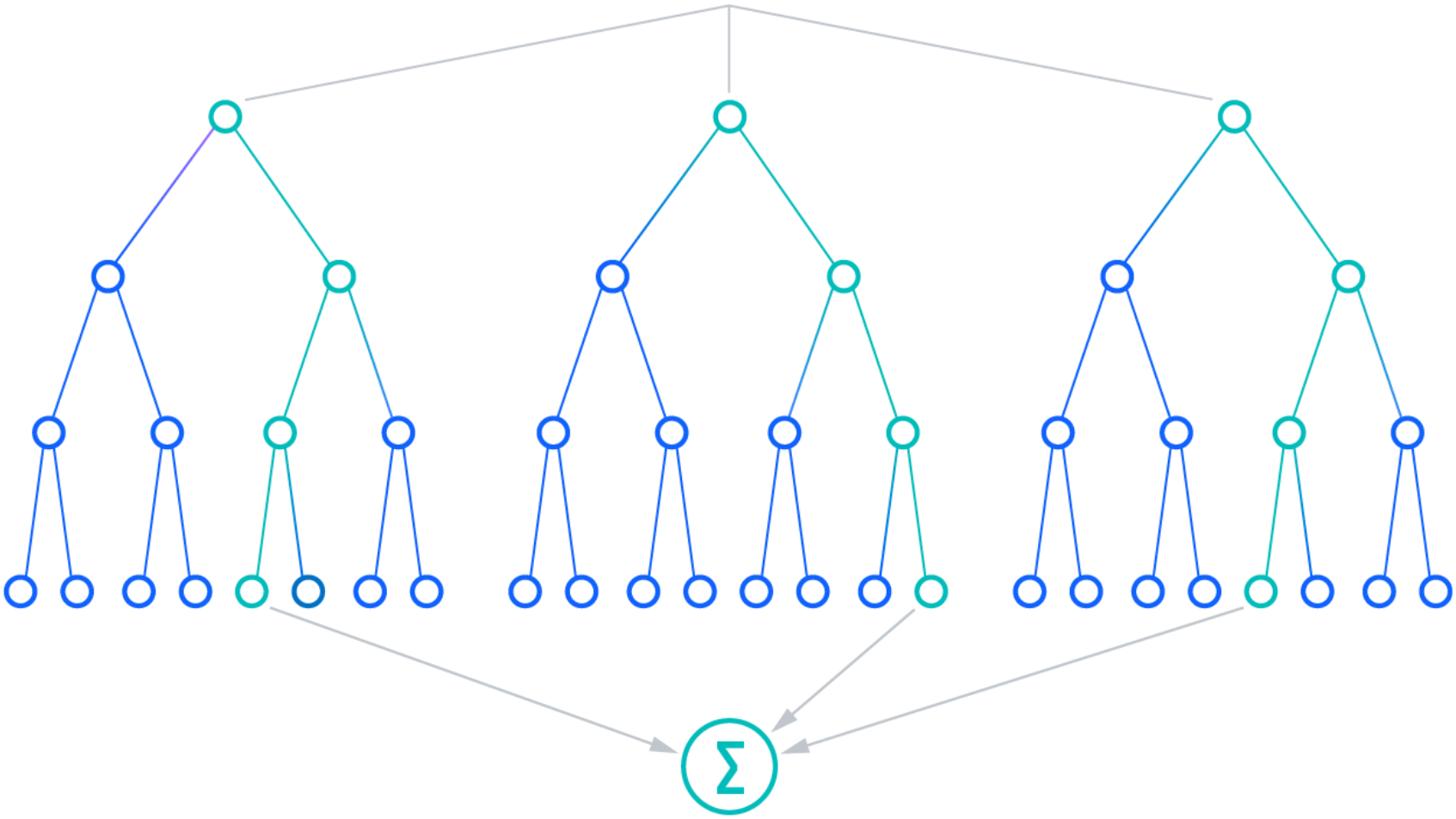


Figure from IBM

Pro and Cons of Random Forests

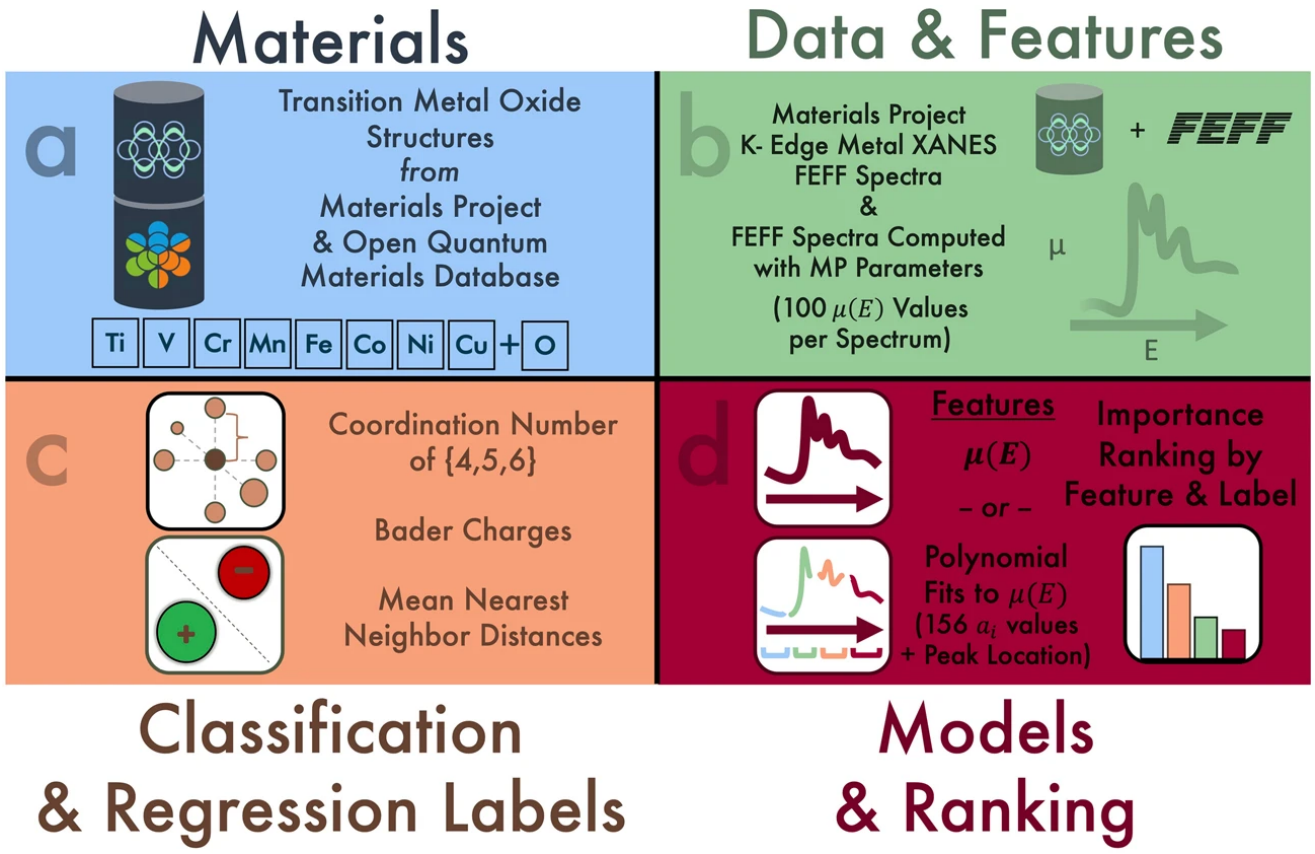
+ Reduced risk of overfitting by averaging.
+ Flexibility and possibility to handle missing data with bagging.
+ Metrics to determine feature importance.
- Time and resource consuming process.
- Complexity in understanding the predictions.

NB: RF can be considered a bootstrap aggregating technique, that aims to reduce variance and to avoid over-fitting.

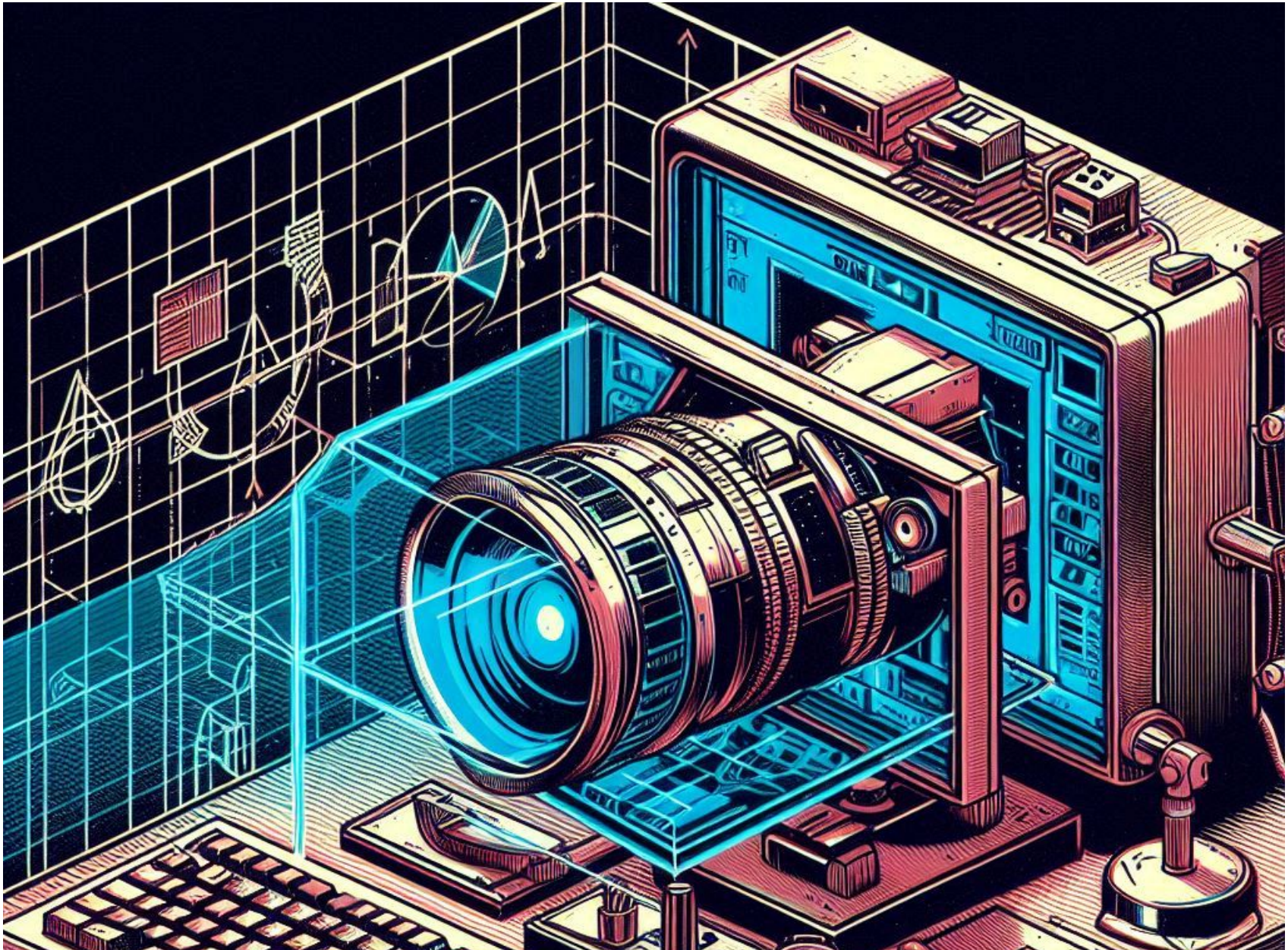
Example of Random Forest application

Materials from Comput Mater 6, 109 (2020).

