LABORATÓRIO DE INSTRUMENTAÇÃO
E FÍSICA EXPERIMENTAL DE PARTÍCULAS
partículas e tecnologia

Machine Learning Tutorial

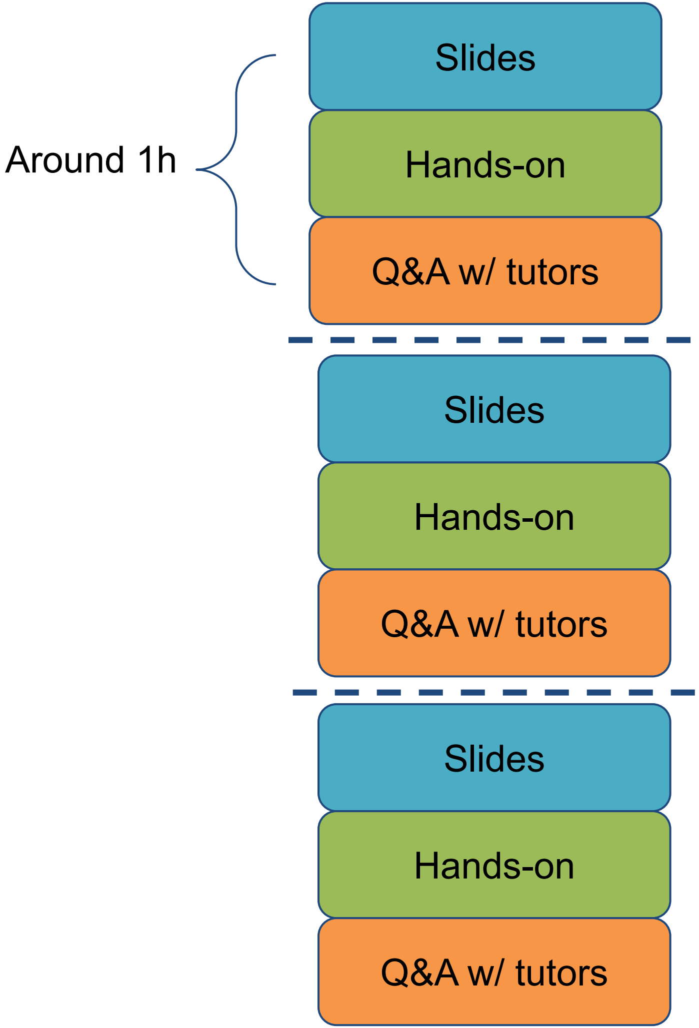
Based on Miguel Caçador Peixoto's Slides

LIP Internship Program - Summer 2025

Fernando Souza
abreurocha@lip.pt

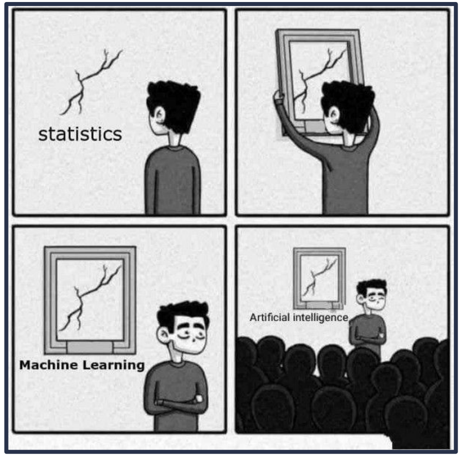
How this tutorial will proceed
General idea

- I will guide you through some concepts using these slides
- We will then move on to Google Colab where I will guide you through a hands-on code-along tutorial to explore the concepts
- After each coding block, there will be room for Q&A and clarifications



How this tutorial will proceed
outline

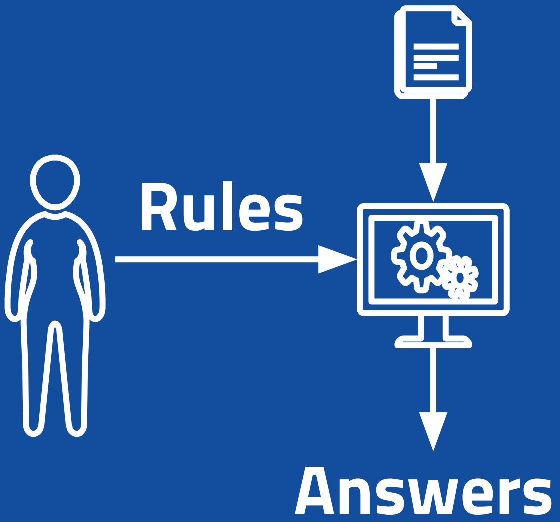
- Part I: What is Machine Learning?
  - Linear Regressions, Decision Trees, Evaluation Metrics
- Part II: Ensembles and Neural Networks
  - Forests, Deep Learning, Standardization, Regularization, Hyperparameters
- Part III: pp collisions dataset



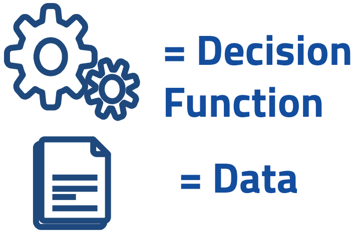
1 - What is Machine Learning?

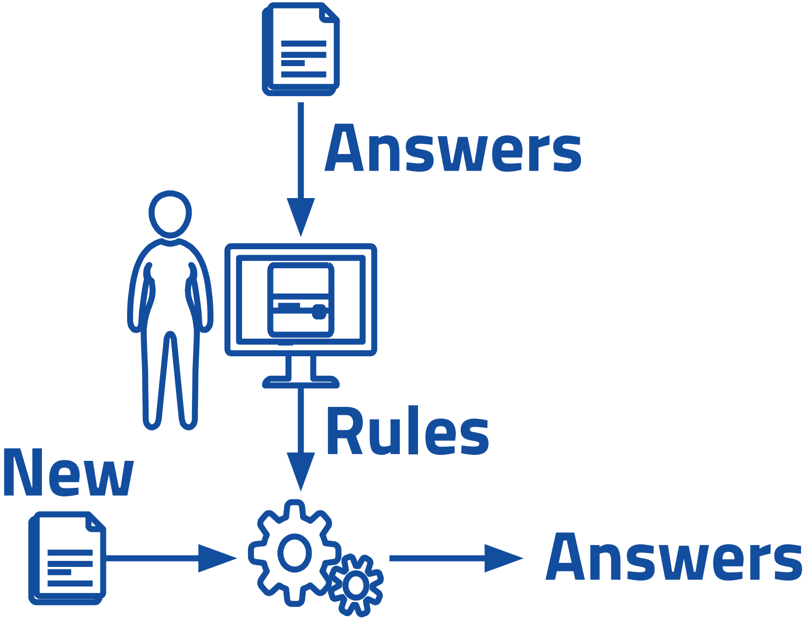
From an Artificial Intelligence Perspective

