

# AI and ML for Chemists

## Topic 3.2: Simple ML Architectures

### Pre-lecture Explore

Dr. Ganna (Anya) Gryn'ova

# Once you have the data: Training the model

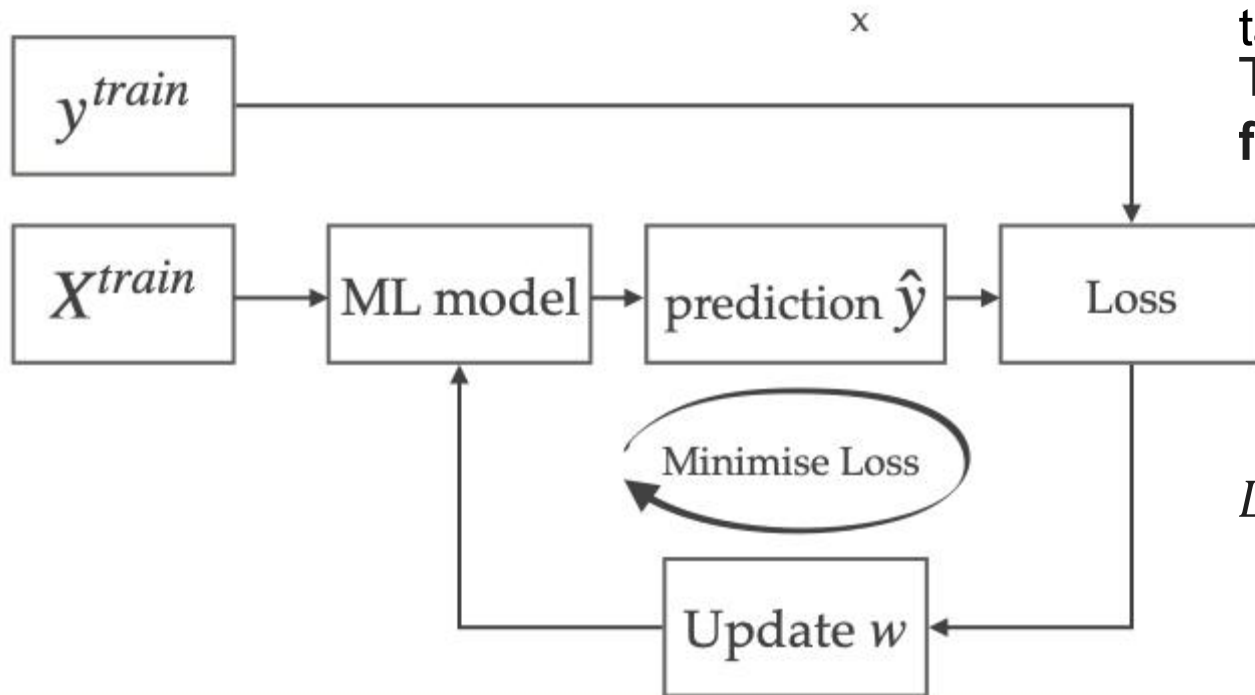
$$y = f(x)$$

output or target (label)      prediction function      input (feature)

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 \dots + error$$

The **weights** are the parameters of a model that determine the strength and direction of the relationship between the features and the target.

The goal of training is to minimise a **loss function** by updating the weights.