RF are used in X-ray absorption spectroscopy to automatise the process of spectral interpretation and prediction of the coordinating environment of the absorbing atom.

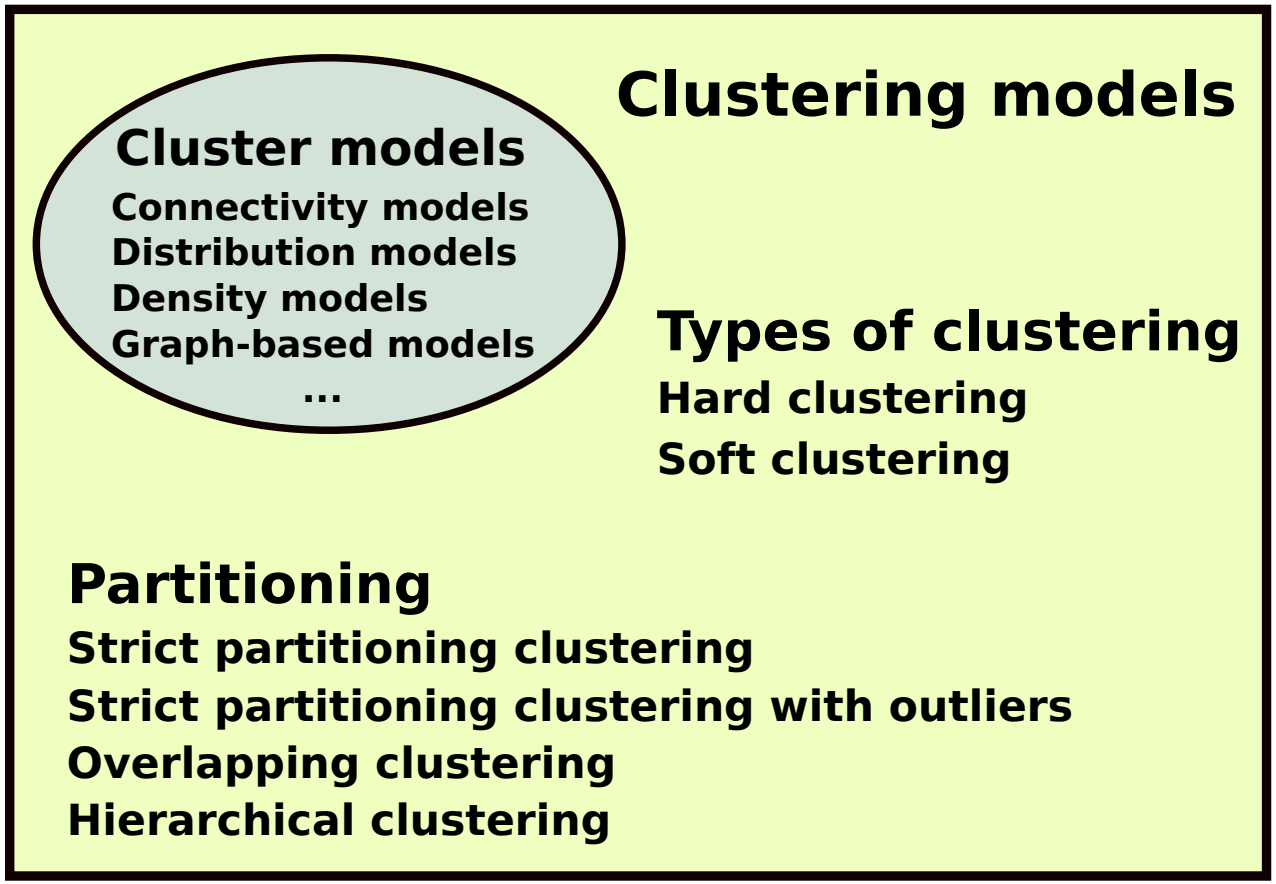


Unsupervised Learning: clustering and dimensionality reduction



Clustering

Clustering: task of grouping a set of objects in such a way that objects in the same group (**cluster**) are more similar to each other than to those in other groups.



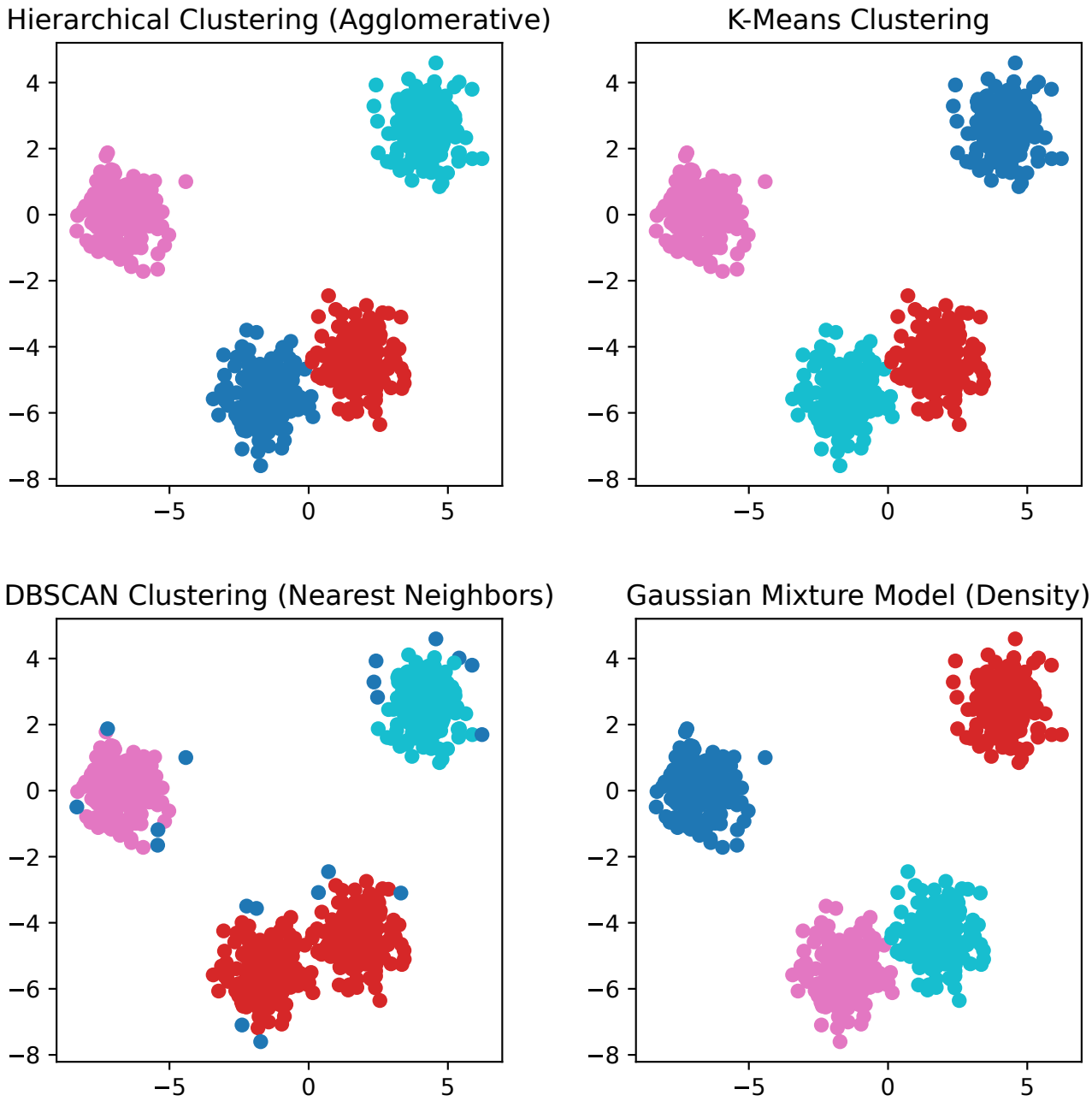
The appropriate algorithm and parameter settings depend on the individual data set and the scope of the analysis.

Unsupervised Learning?

Yes because it involves grouping data points based on their inherent similarities or patterns **without any prior labels or categories**.

1. Clustering algorithms aim to find natural groupings within the data.
2. The goal of clustering is to explore and uncover hidden structures within data.
3. Clustering techniques allow the model to learn relationships within the data.
4. Clustering is often used for exploratory data analysis.

