**Content**

[1. Terminology 2](#_Toc20152287)

[1. Gradient Descent 2](#_Toc20152288)

[1.1.1. What is the Gradient? 2](#_Toc20152289)

[1.1.2. Back Propagation 2](#_Toc20152290)

[1.1.3. Learning Rate 3](#_Toc20152291)

[1.1.4. The Complete Mathematical Notation of a Fully Connected Neural Network 3](#_Toc20152292)

[2. Entropy and Information Gain 4](#_Toc20152293)

[2. Technologies 5](#_Toc20152294)

[1. Jupyter Notebooks 5](#_Toc20152295)

[2.1.1. %matplotlib inline and other ipython magic functions 5](#_Toc20152296)

[2.1.2. Markdown 6](#_Toc20152297)

[2. ML and DL Frameworks – Overview 6](#_Toc20152298)

[3. Scikit-Learn 7](#_Toc20152299)

[2.3.1. Pandas 7](#_Toc20152300)

[2.3.2. NumPy 10](#_Toc20152301)

[3. One-class classification (OOC) 11](#_Toc20152302)

[1. PU learning 11](#_Toc20152303)

1. Terminology

* Overfitting - the splitting process results in fully grown trees until the stopping criteria is reached. But, the fully grown tree is likely to overfit data, leading to poor accuracy on unseen data.
* Variance
* Pruning
* Greedy algorithm – always makes the choice that seems to be the best at the moment. This means that it makes a locally-optimal choice in the hope this choice will lead to a globally-optimal solution.
* Correlation
* Epoch – number of times you give data to the neural network. Less epochs underfit, more epochs overfits the data.
* Batch Size – the number of samples that will be passed through to the network at one time. 1000 Images; 10 batch size -> 100 batch will be needed for 1 epoch. Larger batches = faster training.
  1. Gradient Descent

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. If, instead, one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent.

Chain rule – derivative of nested functions

Calculate the back propagation for a fully connected neural network.

* + 1. What is the Gradient?

Gradient shows us the maximum of the function.

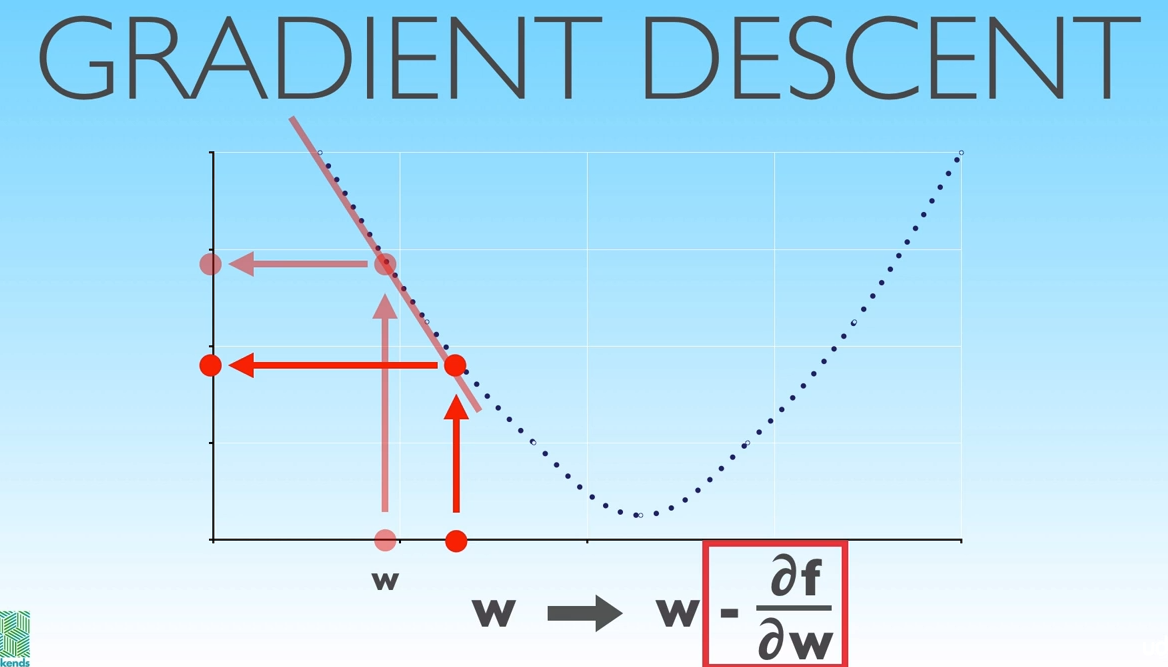
Derivatives: slope, rate of change

Gradient: extends derivative to multivariate functions.