$$L1 = \sum_{i=1}^n y_{true} - y_{predicted}$$

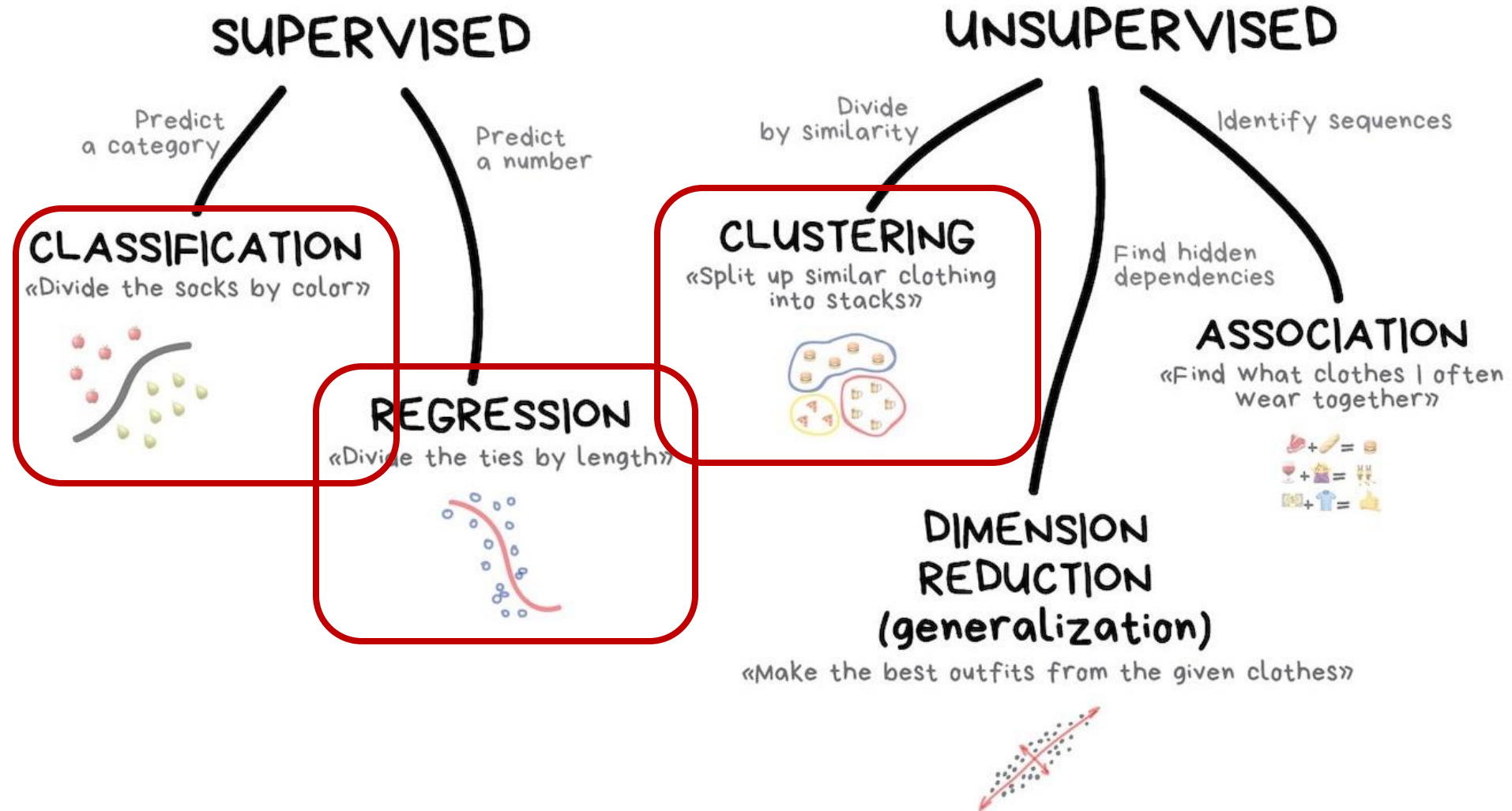
$$L2 = \sum_{i=1}^n (y_{true} - y_{predicted})^2$$

$$MSE = \frac{1}{n} L2$$

Mean Square Error

# Types of ML

Remember the types of machine learning (Topic 1):