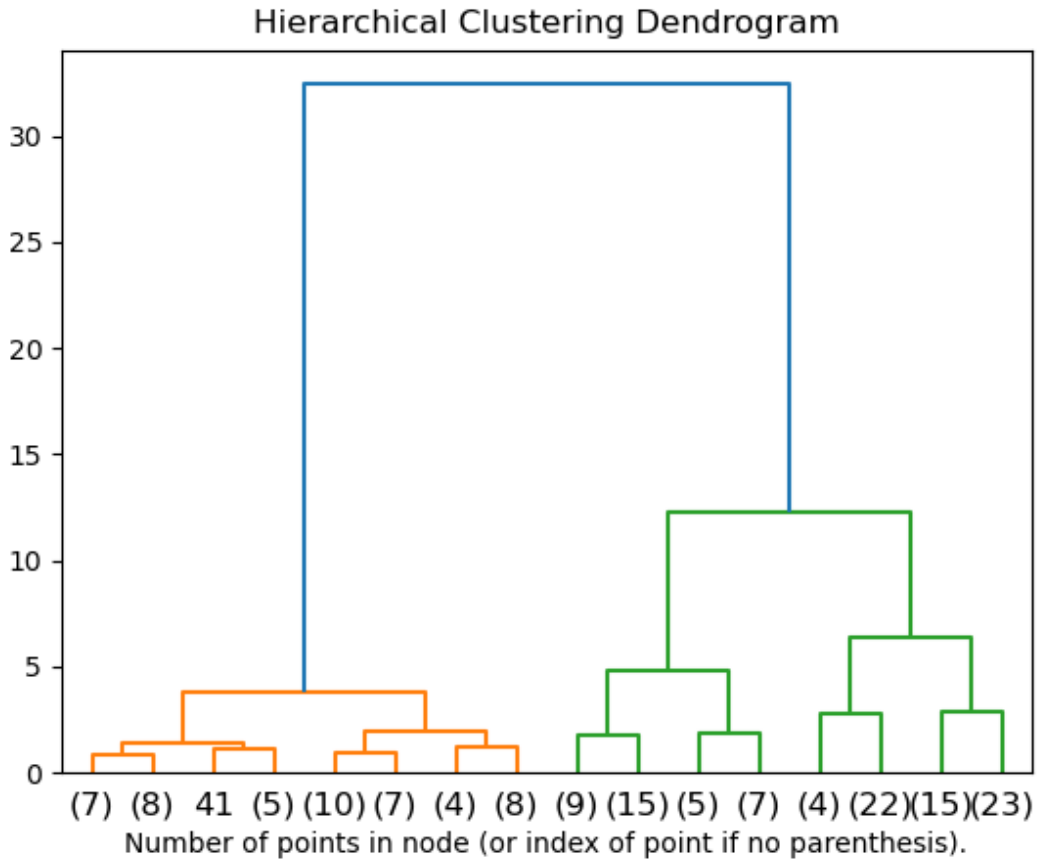
Some clusters



Hierarchical Clustering

Algorithms that build nested clusters by merging or splitting them successively.

The hierarchy is represented as a tree: the root is the unique cluster that gathers all the samples, the leaves are the clusters with only one sample.



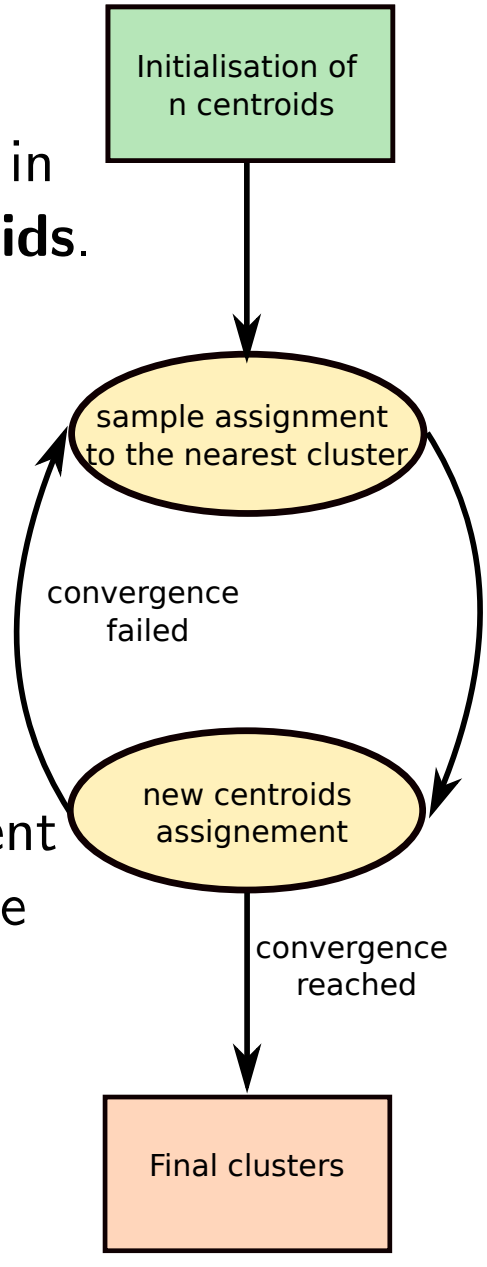
AgglomerativeClustering: bottom up approach, each observation starts in its own cluster, and clusters are successively merged together.

K-Means Clustering

The algorithm divides a set of samples into disjoint clusters, each described by the mean of the samples in the cluster. The means are called the cluster **centroids**. Centroid are chosen to minimise **inertia**.

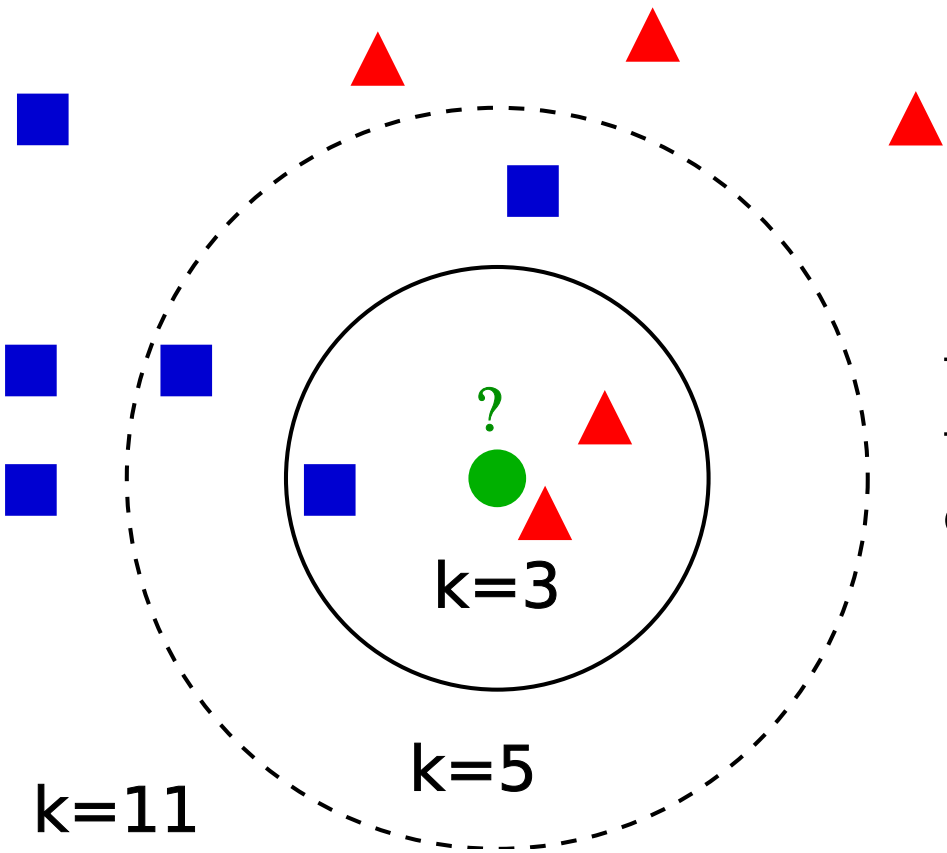
Inertia: measure of how internally coherent clusters are.

Given enough time, K-means will always converge, however this may be to a **local minimum**. The computation is often done several times, with different initializations of the centroids, in order to recover the absolute minima.



Nearest Neighbors Clustering

This is a **supervised** learning method. It is based on finding a predefined number of training samples closest in distance to the new point, and predict the label from these. The distance can be any metric measure: standard Euclidean distance is the most common choice.

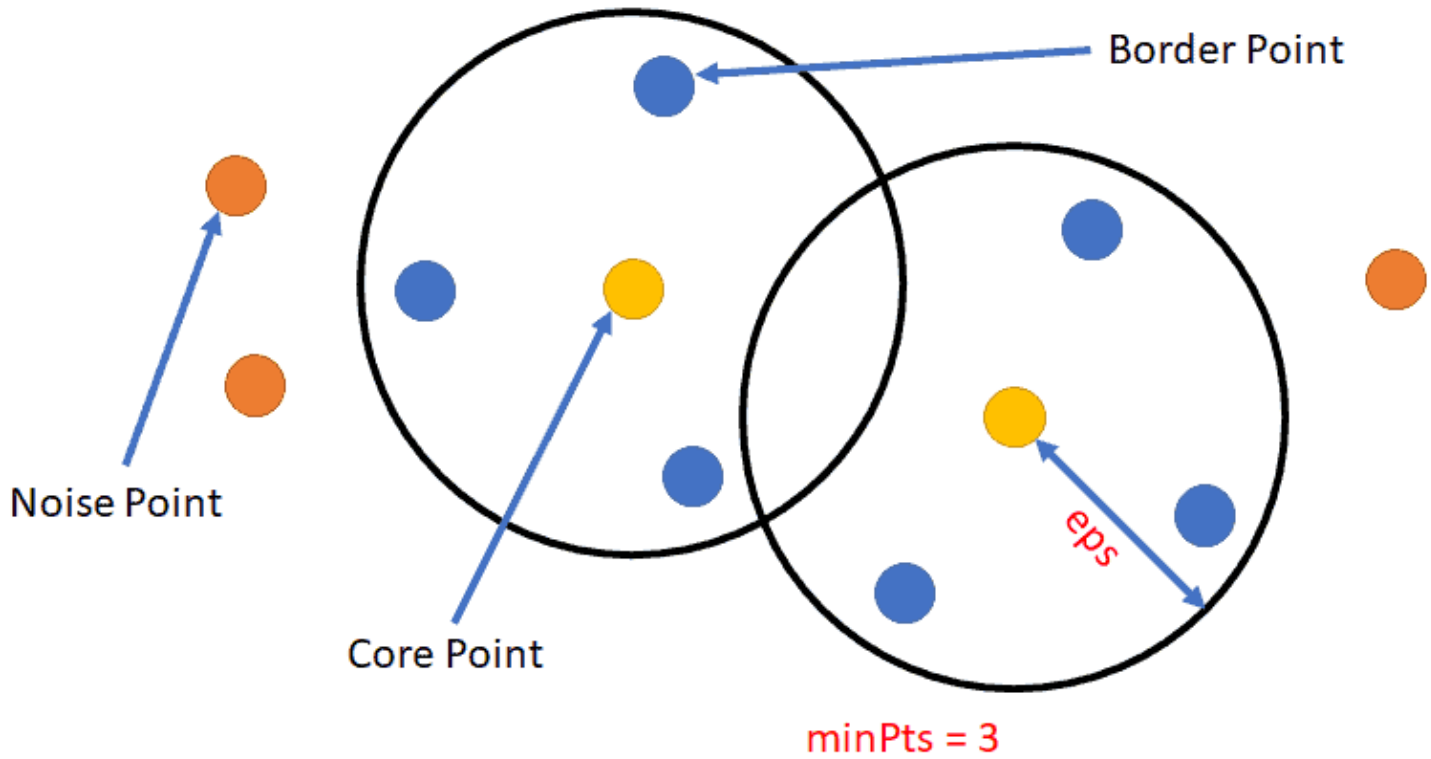


Example of k-NN classification. The test sample should be classified either to blue squares or to red triangles, depending on the value of k.

