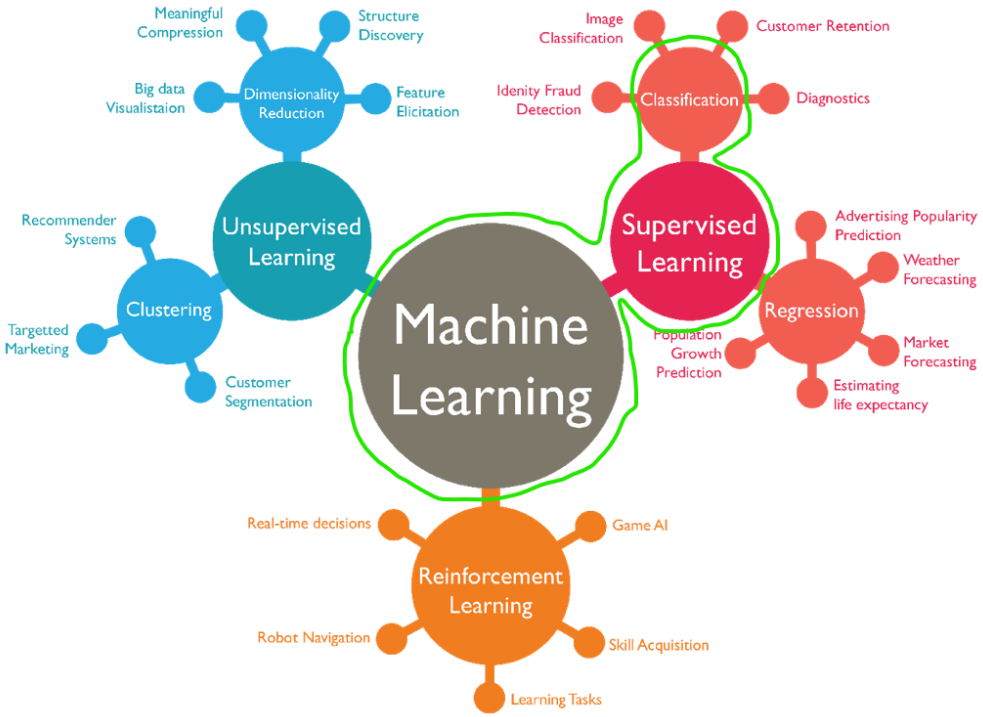
Some basic and essential concepts for this tutorial

On machine learning

Machine learning is a big field. Here is a mind map that shows different types of machine learning algorithms with their typical applications. It highlights the main three types of machine learning algorithms: supervised learning, unsupervised learning, and reinforcement learning. They differ basically on how do you use the dataset to train your algorithm.



(Green Line indicates our focus in this tutorial. Source: https://vitalflux.com/great-mind-maps-for-learning-machine-learning/)

During these tutorials, we will be working on supervised learning for classification tasks. Supervised learning means that an expert has labelled your data with outcomes (labelled training data). A classic example is image classification, where for example an expert has labelled a bunch of images of dogs and cats and then trained a machine-learning algorithm to predict if a new image it's a dog or a cat Then, you could feed this trained algorithm with new pictures of cat and dogs and it could quickly classify them (one drawback here is that if you feed the algorithm with a picture of yourself, the algorithm will take a chance and classify you like a cat or dog, which is wrong even given the similarities between you and our most beloved pets...).

Another big category in machine learning is the family of unsupervised learning algorithms, which basically mean no labels (no expert telling you what's the difference between the data). If you have heard 'data mining', this is what's they are talking about. They are used to extract patterns in datasets. For example, if I feed this type of algorithms with the cats and dogs pictures, and tell it that there might be two things there, it might classify the pics in two clusters by learning to distinguish some differences in the data (without knowing that it is classifying cats and dogs). Again, then you

could feed this trained algorithm with new pictures of cats and dogs and it could quickly classify them into one of the two clusters.

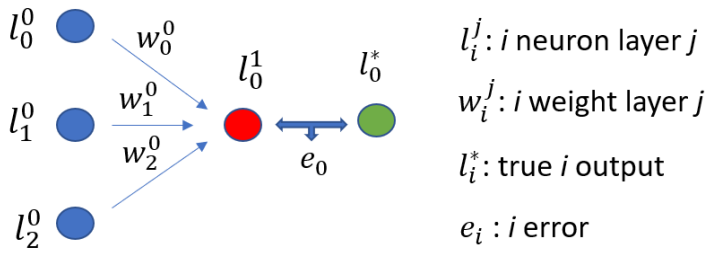
Check this video for a much better introduction of each type of machine-learning algorithm: https://www.youtube.com/watch?v=YlGEQyEM_a8.

On neural networks

What is a neural network?

It's just an algorithm that tries to discover patterns in input to output transformation, like a mathematical function, by just observing the true inputs and outputs of that transformation. It is supposed to mimic the human brain, but actually, we don't know too much about how the brain works, so... The connection comes from the phrase 'neurons that fire together wire together' taken from a popular theory on how synapses work (Hebbian Theory), and in some way inspire the whole neural network development in machine learning. However, machine learning neural networks (NN) are much simpler than the brain, so don't be afraid.

The essential concept is that NN map an input in a layer of nodes or **'neurons'**, and connects them to outputs through wires that we call **'weights'**. Let's take the following simple two-layer NN that transforms 3 inputs into 1 output.



How does the NN calculate an output (l_0^1) from the inputs (l_0^0, l_1^0, l_2^0)? By calculating the dot product between the inputs and the weights (w_0^0, w_1^0, w_2^0), and passed it through a handy function call **sigmoid** (another typical **activation** function if the **relu** function).

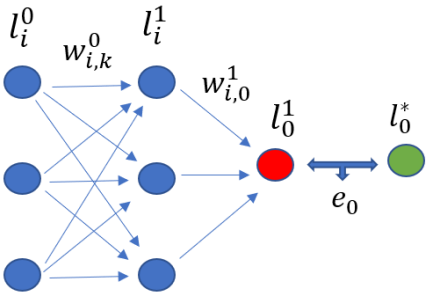
$$l_1 = sigmoid((l_0^0 \, x \, w_0^0) + (l_0^1 \, x \, w_0^1) + (l_0^2 \, x \, w_0^2))$$

This function transforms the dot product result into a number between 0 and 1. We call **forwardpropagation** the act of of transforming inputs into outputs using the weights and values at the neurons.

Now let's consider that we have a **dataset** of true inputs and outputs. We take the input, map it into the first layer of the network and by forwardpropagation estimate an output. Then, we can compare the estimated output with the true output, the one given in the dataset, and calculate an **error**. NN uses that error to calibrated the weights in the hope of better match the true output. The process of **update** the weights using the error information is called **backpropagation**. Backpropagation is the escence of NN. Backpropagation would be the focus of the second tutorial, where we will be coding it from scratch.

For **deep** neural networks (once with more than two layers or '**hidden'** layers) the principals are the same. What changes is that backpropagation update weights for the whole layers and it does it using the calculated error in the output layer and then backpropagated layer by layer. Backpropagation for

deep neural networks would be the focus of the third tutorial, where we will be coding it from scratch.



l_i^j: *i* neuron layer *j*

l_i^*: true *i* output

e_i : *i* error

$w_{i,k}^j$: weight from layer *j, from neuron I to k in layer j+1*

Here a great explanation on the basics of neural networks:

https://www.ibm.com/cloud/learn/neural-networks

For a remarkable explanation on the math behind NN, visit this youtube channel:

https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

Open up the first tutorial and let's begin!