# Linear regression

In algebra

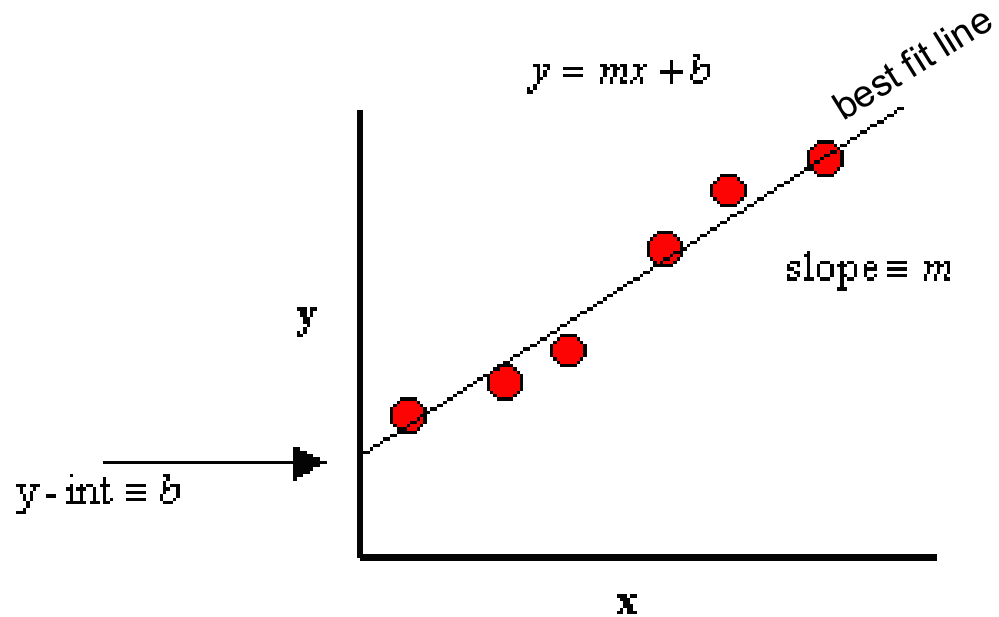
$$y = b + mx$$

$y$  – dependent variable

$x$  – independent variable

$m$  – slope

$b$  – intercept



In ML

$$y = w_0 + w_1x_1 + e$$

$y$  – output or target or label

$x_i$  – input or feature

$w_0$  – bias

$w_i$  – weight

$e$  – error

For more complex input with multiple features

$$y = w_0 + w_1x_1 \dots + w_ix_i + e$$

# Non-linear regression

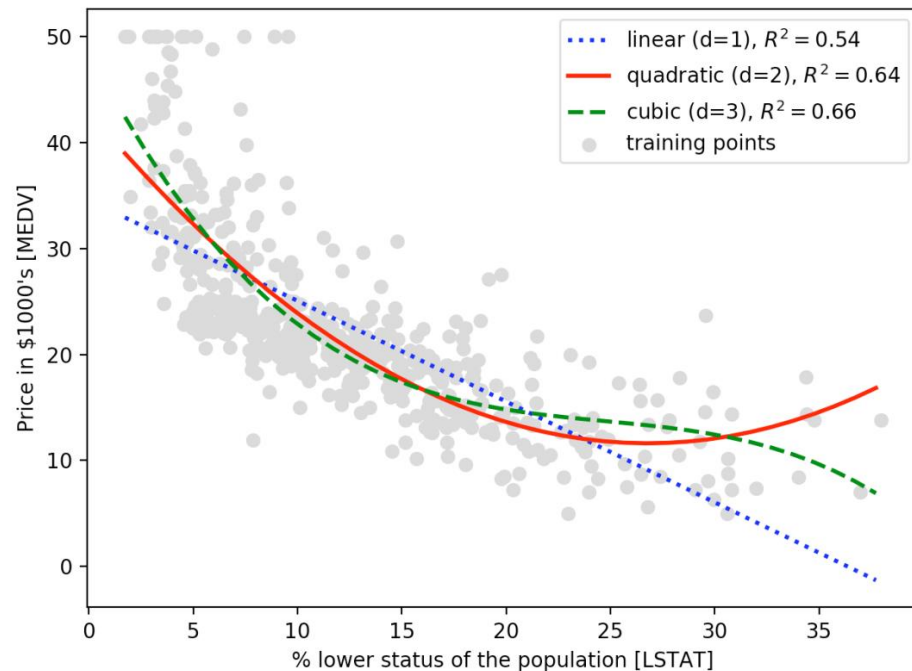
In algebra

$$y = b + m_1x + m_2x^2 + m_3x^3$$

$$y = b + m^2$$

$$y = \log(b + mx)$$

...



In ML

$$y = w_0 + w_1x_1 \dots + w_ix_i + e$$

$$x_1 = x$$

$$x_2 = x^2$$

$$x_3 = x^3$$

...

## Coefficient of determination

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

sum of squares of residuals  
total sum of squares  
(~ variance of the data)

$y_i$  – true output

$\bar{y}$  – mean of all  $y_i$

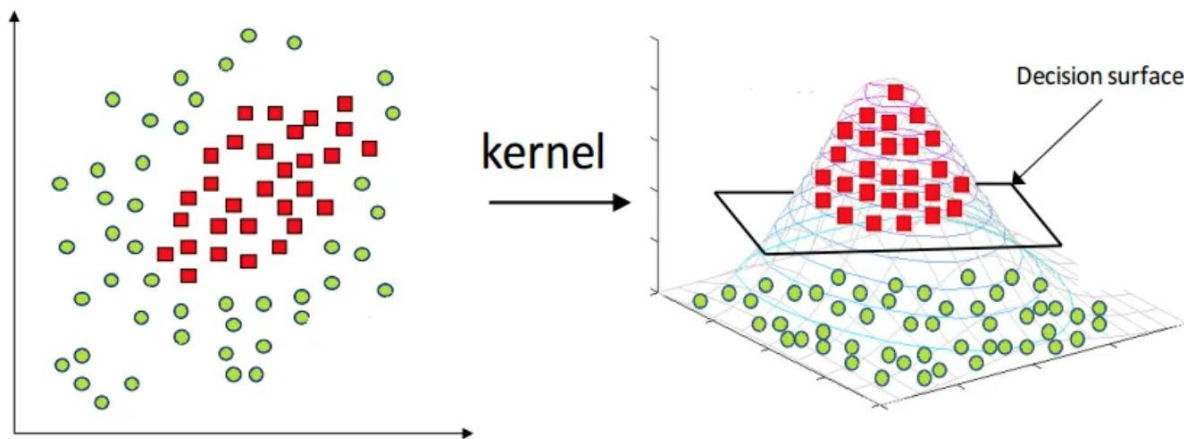
$\hat{y}_i$  – predicted output

# Kernel ridge regression

**Ridge regression** is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables (outputs) are *highly correlated*. Ridge regression allows avoiding *overfitting* through *regularisation*:

$$L2 = \underbrace{\sum_{i=1}^n (y_{true} - y_{predicted})^2}_{\text{loss}} + \lambda \underbrace{\sum_{i=1}^n w_i^2}_{\text{penalty}}$$

However, not all data can be separated linearly. In the real world, data can be randomly distributed, making it difficult to classify the data with linear regression.