Classical Programming

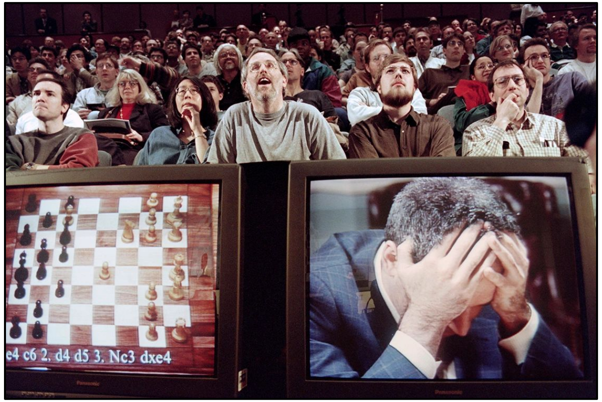


Machine Learning





In 1996 **IBM Deep Blue** beat **Garry Kasparov** in a six-game match (4-2)



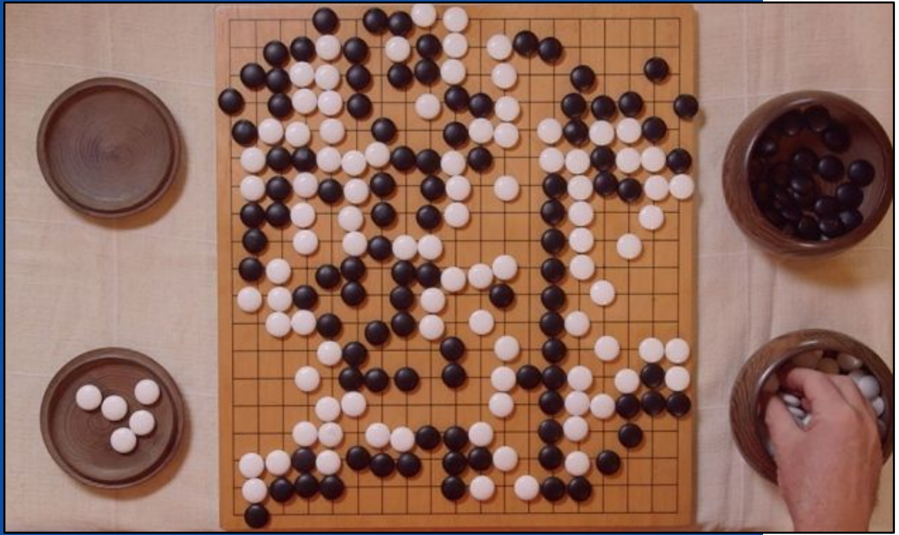
..but it wasn't Machine Learning!

AlphaGo

By DeepMind, Circa 2016

It beat Lee Sedol in a five-game match (4-1)

The Game of **Go**



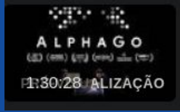
Possible board configurations?

- Chess - **10^{46}**
- Go - **10^{170}**

Number of atoms in the observable universe?

10^{82}

There is a movie about it!



AlphaGo - The Movie | Full award-winning documentary

YouTube · DeepMind

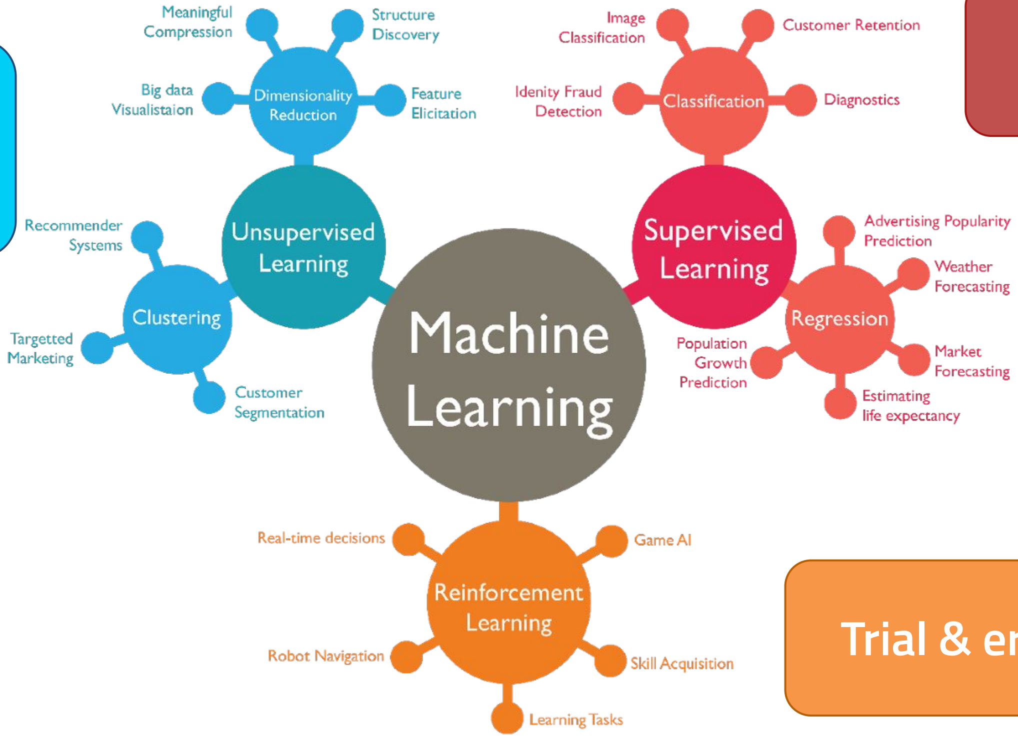
13/03/2020

Machine Learning Taxonomy

What is out there and what tasks can we solve?