* + 1. Back Propagation

Csokkeno fuggveny - derivacio negative

Novekvo fuggveny – derivalt pozitiv



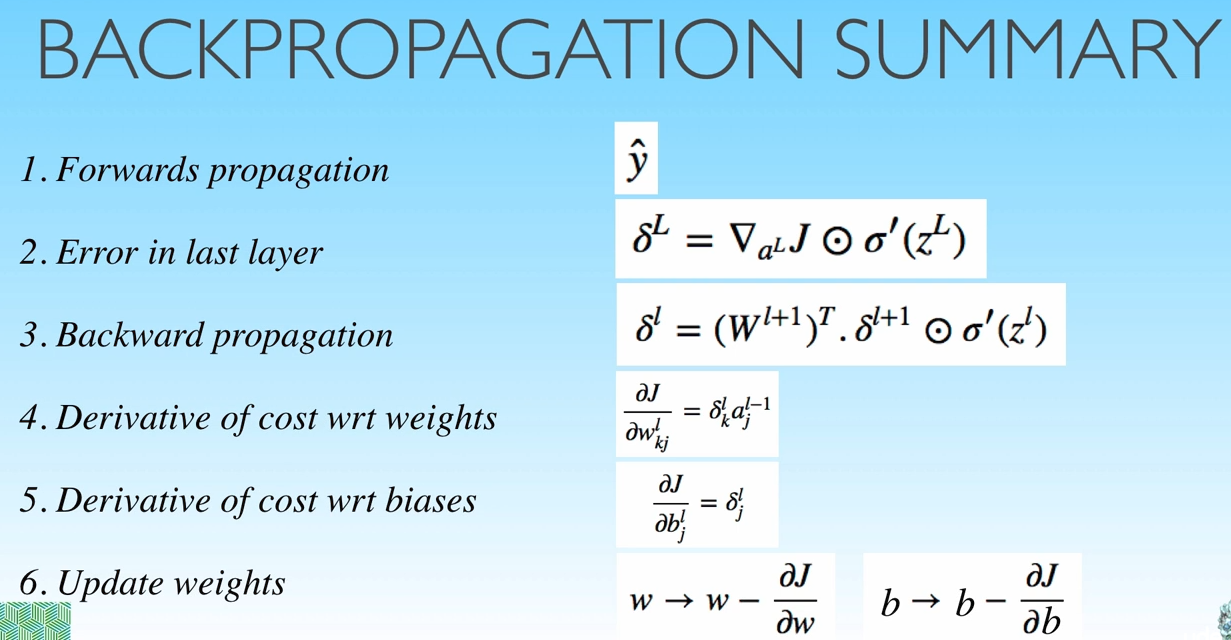
For a neural network we define cost function, that depends on the values of the parameters. We find the values of the parameters by minimalizing the cost by gradient descent. All we are really doing is taking the cost function, calculating its partial derivatives with respect to each parameter and then using the update rule (x := x – df/dx) we decrease the cost by updating the parameters we do this by **subtracting the value of the negative gradient from each of the parameters.**

* + 1. Learning Rate

If the function is very flat we move very slowly towards the minimum (using the update rule) but if the function is very steep we can jump over the minimum.