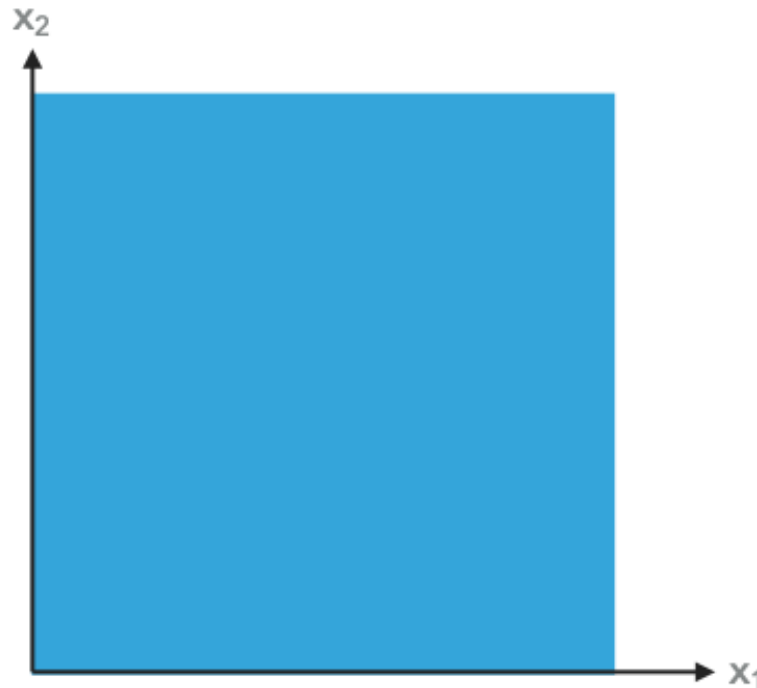
The **kernel trick** allows forming a more complex model in the original feature space without incurring huge computing costs to transform the data into a higher dimensional space. It is more efficient and less expensive to use the kernel function than to do more complicated computations in multidimensional space.

# Classification: decision trees

Decision trees divide the data feature space into a set of hypercubes that are classified as *signal* (+1) or *background* (-1).

- Each region can be fitted with a constant to represent the data in that region.
- We can recursively continue to sub-divide the data until some minimum number of examples are left in each sub-division.

Can describe the data  
as the root node.

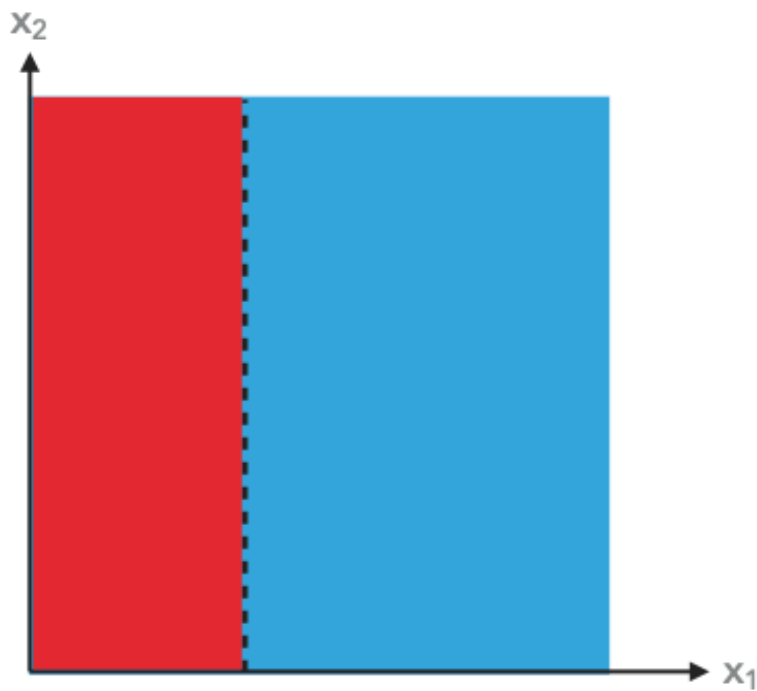
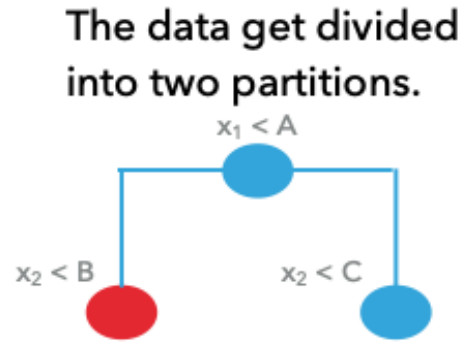


Example feature space  
described by  $X=\{x_1, x_2\}$

# Classification: decision trees

Decision trees divide the data feature space into a set of hypercubes that are classified as *signal* (+1) or *background* (-1).