Machine Learning
Taxonomy: Types of Learning

Infers patterns & predicts outputs



Learns from labeled data

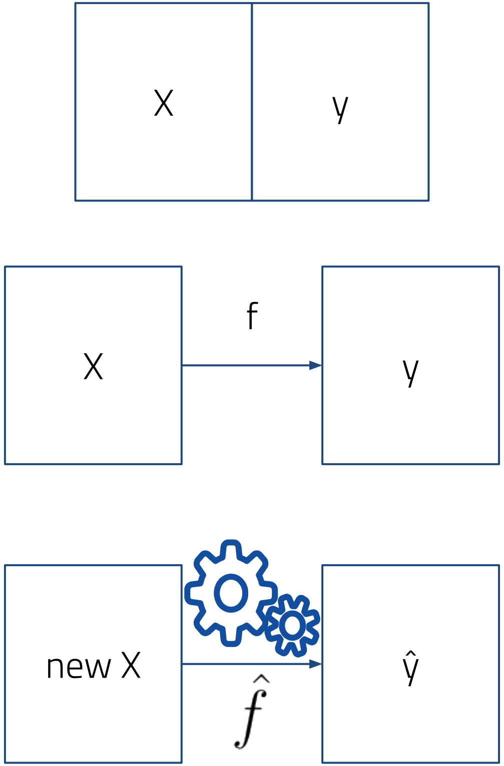
Trial & error

Machine Learning
Taxonomy: Supervised Learning

- The training data includes the answer we want to reproduce
  - $\mathcal{D} = \{(X_i, y_i)\}$
  - X: Independent Variables/Features
  - y: Target Variables/Labels
- Assume (hope?) there exists a relation such that

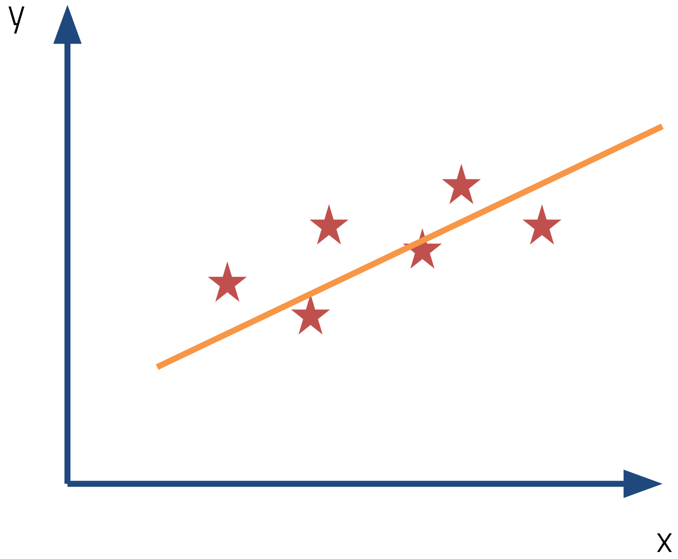
$$f : X_i \mapsto y_i$$

- The model will approximate f, $\hat{f}$
- The type of y defines two sub-classes
  - y is a real variable: **Regression**
  - y is categorical: **Classification**



Regression Example

Linear Regression



$$y = wx + b$$

ML = **optimization** using data

$$C(w, b) = \frac{1}{n}\sum_{i=1}^{n}(f(x_i, w, b) - y_i)^2$$

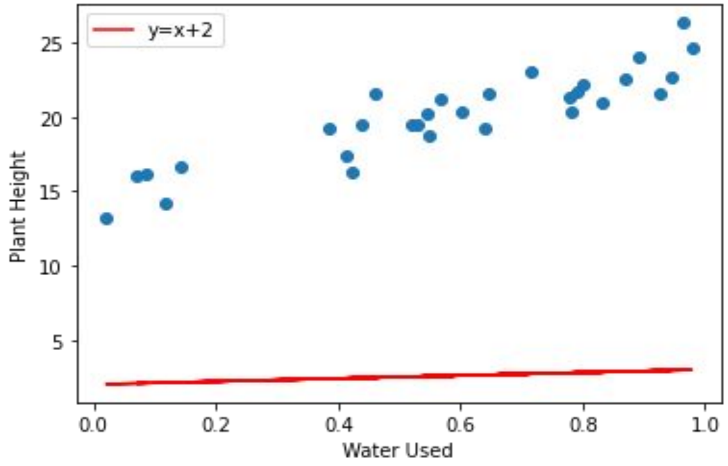
- C: cost function
- w: weights
- b: bias

Regression Example
Linear Regression

$$y = wx + b$$

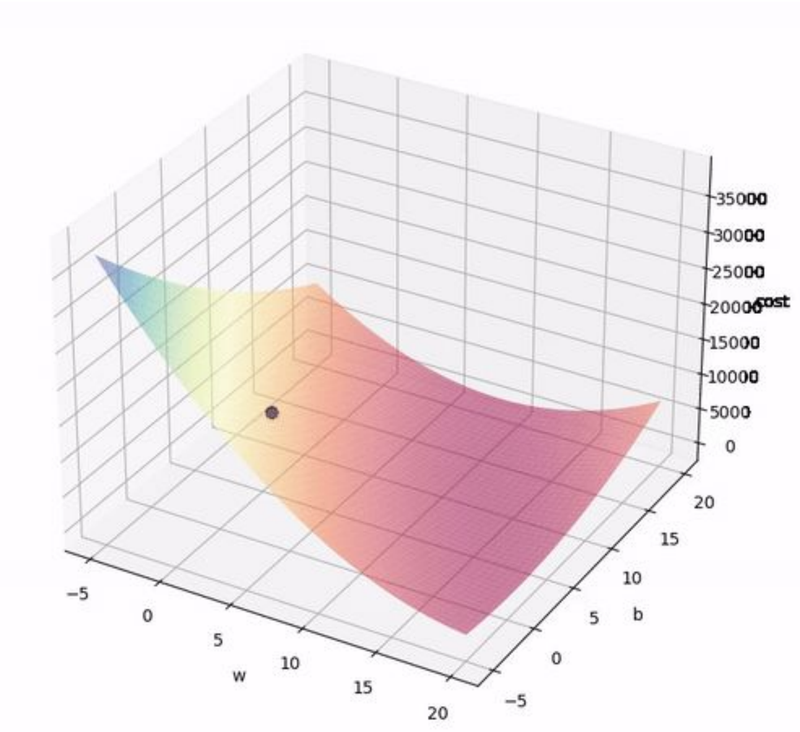
The Algorithm

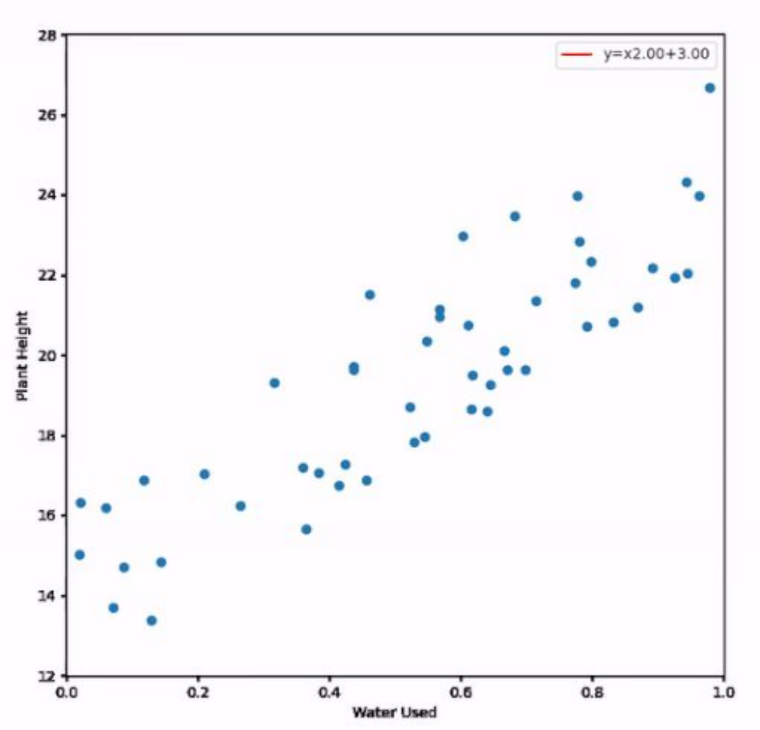
1. Let’s start with a guess. Let's say w=1 and b=2.
2. Calculate the gradient of our loss function for our parameters.
3. Update the parameters.
4. Go to step 2 and repeat until we’re satisfied.