Density Clustering

Clusters are seen as **areas of high density** separated by areas of low density. The central component is the **core samples**, which are samples that are in areas of high density.

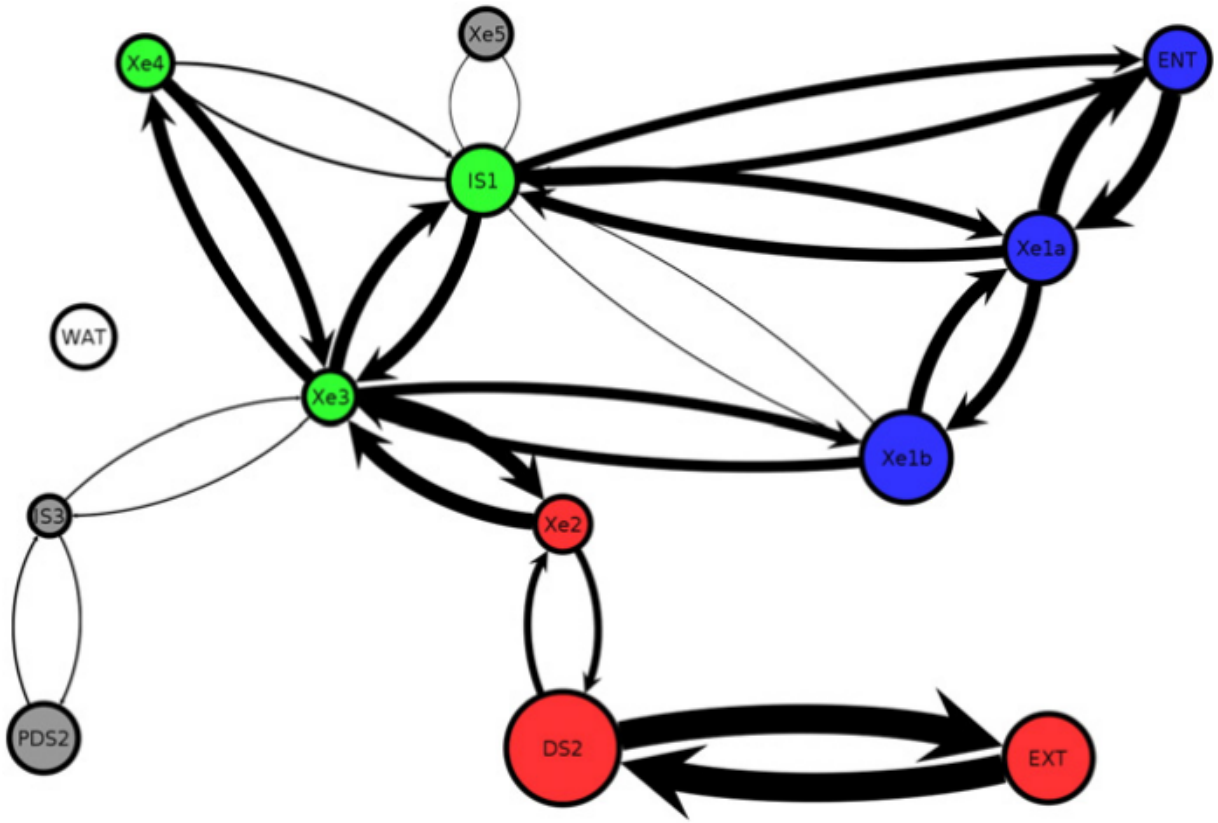
Any sample that is not a core sample and is at least eps in distance from any core sample is considered an outlier by the algorithm.



importance of *k*-means in computational chemistry

Material from Biochimica et Biophysica Acta, 1850,5,996-1005 ,2015.

K-Means have been used to build a transition network analysis from MD of O_2 in truncated Hb.



The size of the pockets and of the arrows is proportional to their weight in the network.

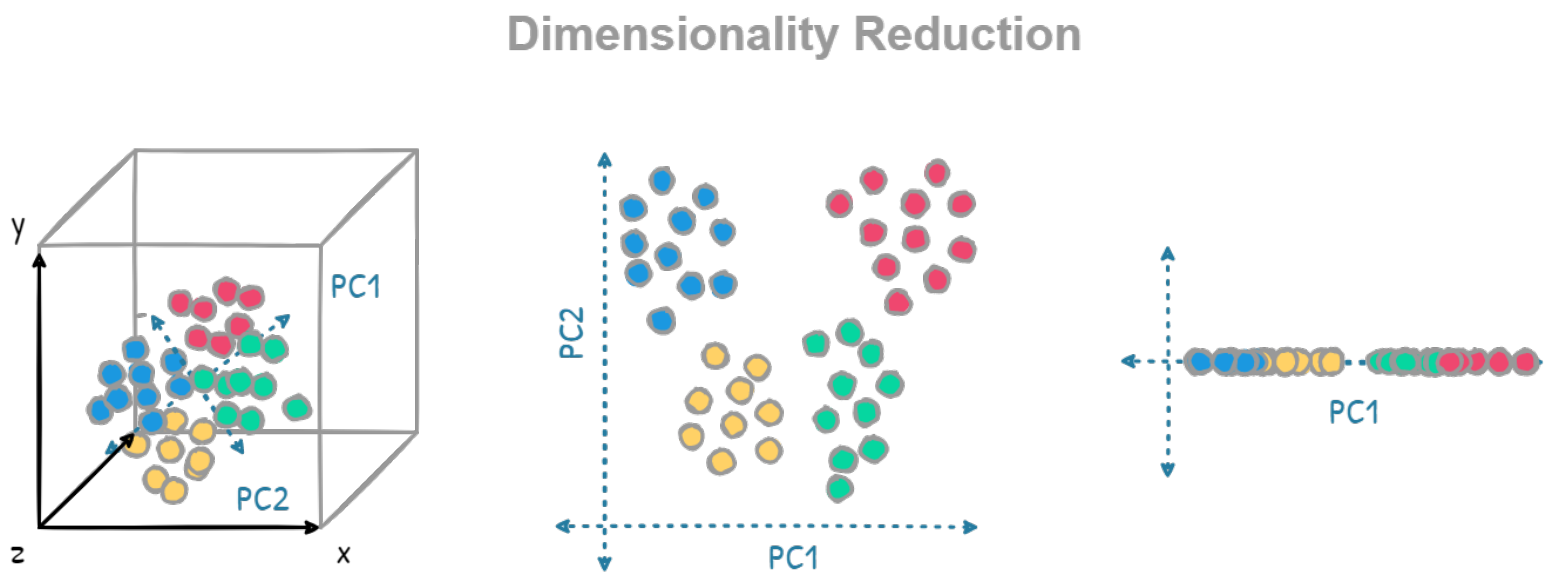
Each colour indicates a channel of the ligand migration.

Dimensionality reduction

The primary goal is to transform high-dimensional data into a lower-dimensional space while minimizing information loss.

Strategies:

- **Feature Selection:** selection of a subset of the most relevant features from the original dataset.
- **Feature Extraction:** creation of new features by combining the original features into a more compact representation.



Principal Component Analysis

The data points are linearly transformed onto a new coordinate system such that the directions capturing the largest variation in the data can be easily identified.

Algorithm:

1. Calculation of the covariance matrix: $\Sigma = \frac{1}{n-1}(X - \bar{X})^T(X - \bar{X})$
2. Eigenvalue decomposition: $\Sigma = W\Lambda W^{-1}$
3. Principal components selection: $P = [w_1, w_2, \ldots, w_k]$
4. Data projection: $Y = XP$

X: original dataset.

n: number of samples in X.

$\bar{X}$: mean of the dataset.

σ: covariance matrix.

W: eigenvectors matrix.

Λ: diagonal eigenvalue matrix.

P: projection matrix

$w_1, ..., w_k$: k most important eigenvectors.

Y: lower dimension dataset.