- Each region can be fitted with a constant to represent the data in that region.
- We can recursively continue to sub-divide the data until some minimum number of examples are left in each sub-division.

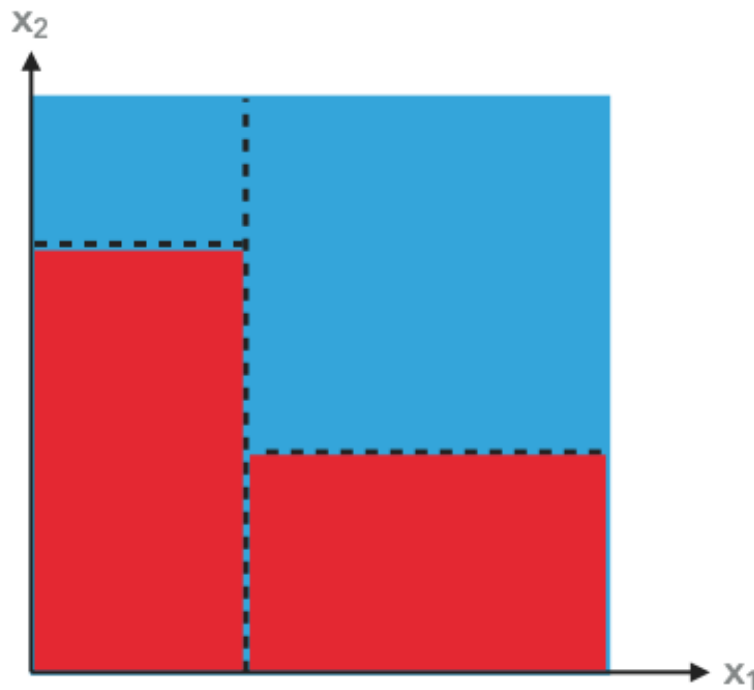
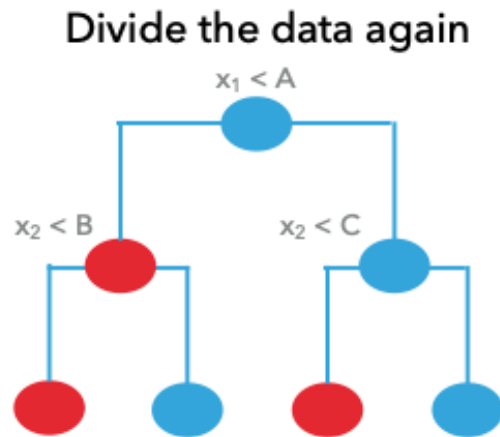


Cut on the feature space to separate the data into two different regions.

# Classification: decision trees

Decision trees divide the data feature space into a set of hypercubes that are classified as *signal* (+1) or *background* (-1).

- Each region can be fitted with a constant to represent the data in that region.
- We can recursively continue to sub-divide the data until some minimum number of examples are left in each sub-division.