$$\theta = \theta - \eta \nabla Cost$$

Regression Example
Linear Regression



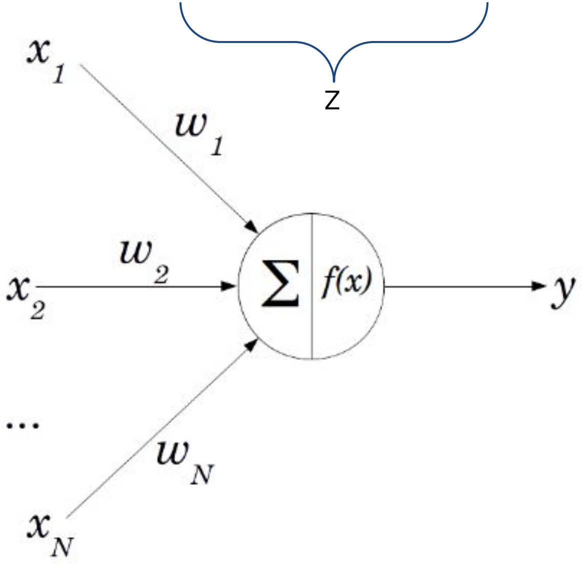


"This idea of taking small steps in the right direction is what is called Gradient Descent, and it's the heart of ML."

Classification Example

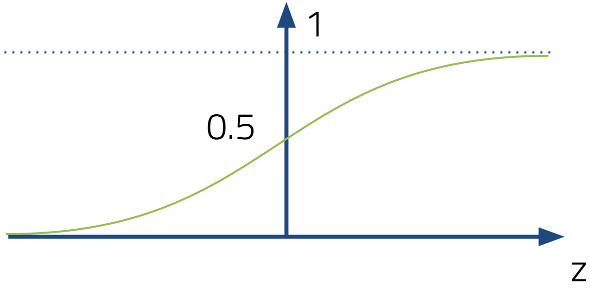
Logistic Regression Generalization

$$y = f\left(\sum_{i=0}^{N}(\boldsymbol{w_i} \cdot \boldsymbol{x_i}) + b\right)$$

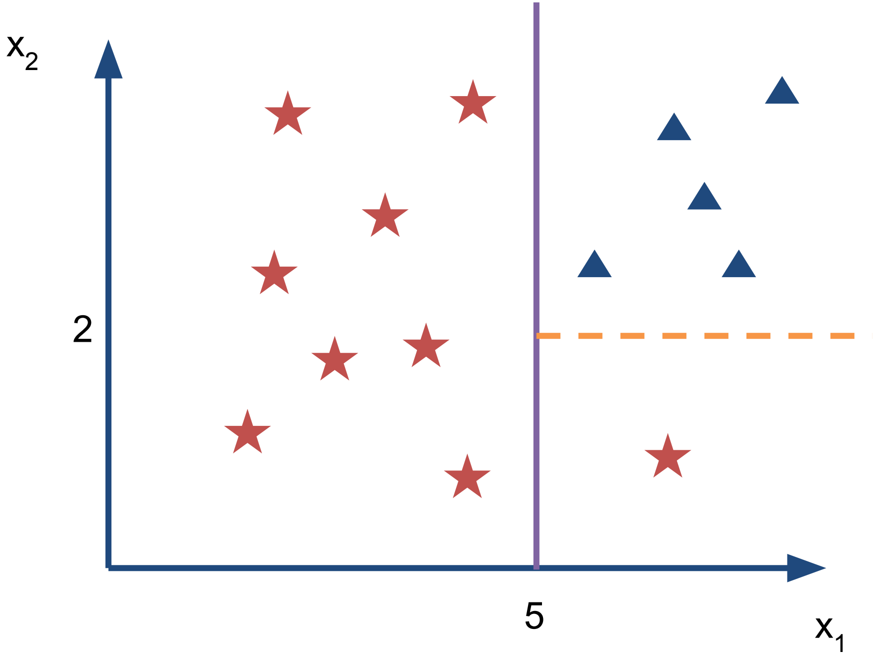


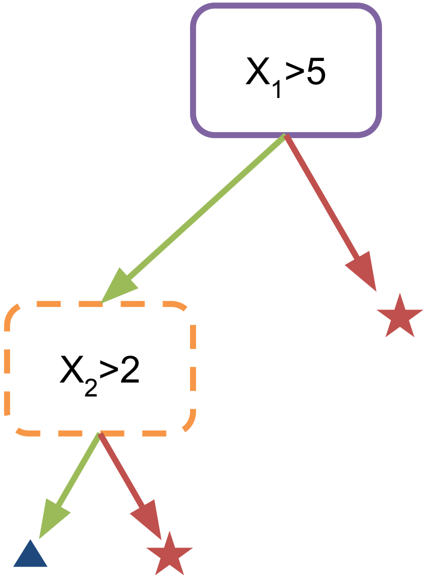
Sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Classification Example
Decision Tree





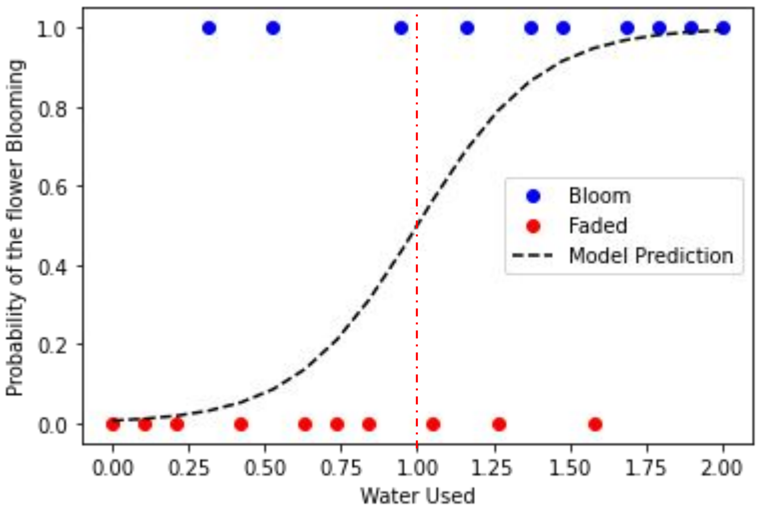
Classification Example
Decision Tree Training

- For each feature, order the points by their values
- Find a value for that feature that maximises purity of a class on each side of the split
- Repeat until there are no more splits left -- either all truncations are pure in one class or each data point is in its own leaf

Machine Learning
How to evaluate a classifier

- There are many metrics in the Machine Learning literature that help you assess the performance of a classifier
- We will be focus on two
  - **Accuracy**: The percentage of instances that are correctly classified
  - **Area under ROC** (Receiver operator characteristic) curve

Machine Learning
ROC



How good is this model?