PCA: visual example

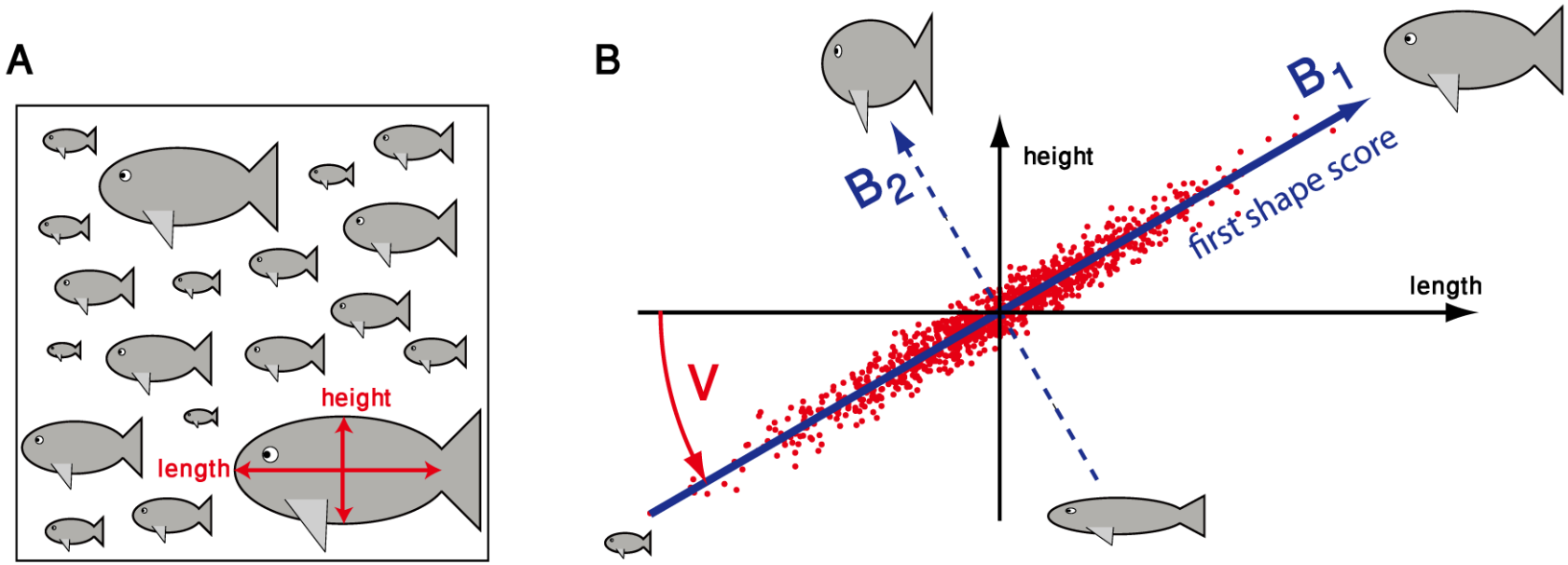
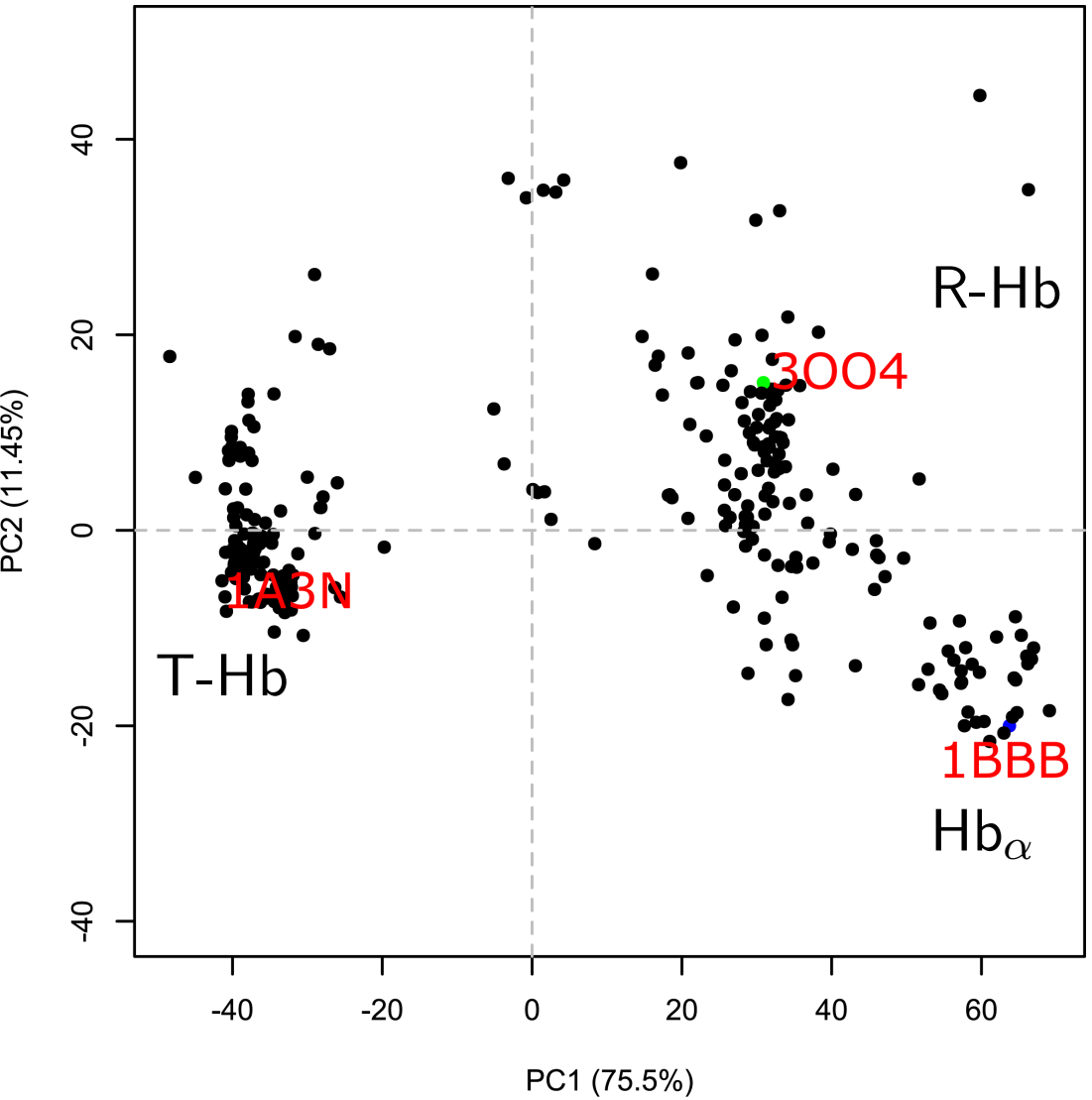


Figure from PLoS ONE 9(11): e113083, 2014.

Importance of PCA in computational chemistry

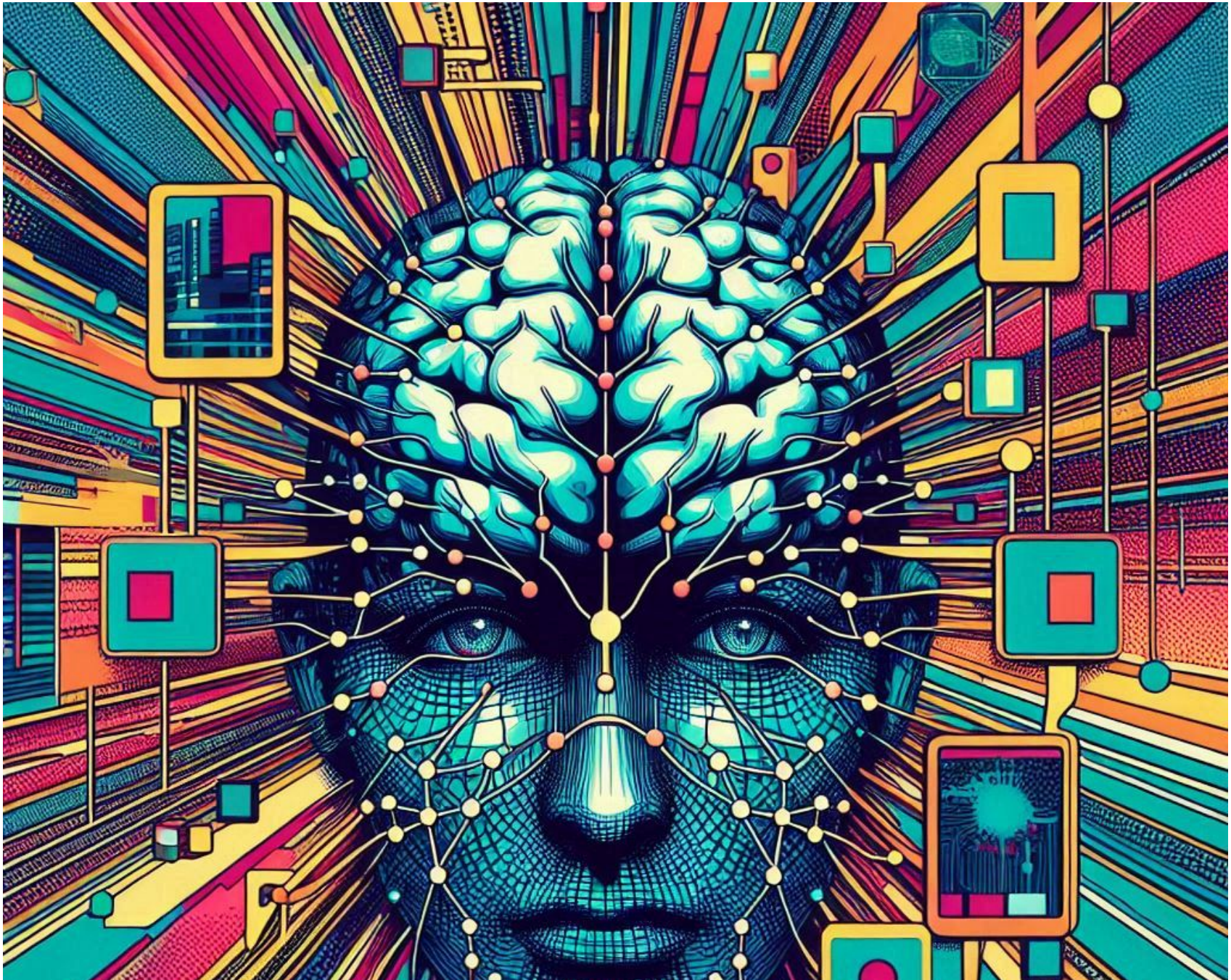
Material from PLoS ONE 13(12): e0208465, 2018.



The author used an advanced PCA model to compare globin proteins with multimeric states raging from monomers to 24mers.