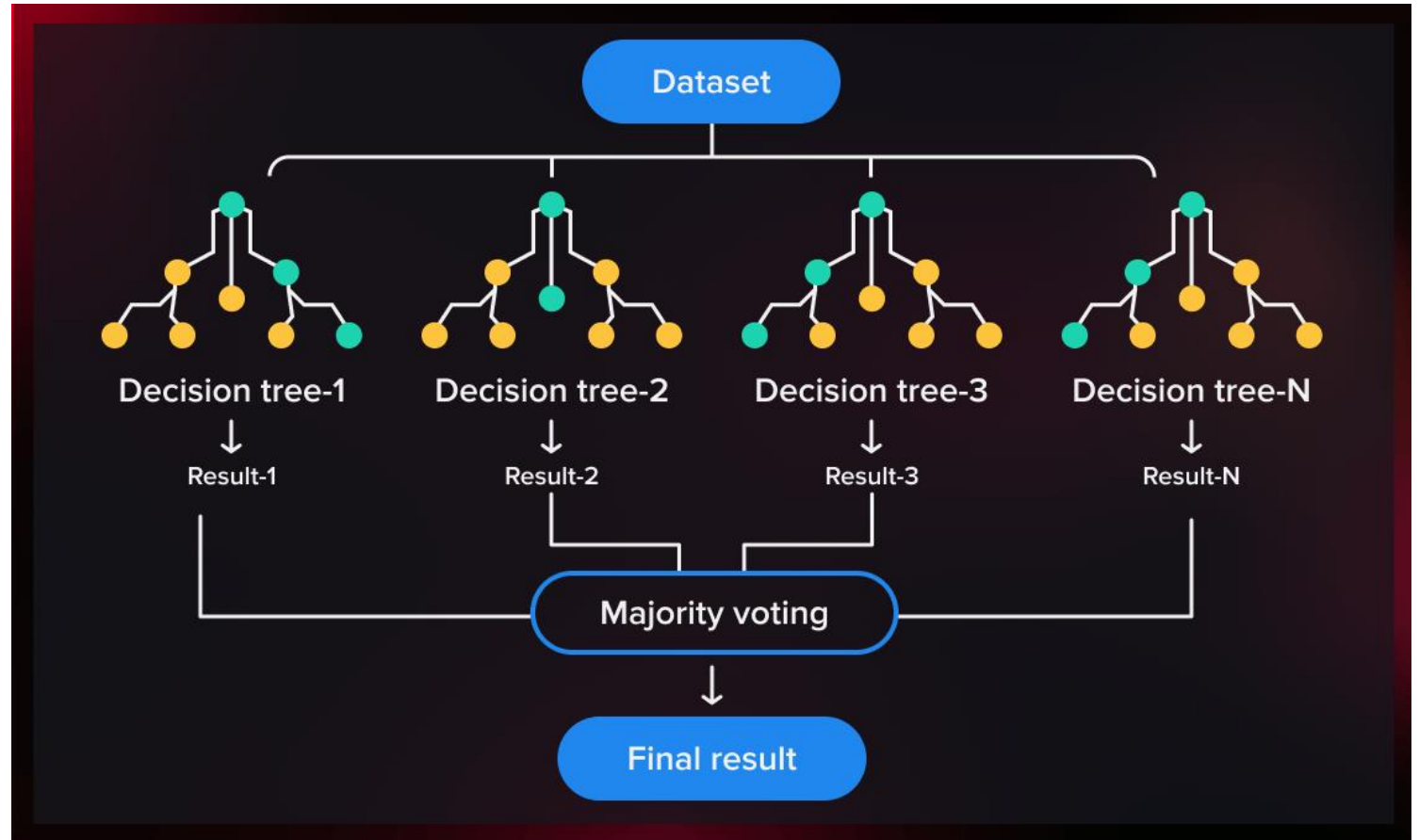


# Random forests

- Decision trees are “*weak learners*”, as they can take input features that only weakly separate types of example and combine those features to increase the separation.
- A single tree is susceptible to overtraining.

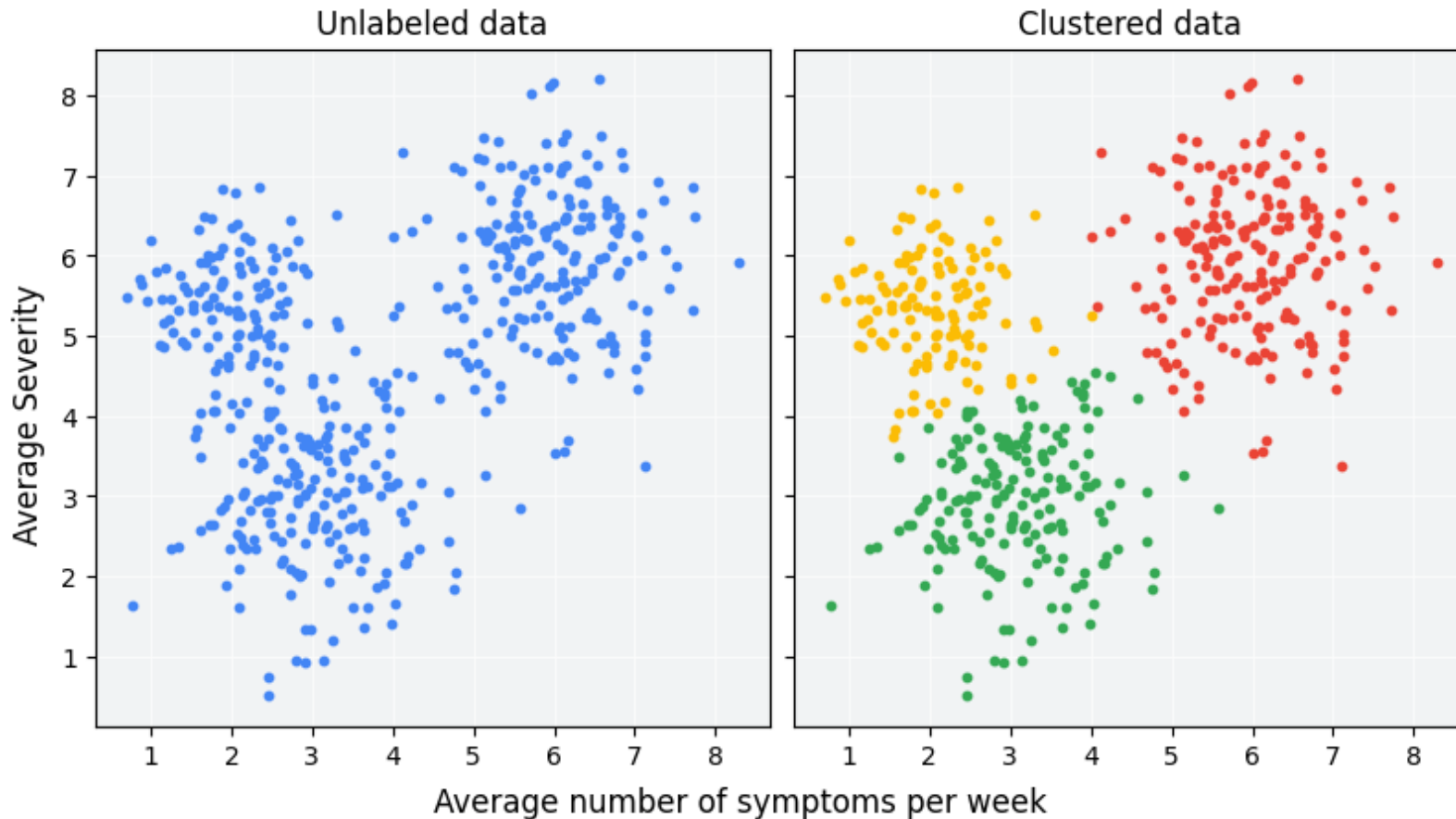
**Random Forests** are constructed from an *ensemble* of individual trees.