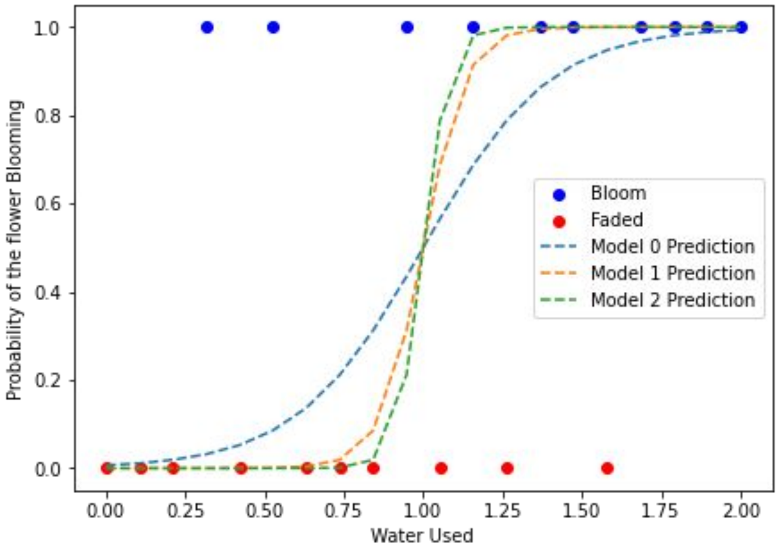
Just measure the accuracy!

If the output of the model is >0.5, then the flower bloomed (class 1),

Otherwise, the flower faded (class 0).

accuracy = 14 / 20 = 70 %

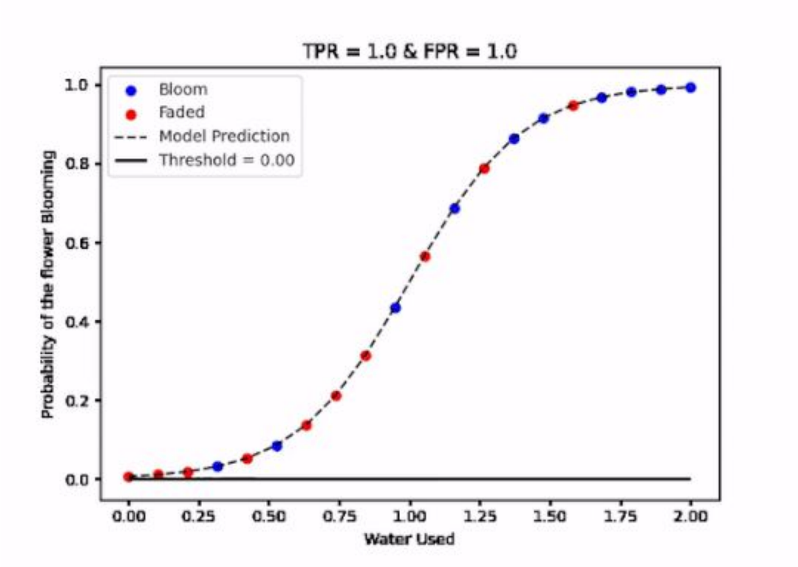
Machine Learning
ROC



-> They all have the same accuracy!

... we need a better metric.

Machine Learning
ROC



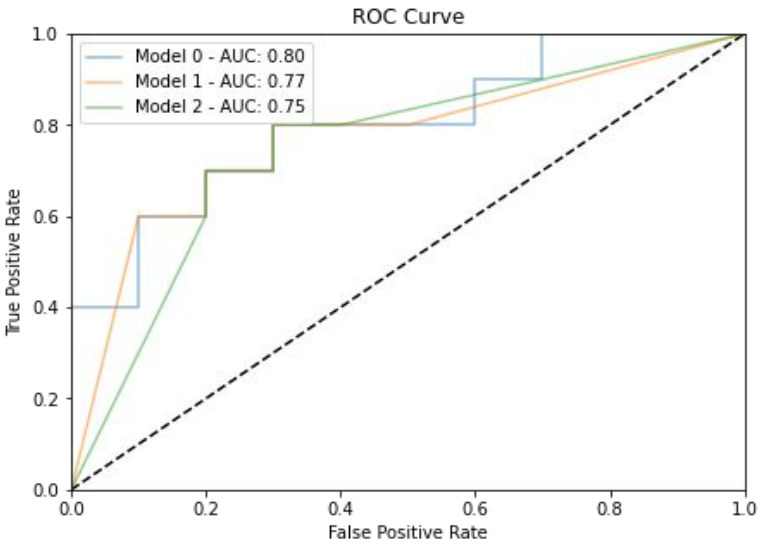
True Positive Rate = Sensitivity
False Positive Rate = 1-Specificity

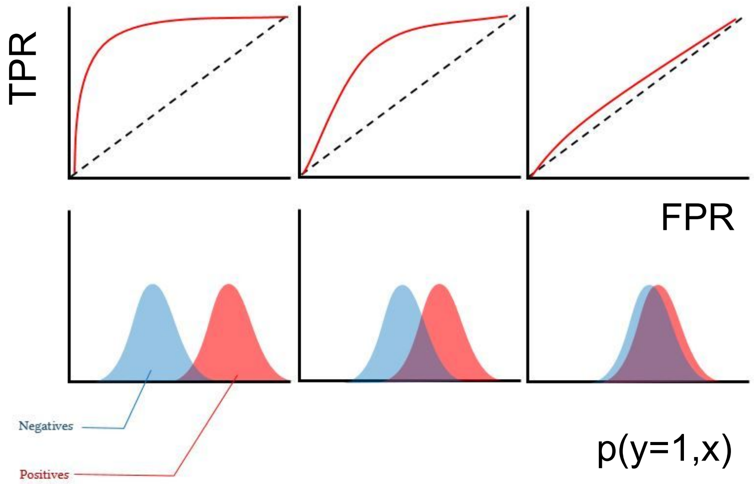
Predicted Class

		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
Actual Class	**Negative**	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Confusion Matrix

Machine Learning
ROC





Cheatsheet:

https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Google Colab

- An online jupyter notebook host solution where you can do Machine Learning in Python
  - https://colab.research.google.com/
  - You do need a Google account
- It has all the relevant packages to do Data Science and Machine Learning pre-installed
- You can use GPU and TPU acceleration, for free

Scikit-Learn
and the python Machine Learning ecosystem

- Scikit-Learn (https://scikit-learn.org/) is the go-to ML package for python
- It defined the best practices for ML API development
- Has great documentation and tutorials
- **If this tutorial fails to teach you anything...**

  learn ML from Scikit-Learn documentation!