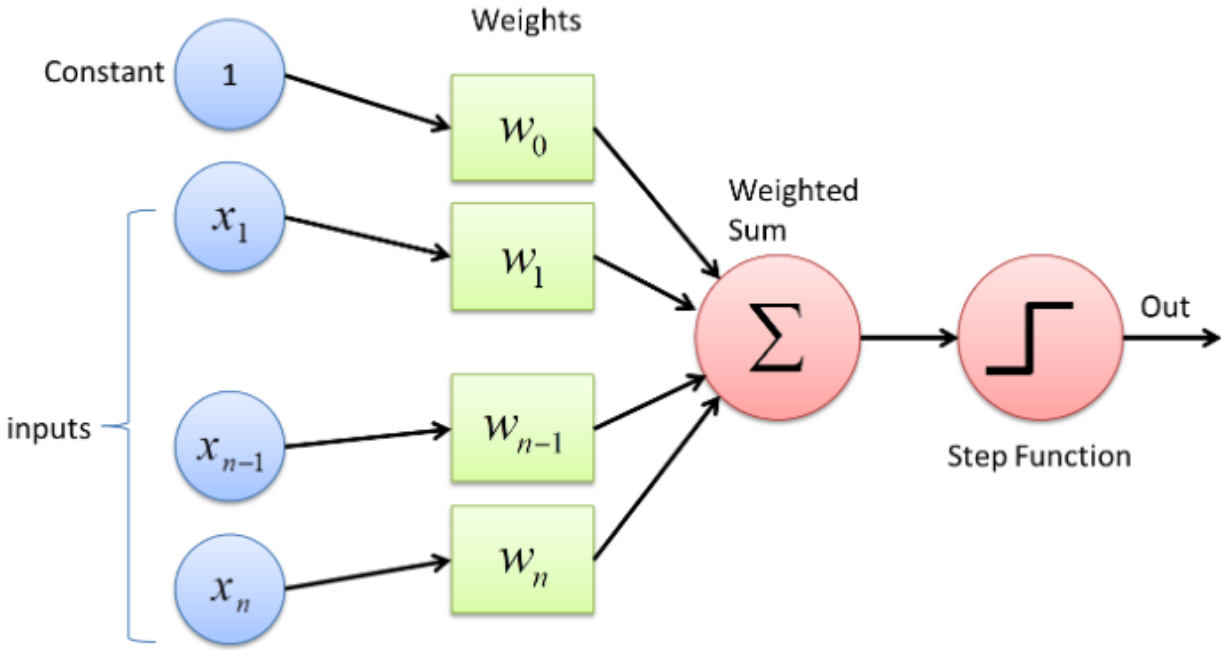
The clustering results of the globin chains reflect the quaternary structure of the chains.

Neural Networks



Multilayer Perceptron

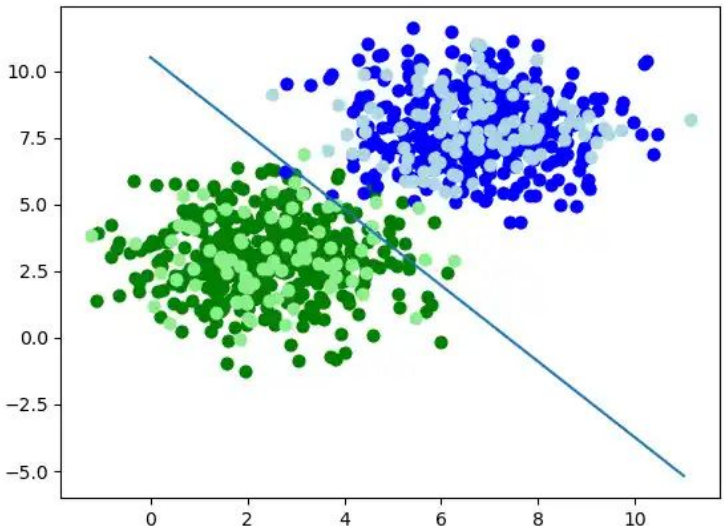
MLP is a **supervised learning** algorithm that learns a function $f : R^{input} \rightarrow R^{output}$ by training on a dataset. It can be used for learning non-linear function approximator for either classification or regression.



Minimsation of the weigths is done using Stochastic Gradient Descent or the Adam optimiser, using the loss function as target.

$$w \leftarrow w - \eta(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w})$$

Simple example of MLP




```
from sklearn.datasets import make_blobs

n_samples = 1000
samples, labels = make_blobs(n_samples=n_samples,
                             centers=([2.5, 3], [6.7, 7.9]),
                             cluster_std=1.4,
                             random state=0)
```

```
from sklearn.model_selection import train_test_split
res = train_test_split(samples, labels,
                       train_size=0.8,
                       test_size=0.2,
                       random_state=1)

train_data, test_data, train_labels, test_labels = res
```

```
from perceptrons import Perceptron

p = Perceptron(weights=[0.3, 0.3, 0.3],
               learning_rate=0.8)

for sample, label in zip(train_data, train_labels):
    p.adjust(label,
             sample)

evaluation = p.evaluate(train_data, train_labels)
print(evaluation)
```

Can we build all the models with scikit-learn?

Nope! there are limitations!

- Lack of Deep Learning Support, as it supports only simple models and does not have access to GPU acceleration.
- There is no possiblity of building custom NN models, and consequent difficulty in optimising hyperparameters.
- Limited capabilities in training, as scikit learn can deal only with small datasets.

And so?

We can still use scikit-learn as support for more complex libraries (as tensorflow and pytorch).

And so?

Initialisation

1. Load the Data set.
2. Separate Inputs and Target Variables.
3. Splitting the Data into Training (70/80 %), Testing (20/10 %) and Validation Sets (10 %).
4. Normalise Input Data.

Marco's Mystery Magical Machine

Data analysis and vasualisation

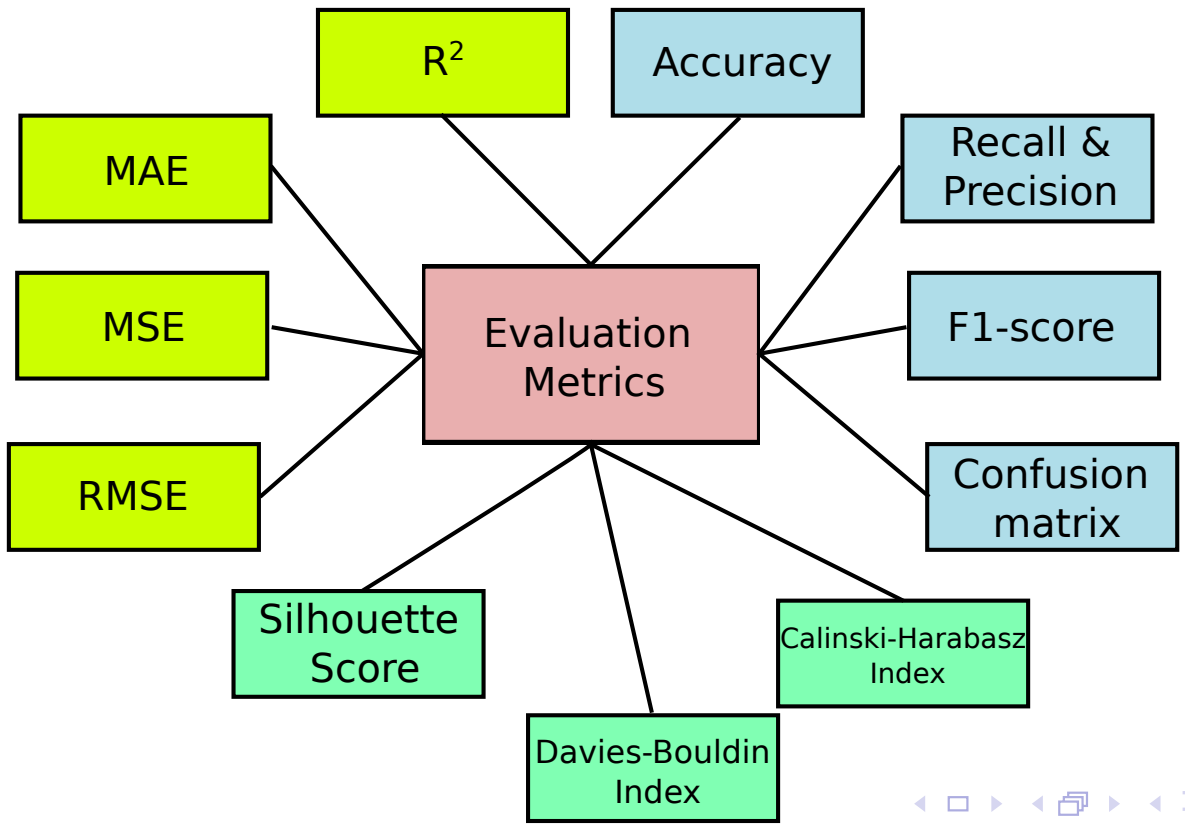
You can use scikit-learn for initialising your dataset, let your other library to perform the data training and evaluation, and use metrics for the final analysis.

Evaluation Metrics

Evaluation Metrics

It is a quantifiable measure to evaluate the performance of a machine learning model. It is impossible to get a 100% accurate model, so it is important to have a measure of the error.

Different metrics provide different insights into how well the model is performing and help in assessing its effectiveness for a specific task.



R squared (R^2)

It is a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- n: number of data in our model.
- y_i: observed value for the ith variable.
- $\hat{y}_i$: predicted value for the ith variable.
- $\bar{y}$: mean value of y in the model.

It varies between 0 (**no agreement**) and 1 (**perfect agreement**).

+ It measures how the model fits the observations and it allows comparison of different models.

! it is not suitable for comparisons of models with different sample sizes.

- It is sensitive to outliers and does not capture model complexity.

Adjusted R squared:

$$R^2 = 1 - \frac{\left(1 - R^2\right)(n-1)}{n-k-1}$$

k: number of independent variables.

Mean Absolute Error (MAE)

It measures the magnitude of the errors in a set of predictions as average of the absolute difference.

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

n: number of data in our model.

y_i: observed value for the ith variable.

$\hat{y}_i$: predicted value for the ith variable.

+ Easy to understand and not size dependent.
+ Same units as the measurement
- Less sensitive to outliers.
- Not differentiable.

Mean Square Error (MSE)

it measures the average of the squares of the errors.

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

n: number of data in our model.

y_i: observed value for the ith variable.

$\hat{y}_i$: predicted value for the ith variable.

+ Penalise larger errors more than MAE.
+ Differentiable
- Disproportionately influenced by outliers.
+ Not the same units as the measurements.

Root Mean Square Error (RMSE)

it is the squre root of the MSE.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$$

n: number of data in our model.

y_i: observed value for the ith variable.