- Each tree in the ensemble uses a randomly selected subset of the feature space, and the minimum node size is usually set to 1, so the classifier prediction is almost always accurate.
- The mode (classification) or mean (regression) of the ensemble is the output of the Random Forest.



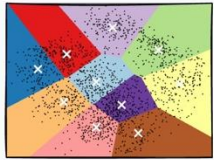
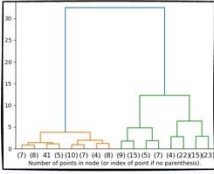
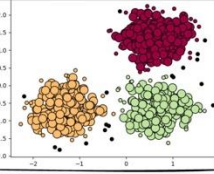
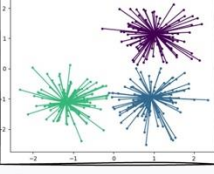
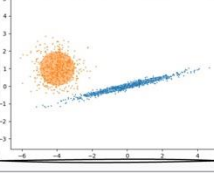
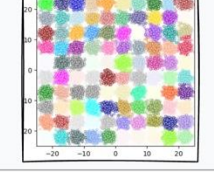
# Clustering

... is an unsupervised machine learning technique designed to group *unlabeled* examples based on their **similarity** to each other. (If the examples are labeled, this would be *classification!*)



**Similarity measure (function, metric)** –a function that quantifies the similarity between two objects.

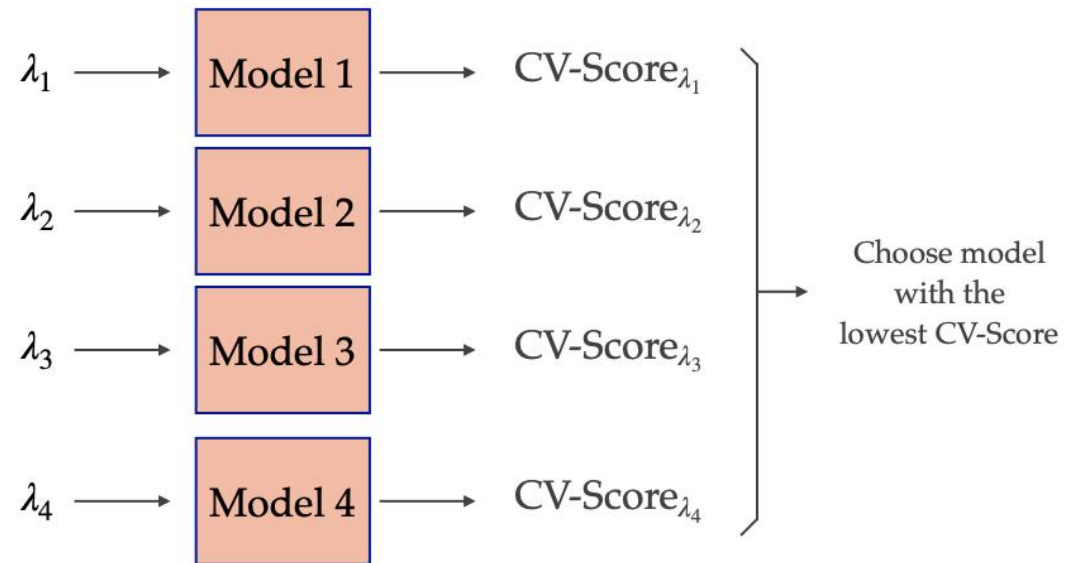
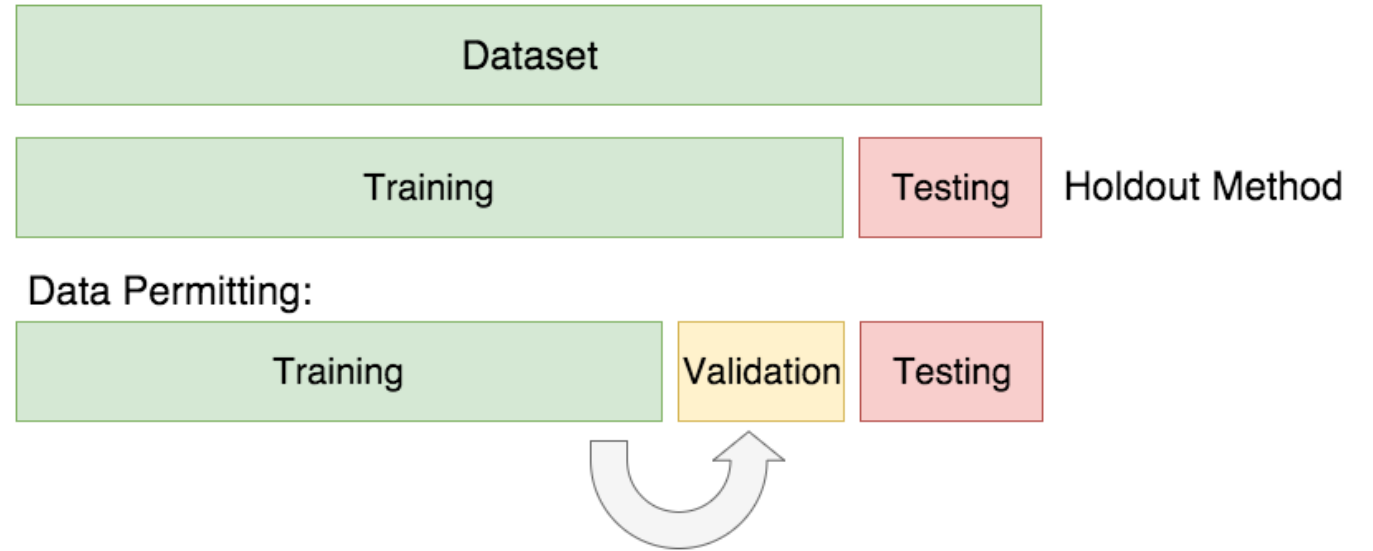
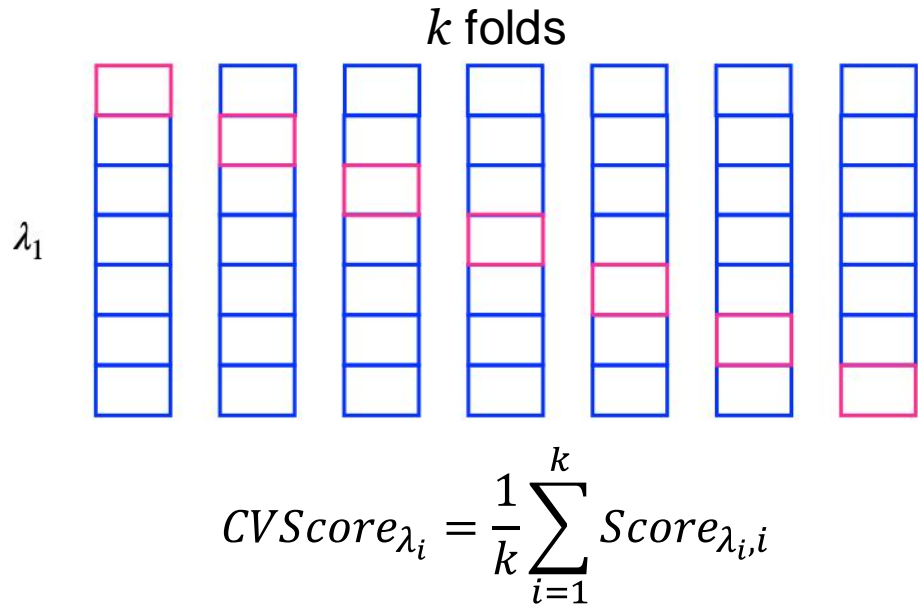
# Clustering

Clustering Algorithm Type	Clustering Methodology	Algorithm(s)	
	Centroid-based	Cluster points based on proximity to centroid	KMeans KMeans++ KMedoids
	Connectivity-based	Cluster points based on proximity between clusters	Hierarchical Clustering (Agglomerative and Divisive)
	Density-based	Cluster points based on their density instead of proximity	DBSCAN OPTICS HDBSCAN
	Graph-based	Cluster points based on graph distance	Affinity Propagation Spectral Clustering
	Distribution-based	Cluster points based on their likelihood of belonging to the same distribution.	Gaussian Mixture Models (GMMs)
	Compression-based	Transform data to a lower dimensional space and then perform clustering	BIRCH

# Model validation

**Data splitting** into *training*, [*validation*], *testing* sets helps ensure the creation of data models and processes that use data models are accurate.

To assess the quality of the model before applying it to unseen inputs (testing), we perform **cross-validation**.



# Hyperparameter optimisation (tuning)

... is a process of choosing optimal **hyperparameters** – parameters that control the learning process and that are defined before the learning process starts.