Additional Packages
For the python Machine Learning ecosystem

- We will start by implementing a logistic regression and a decision tree
  - `sklearn.linear.LogisticRegression`
  - `sklearn.tree.DecisionTreeClassifier`
- Not estimator modules worth remembering:
  - `sklearn.preprocessing`
  - `sklearn.model_selection`
  - `sklearn.metrics`

1st hands-on

We will use Google Colab to run a few examples of classification algorithms using Scikit-Learn

2 - Ensembles and Neural Networks

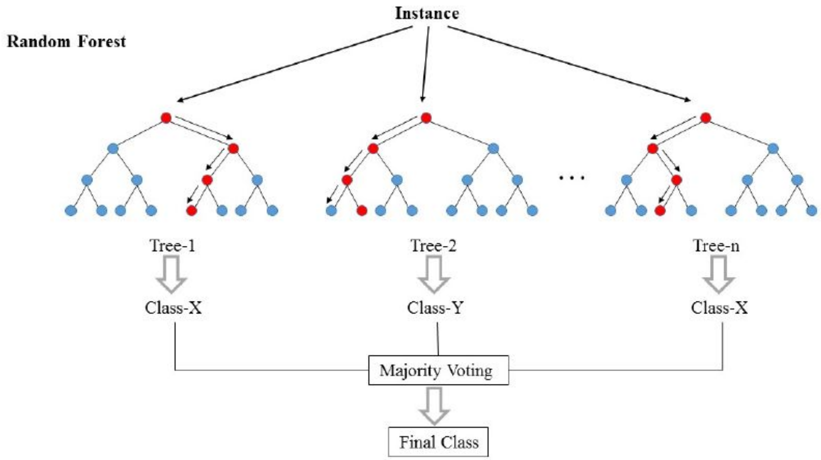
Forests, neurons, and all that jazz

Ensembles
Strength in numbers

- An Ensemble is an... ensemble of ML models
- The idea is that the many **weaker learners perform better together and produce a stronger learner**

Ensembles
Strength in numbers

- Example: **Random Forest** is a collection of smaller trees (with a maximum depth) trained on subsamples of the data (bootstrapping)
  - The final prediction is given by average of the predictions -> This gives better generalisation than using a big tree alone



- Parallel Training
- Strong Predictive Power

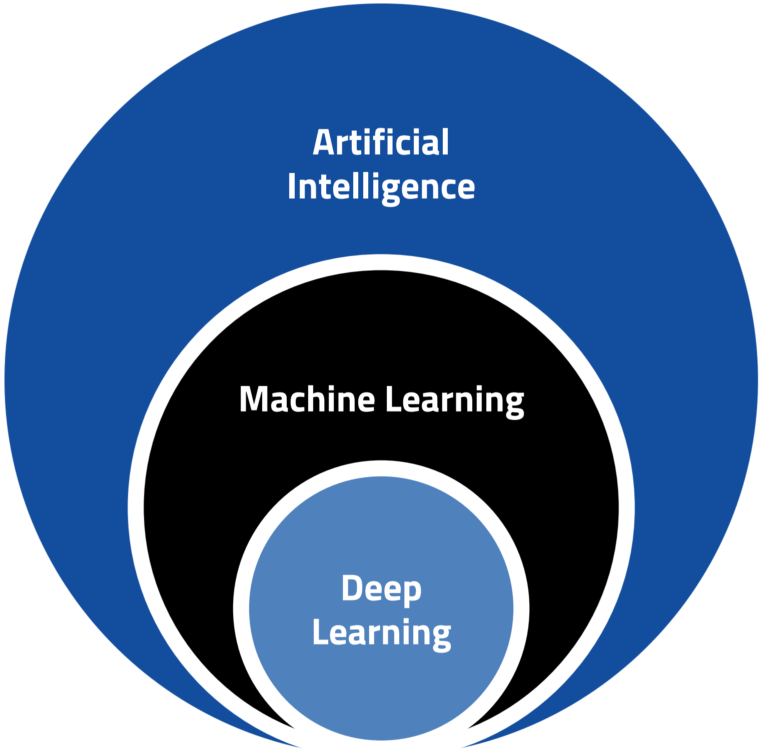
Ensembles
Come in different shapes

- Although most of the ensembles techniques are based on Trees as the base model, there are many ways of building
  - I already mentioned Forests (a type of Bagging)
  - Another famous class are the Boosted ensembles (e.g. Boosted Decision Trees and Gradient Boosted Trees):
    - A sequence of trees that learn progressively more difficult cases
    - XGBoost

Ensembles
They are better than individual models

- Ensembles of Trees are **very good baseline models** and should be your **first go-to choice for tabular data** (i.e. excels, csv, etc)
- They improve generalisation of the base estimator and reduce the risk of overfitting
- They **require little to no data preprocessing** (when based on Trees), making them very attractive as out-of-the-box solutions

Deep Learning is a subclass of Machine Learning algorithms that train Neural Networks to perform tasks



Deep Learning and Neural Networks

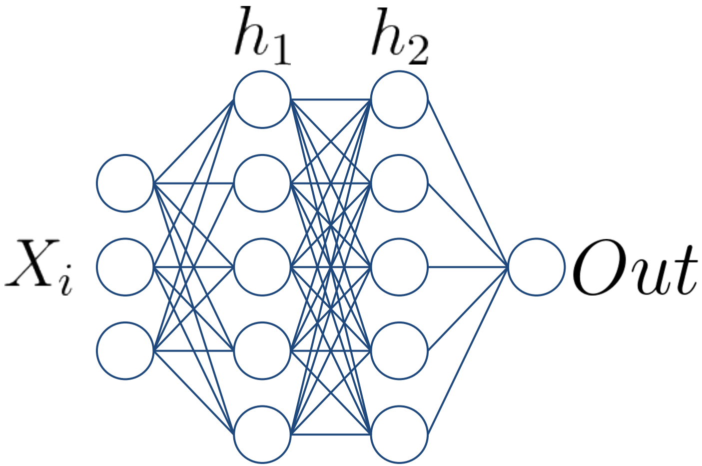
Terrible name, great idea

Differentiable models that can be trained with **Stochastic Gradient Descent**

Unmatched **representational power** and are capable of **feature abstraction**: deeper layers abstract more complex relations