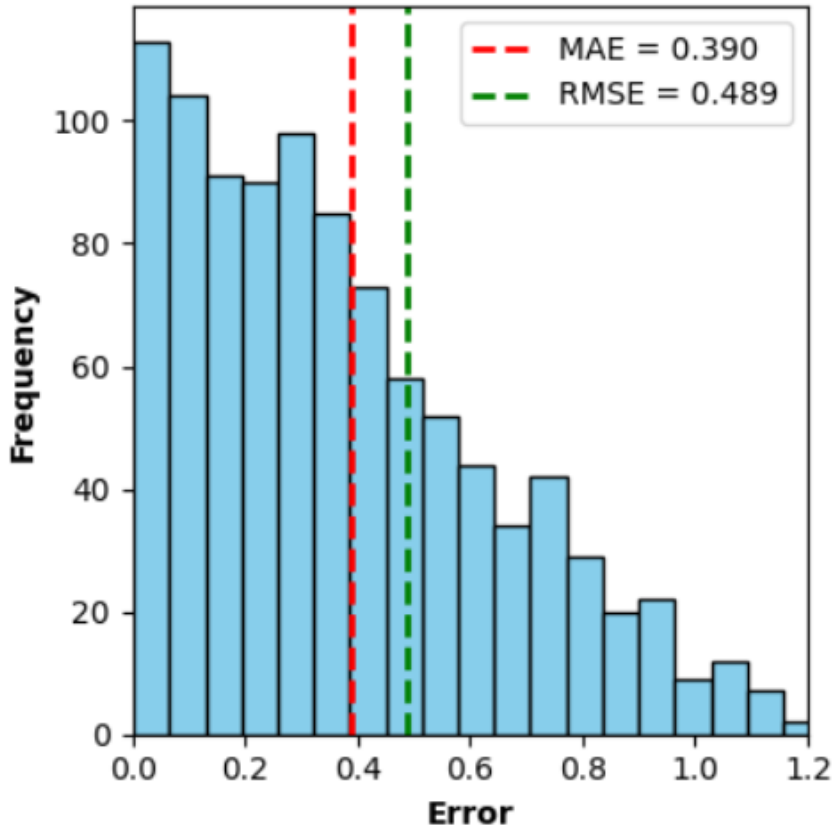
$\hat{y}_i$: predicted value for the ith variable.

\+ It can be used in combination with MSE

\- highly influenced by outliers.

! Same units as the measurements, but more difficult to understand than MAE.

MAE *vs* RMSE



As you can see, the values of the MAE and RMSE differ by a 0.1 units within the dataset.

Classification Metrics

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1-score = 2 * \frac{Recall \times Precision}{Recall + Precision}$$

TP: True Positive.

TN: True Negative.

FP: False Positive.

FN: False Negative.

Precision and Recall are important for unbalanced classes, Accuracy and F1-score are helpful for balanced ones.

Confusion Matrix

It is a specific table layout that allows visualization of the performance of an a classification algorithm. It is known as **matching matrix** in unsupervisioned learning.

		Predicted Condition	
Total Population		**P**	**N**
Actual Condition	**P**	**TP**	**FN**
	N	**FP**	**TN**

More complex models can be built, both increasing the prediction or including more metrics.

Confusion Matrix

It is a specific table layout that allows visualization of the performance of an a classification algorithm. It is known as **matching matrix** in unsupervisioned learning.

		Predicted Condition	
Total Population		**P**	**N**
Actual Condition	**P**	**TP**	**FN**
	N	**FP**	**TN**

More complex models can be built, both increasing the prediction or including more metrics.

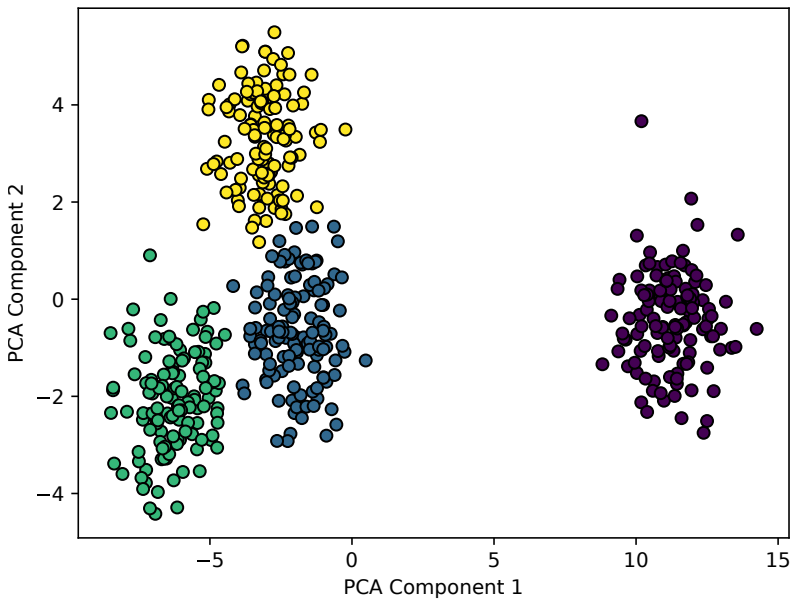
		Predicted Condition		
Total Population		**PP**	**PN**	**PU**
Actual Condition	**P**	**TP**	**FN**	**FU**
	N	**FP**	**TN**	**FU**
	U	**FPU**	**FNU**	**TU**

Clustering metrics

Silhouette Score:
It measures how similar a data point is to its own cluster compared to other clusters. It ranges from -1 (bad) to 1 (good).

Davies-Bouldin Index:
Measures the average similarity between clusters. Lower values indicate better separation between clusters.

Calinski-Harabasz Index: Evaluates the ratio of between-cluster dispersion to within-cluster dispersion. A higher score usually indicates that the clusters are dense and well separated.



Silhouette Score: 0.66

Davies-Bouldin Index: 0.47

Calinski-Harabasz Index: 4289.21

Distribution measurements

Skewness

- **Positive Skew:** long tail on the right side.
- **Negative Skew:** long tail on the left side.
- **Zero Skew:** normal distributions have a skewness of approximately zero.

Kurtosis

- **High Kurtosis(> 3):** heavier tails and a sharper peak than a normal distribution; higher likelihood of outliers.
- **Low Kurtosis (< 3):** lighter tails and a flatter peak than a normal distribution; fewer extreme outliers.
- **Normal Kurtosis ($= 3$):** similar to a normal distribution.

So what?

The choice of evaluation metric depends on the requirements of your analysis and the nature of the data. You need to consider multiple metrics, as a single one is not enough for evaluation.

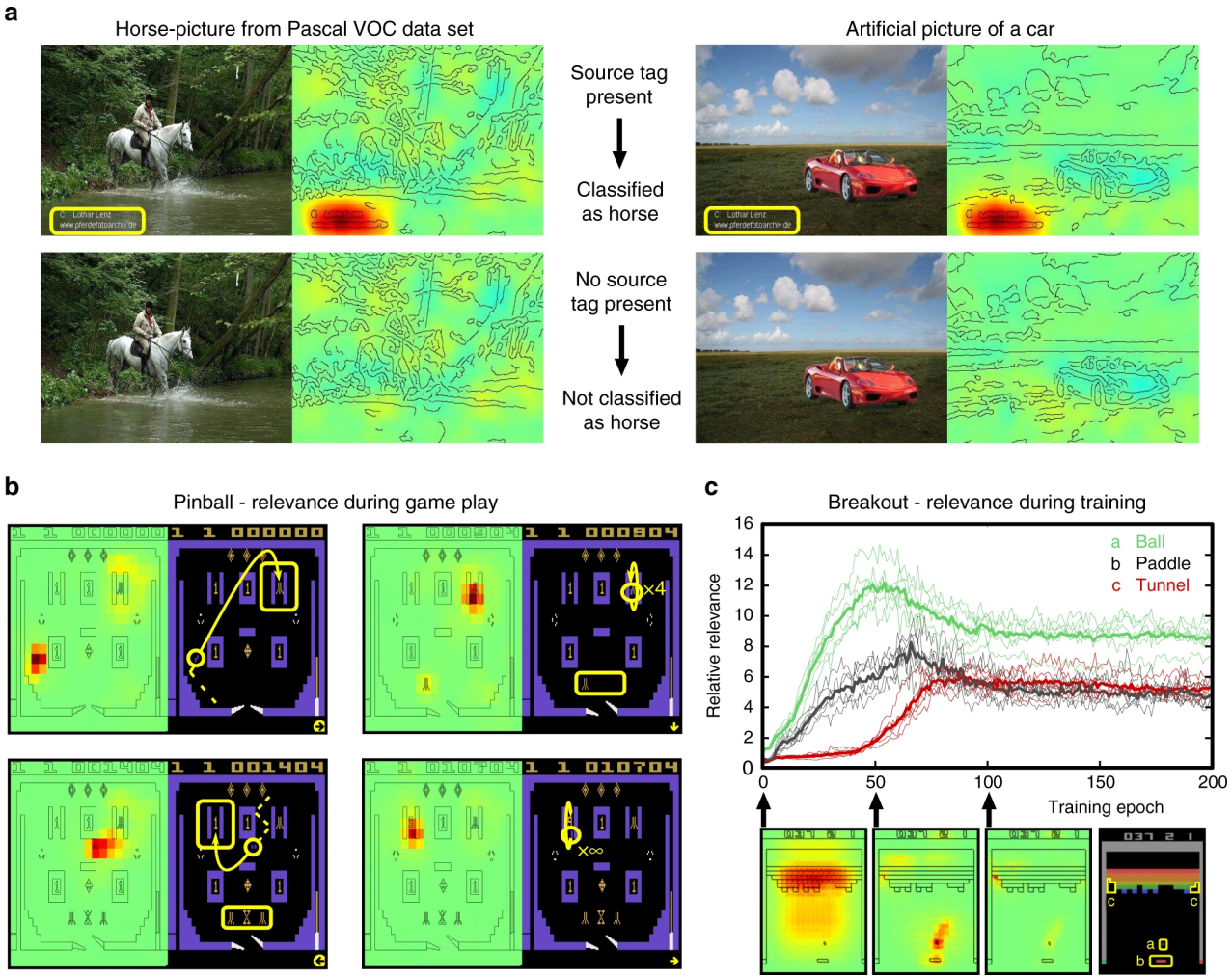
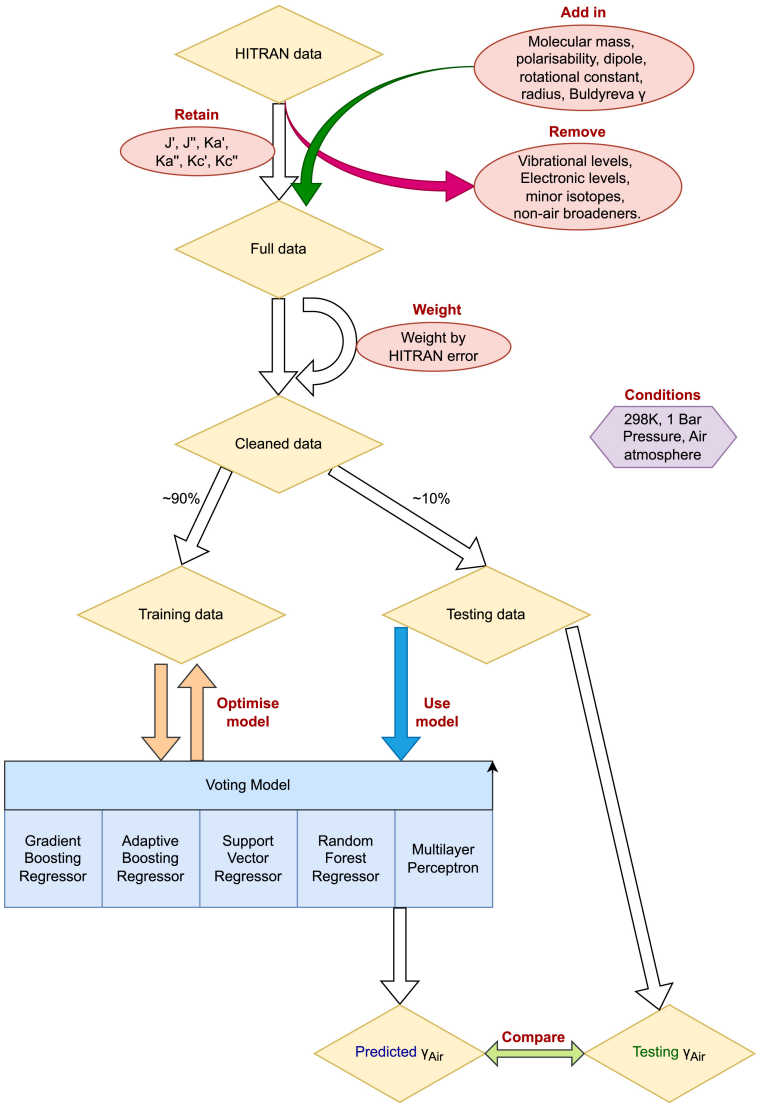