* + 1. The Complete Mathematical Notation of a Fully Connected Neural Network

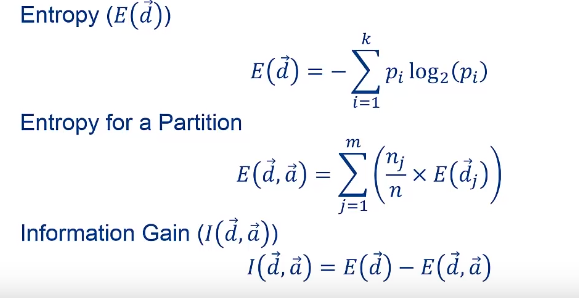


* 1. Entropy and Information Gain

*Entropy is a measure of disorder in a dataset.*

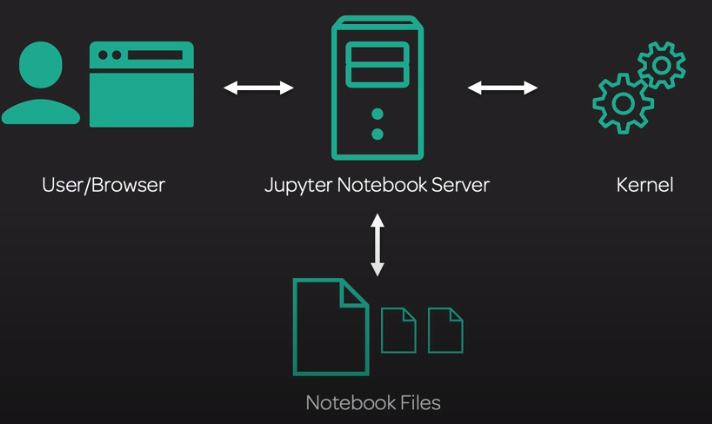
*Information gain a measure of the decrease in disorder achieved by partitioning the original dataset based on some additional attributes.*

We will use the concept of entropy, borrowed from physics, to build a classification method.



Entropy: why ‘-‘? Just because we want in to measure in positive. More disorder will be a larger positive value than a negative. Logarithmic function in interval (0 – 1) will be negative.

1. Jupyter Notebooks



Jupyter Notebook (formerly Ipython (Interactive Python) Notebooks) is a web-based interactive computational environment for creating Jupyter notebooks documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A **Kernel** is a configuration, a type of environment, which you have on the server. When you create a notebook you have to choose, which kernel you want to run. They are like python environments. Using python environments is a good habit, so your different projects will not interfere with each other.

To shutdown a notebook (not consume resources)

* + 1. %matplotlib inline and other ipython magic functions

*Provided you are running IPython, the %matplotlib inline will make your plot outputs appear and be stored within the notebook.*

Magic functions work more like OS CLI calls.

With this backend, the output of plotting commands is displayed inline within frontends like the Jupyter notebook, directly below the code cell that produced it. The resulting plots will then also be stored in the notebook document.

* + 1. Markdown

1. # - Header
2. ## - smaller header
3. \_\_abc\_\_ - strong / bold
4. ```![Pinehead](pinehead.jpg)``` --way to write code samples
5. “1. 2. 3.” – lists, ordered
6. “ - , - “ lists, unordered
7. \_ -- italic
8. ![I'm not heavy, I have dense feathers!](./animal-animal-photography-cold-86405.jpg)

Insert an image.