Extremely versatile and can take in **data of many different shapes and formats**

All state-of-the-art Machine Learning applications are based on Deep Learning and implement Neural Networks



$$\vec{h}_1 = a_1(\mathbf{w_1} \cdot \vec{x} + \vec{b}_1)$$

$$\vec{h}_2 = a_2(\mathbf{w_2} \cdot \vec{h}_1 + \vec{b}_2)$$

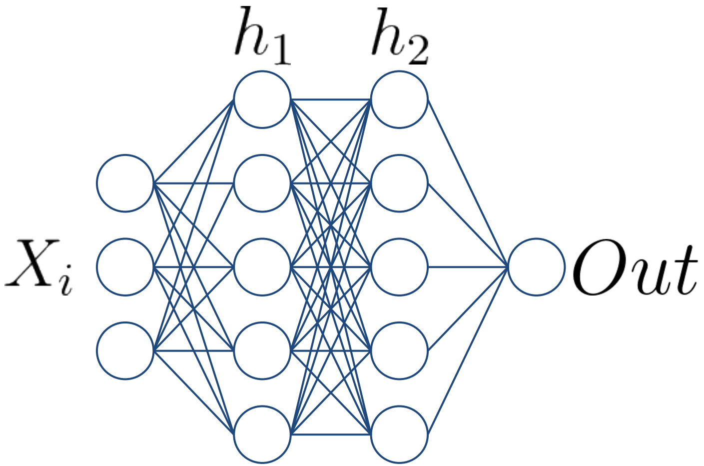
$$Out = a_{Out}(\vec{w}_{Out} \cdot \vec{h}_2 + b_{Out})$$

$$a_i = \{\tanh, \sigma, \mathrm{ReLu}, \dots\}$$

$$NN = Out \circ \vec{h}_2 \circ \vec{h}_1$$

Deep Learning and Neural Networks
Defining and training

- Define how many layers and how many units (neurons) are in each layer, in addition to the non-linear activation
- Define the output
  - For binary classification: sigmoid
- Define the Loss function
  - For binary classification: binary cross-entropy
- Iteratively train on mini-batches of data. This is performed by an optimisation algorithm (we won't be able to cover these in detail)



$$\vec{h}_1 = a_1(\mathbf{w_1} \cdot \vec{x} + \vec{b}_1)$$

$$\vec{h}_2 = a_2(\mathbf{w_2} \cdot \vec{h}_1 + \vec{b}_2)$$

$$Out = a_{Out}(\vec{w}_{Out} \cdot \vec{h}_2 + b_{Out})$$

$$a_i = \{\tanh, \sigma, \mathrm{ReLu}, \dots\}$$

$$NN = Out \circ \vec{h}_2 \circ \vec{h}_1$$

Deep Learning and Neural Networks
Preprocessing: Standardisation

- Unlike trees, Neural Networks require some preprocessing
- The most common requirement is to standardise the inputs: **set mean to 0 and standard deviation to 1**

$$X \to \frac{X - \bar{X}}{\sigma_X}$$

- The reason for this is that the SGD applies weight updates layer-by-layer (chain rule over function composition), and too large activations will lead to too large updates => **gradient explosion and unstable learning** (see also **vanishing gradients**)
- Scikit-Learn is your friend
  - `from sklearn.preprocessing import StandardScaler`
  - `from sklearn.pipeline import make_pipeline`

Neural Networks
In python

- Scikit-Learn has a simple implementation of a Neural Network for classification (usually called a Multi-Layer Perceptron)
    - `from sklearn.neural_network import MLPClassifier`
- But we will look into more famous frameworks:

Neural Networks
Are the present and the future

- Neural Networks have unleashed a revolution in Machine Learning
- Getting them to work requires some work and care, but the outcome is usually worth the trouble
- This is by no means a complete introduction, I recommend investing some time with documentation of the modules covered and some books:
    - **The 100 Page ML book**; Hands On ML With Scikit-Learn, Keras & Tensorflow; Deep Learning with PyTorch

Neural Networks
In python using TensorFlow/Keras

- We will use Keras packaged with TensorFlow
- A model is initiated with a Model class. We will use the Sequential
  - It takes a sequence of layers (classes from the layers module)
  - It connects them automatically sequentially

```
model = keras.models.Sequential([
     keras.layers.Dense(100, activation='relu', input_shape=(2,)),
     keras.layers.Dense(1, activation='sigmoid')
    ])
```

- You then compile to define the Loss function, metrics, and the optimizer

```
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy', keras.metrics.AUC()])
```

- Which you can then fit

```
model.fit(X_train, y_train, epochs=100)
```

How SGD is implemented. Adam is always a good first choice

Model choice and Hyperparameter Tuning

Neural Network shape

- How does the shape of the network affects its performance?
  - The deeper (more hidden layers) and wider (number of units) the greater is the capacity
- The performance of the Neural Network can also be affected by the choice of non-linear activation function
- How to choose?
- Is there a risk of using too large a network?