Figure from Nature Communications 10, 1096 (2019)

Few final words

Predicting the rotational dependence of line broadening using machine learning

Material from Journal of Molecular Spectroscopy 401 (2024) 111901.

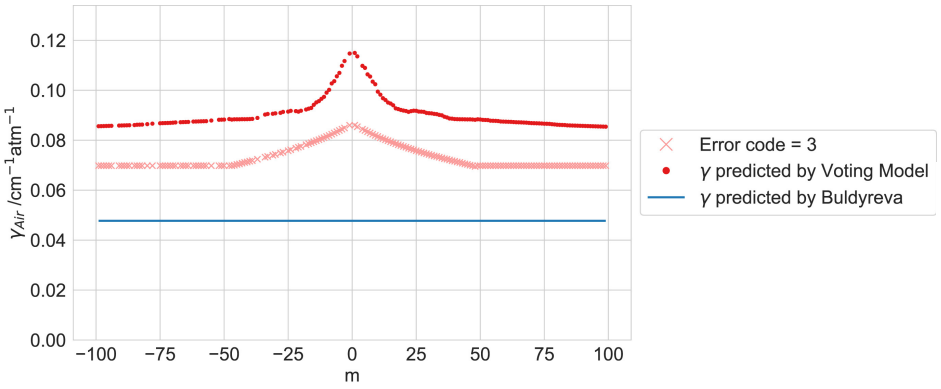


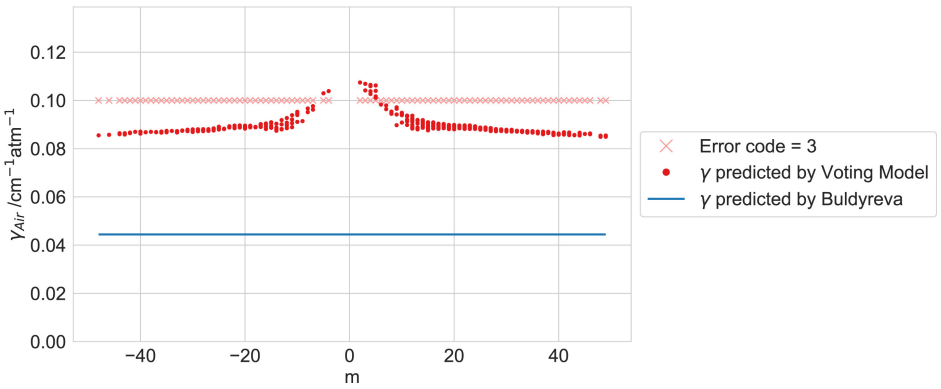
The scope is to use ML methods to mass produce **pressure broadening** parameters for a large number of molecules.

Training on 43 molecules to make predictions for the other 5. The final operational model is trained on all 48 molecules which can be used to make predictions for molecules not in the dataset.

Predicting the rotational dependence of line broadening using machine learning

The model successfully predicts about 69% of the data provided by HITRAN within uncertainties. For a significant number of molecules the data is actually highly uncertain: being unavailable, fixed as a constant or estimated.

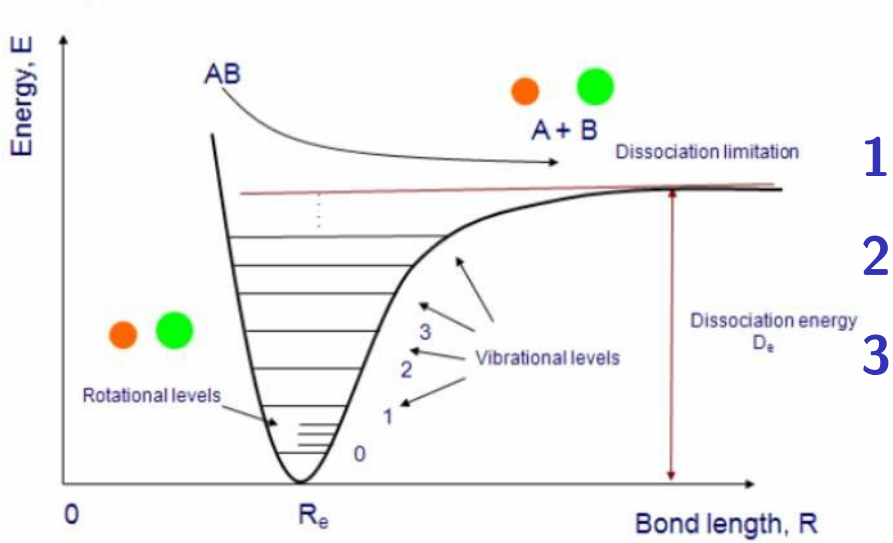




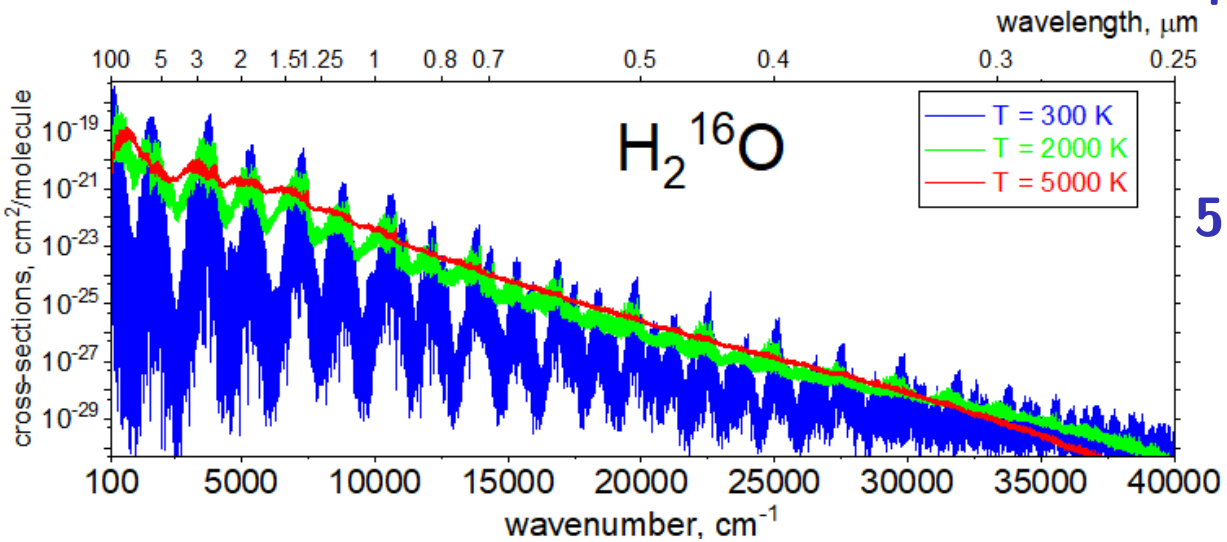
Model	RMSE/%
Decision Tree	35.9
Random Forest	32.3
Gradient Boosting	38.0
Adaptive Boosting	32.9
SVR	40.5
SGD	36.2
Dummy Regressor	42.4
MLP	27.9
Voting Regressor	29.3
Buldyreva Model	49.0

Hands-on

$$E_{\text{rovib, anharmonic}}(v, J) = BJ(J+1) + \left(v + \frac{1}{2}\right) h\nu - \left(v + \frac{1}{2}\right)^2 h\nu x_e$$



1. Get diatomic molecular constants from NIST.
2. Get diatomic energy levels from ExoMol.
3. Predict the molecular constants.



4. Get Energy levels from the MARVEL website for polyatomics.
5. Create models to predict the molecular energies below chemical accuracy.

Online resources

- Scikit-learn user guide.
- GeeksforGeeks.
- Python course.
- Medium.

Acknowledgments

To the survivors: thank you for your attention and your participation!

Thanks to Profs **Faginas-Lago** and **Fanó** for letting me participate to this amazing occasion!

Thanks to Luis and Nicolò for the proofreading and the proofcoding!

Thanks to Jonathan for the suggestions for the tutorial!


That's all Folks!