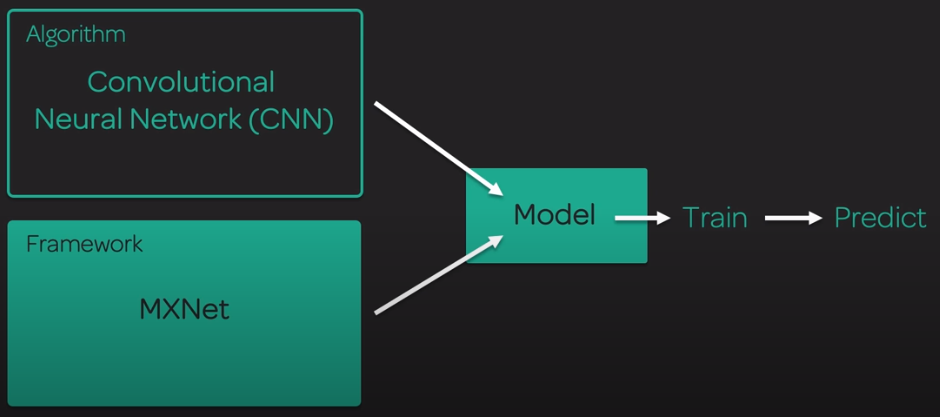
! – to run commands

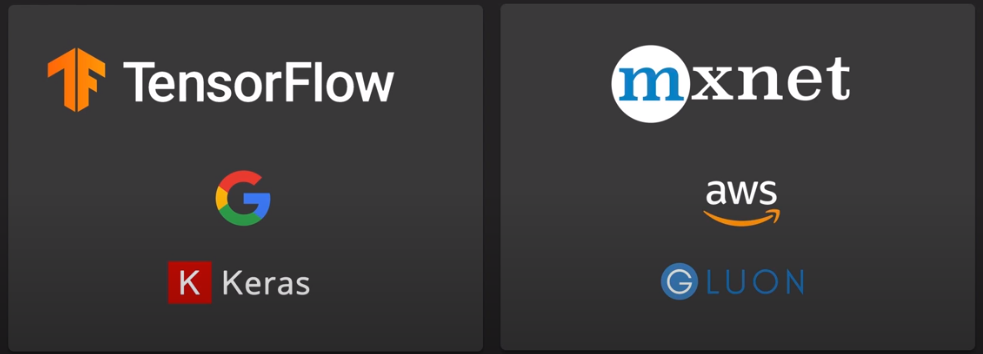
1. ML and DL Frameworks – Overview



What’s the difference between an algorithm and a framework?

Framework is a set of APIs, set of libraries, set of something that someone have already done to do the jobs you will need to do anyways to create an algorithm. The algorithm is based on Math. So we install python, install a library and use it to do the maths of the algorithm.





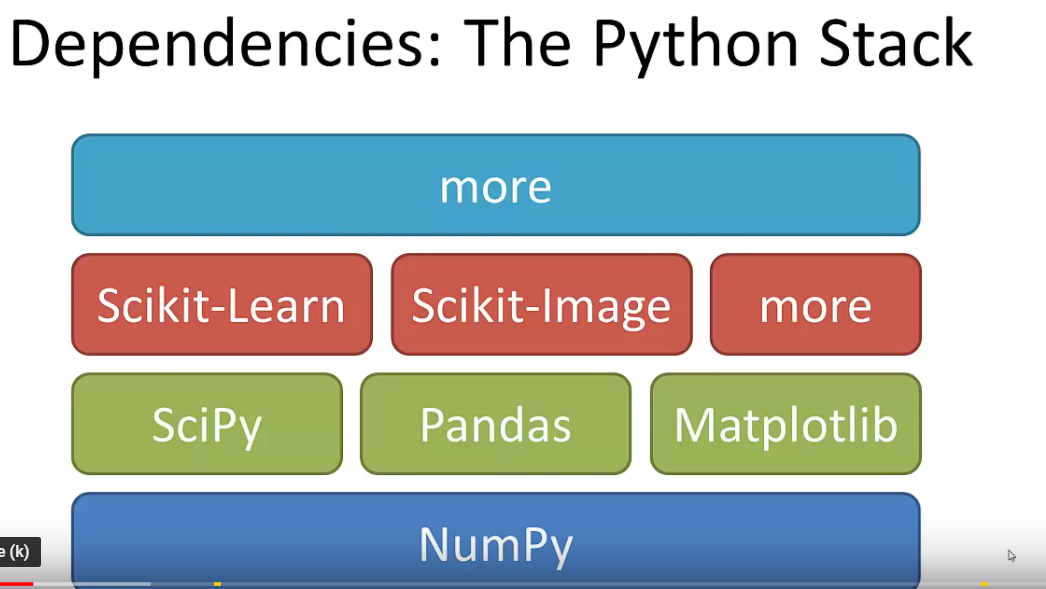


Do not build your own algorithm from scratch, in order to be efficient you need to reuse.