Sigmoid

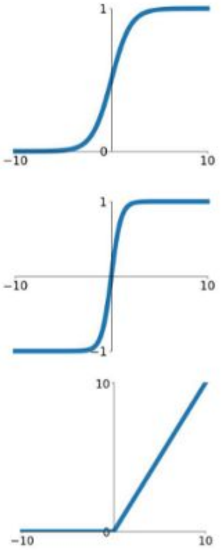
$\sigma(x) = \frac{1}{1+e^{-x}}$

tanh

$\tanh(x)$

ReLU

$\max(0, x)$

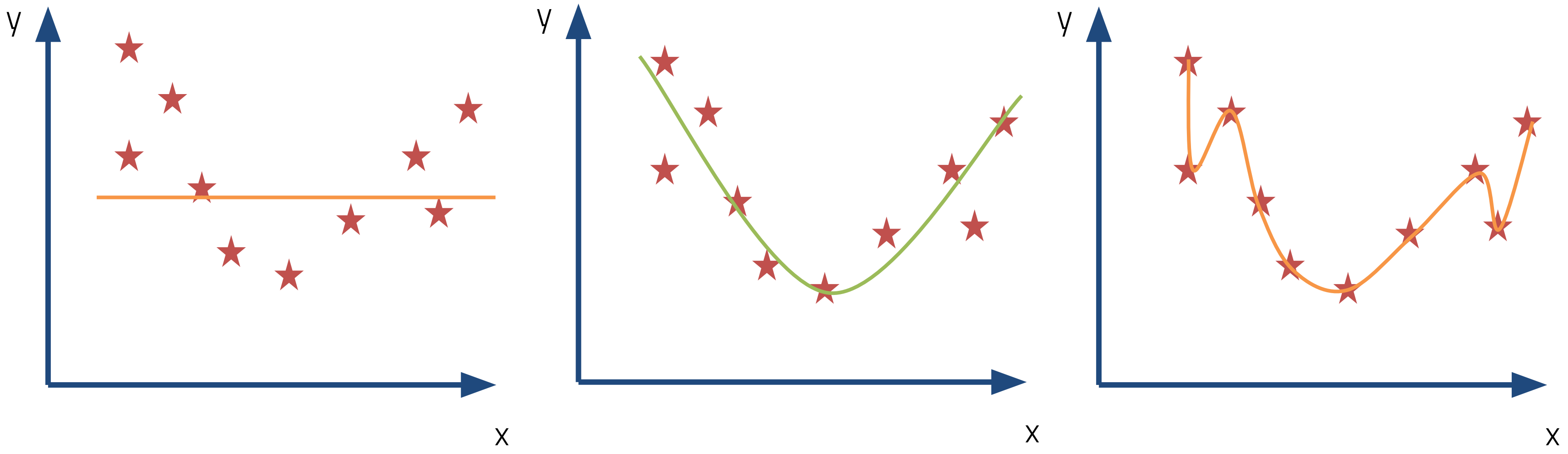


Model choice and Hyperparameter Tuning

Model Capacity

A model with insufficient capacity will fail to fit f: **underfitting.**

A model with too much capacity will fit the noise: **overfitting.**

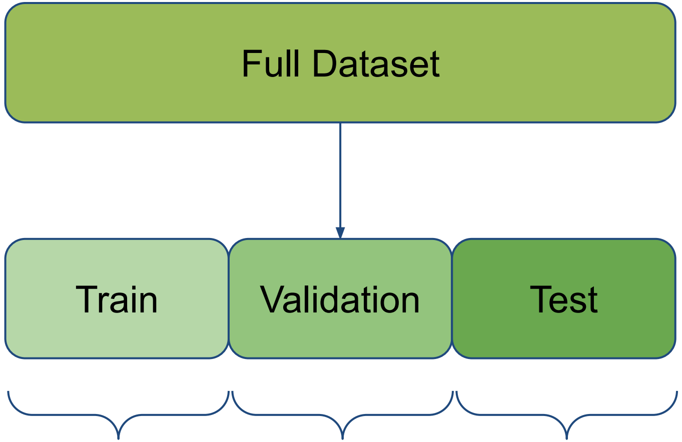


Regularisation

In practice, one usually overestimates the capacity needed and then applies regularisation to prevent overfitting

Model choice and Hyperparameter Tuning
Best practices: Three different splits!

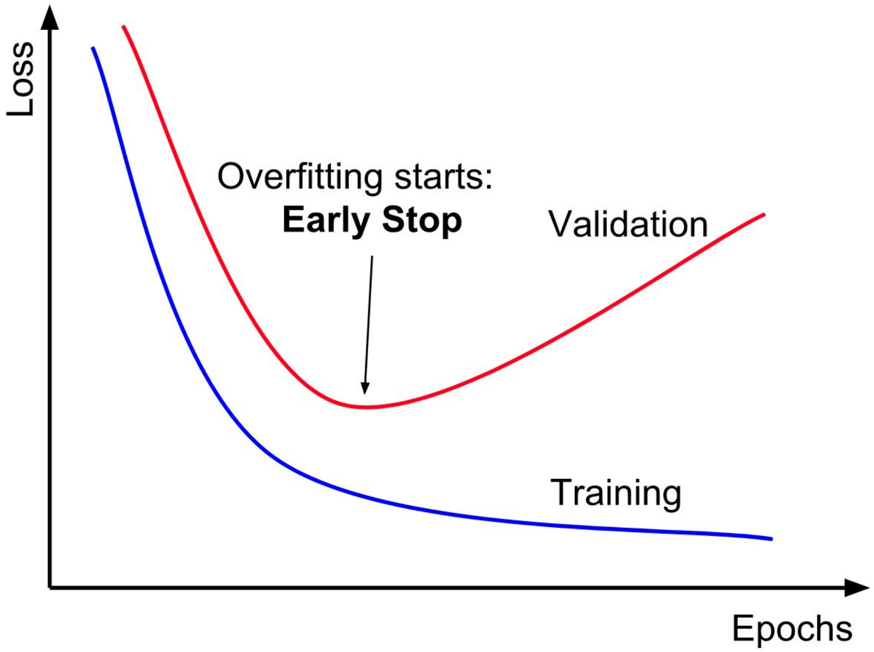
- Split the dataset into three sets
    - Train: for fitting
    - Val: for validation
    - Test: to derive the final performance
- **Never use the Test set at any stage of your training or validation => Information Leakage (a.k.a. cheating)**



In our case we want to retain a good statistical description of our data
1:1:1

Model choice and Hyperparameter Tuning
Regularisation

- Many ways of regularising a ML model, which depend on the type of algorithm
- One that always helps with Neural Networks (and other iteration-based training algorithms) is **early stop**
    - Stop training when the loss/metric worsens on a validation set



Model choice and Hyperparameter Tuning
Choosing the final hyperparameters

- Try different combinations of hyperparameters. **For each:**
  - Train the network with the training set
  - Use the validation set to stop early
  - Measure the metrics on the validation set
- In the end: pick the **hyperparameter combination** with the best validation set metrics
- If you learn how to do this you can become a professional Machine Learning engineer in the industry

2nd hands-on

Let’s implement some ensembles and neural networks using both Scikit-Learn and TensorFlow

3 - Finding new Physics signals

Because you only learn by doing

Machine Learning in New Physics Analyses
Finding a needle in a particle haystack

- Now that you are proficient Machine Learning engineers, let's do some physics with this!
- The idea is simple:
  - Data come
  - Data might have a signal we want to discover
  - Train a classifier to separate interesting events from the background
  - Make a discovery and profit (joking, someone else gets the Noble)

Simulated pp collisions Dataset

https://zenodo.org/record/5126747

- Created in 2021, the dataset is composed of different Beyond the standard model events (Signal) and Standard-Model events (Background)
- The objective is to isolate as much signal as possible (Classification problem)

The pp collisions Dataset
A few words on weights...

- The dataset is **simulated** (Monte Carlo)
- In order to be sure that we are covering a full description of the simulated event we often **simulate far more events than those expected**
- Furthermore, each event has different probabilities of happening (given by the **cross-section**)
- In the end the simulation is composed of different simulated events at different rates, and we need to **reweight** their contribution in order to **keep the statistical description** of the data

3rd hands-on