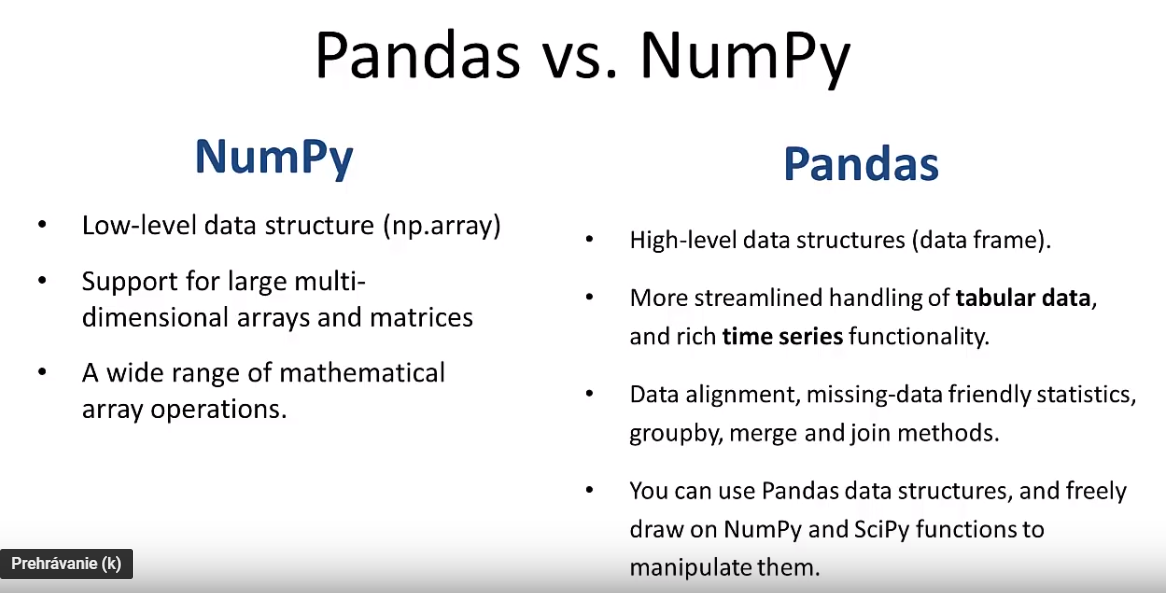
* 1. Scikit-Learn

While other frameworks (mxnet, pytorch, tensorflow) focus on Deep Learning – it is, where the magic happens – scikit learn is the easiest, supports almost all of the algorithms.

A free software machine learning library for the Python programming language.



* + 1. Pandas



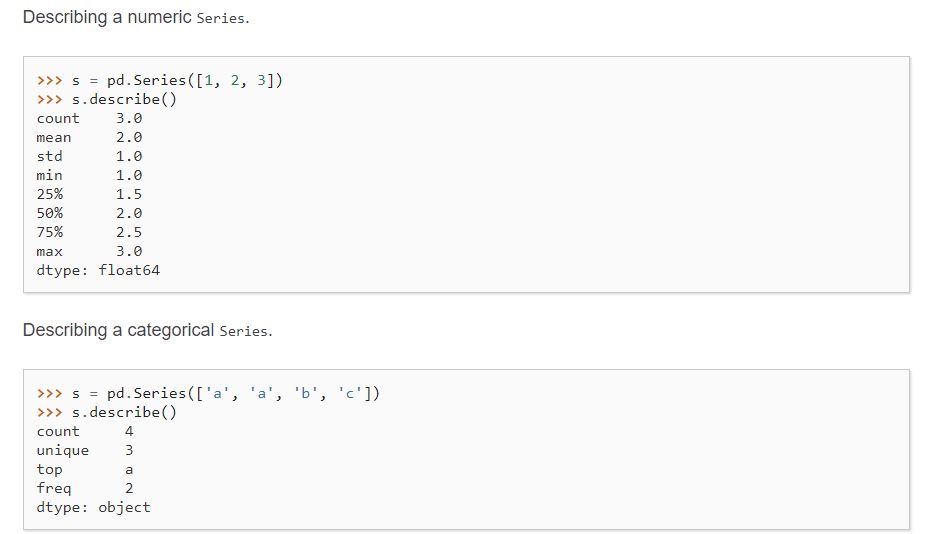
Pandas is an open source Python library providing high-performance, easy-to-use data structures and data analysis tools, runs on top of NumPy.

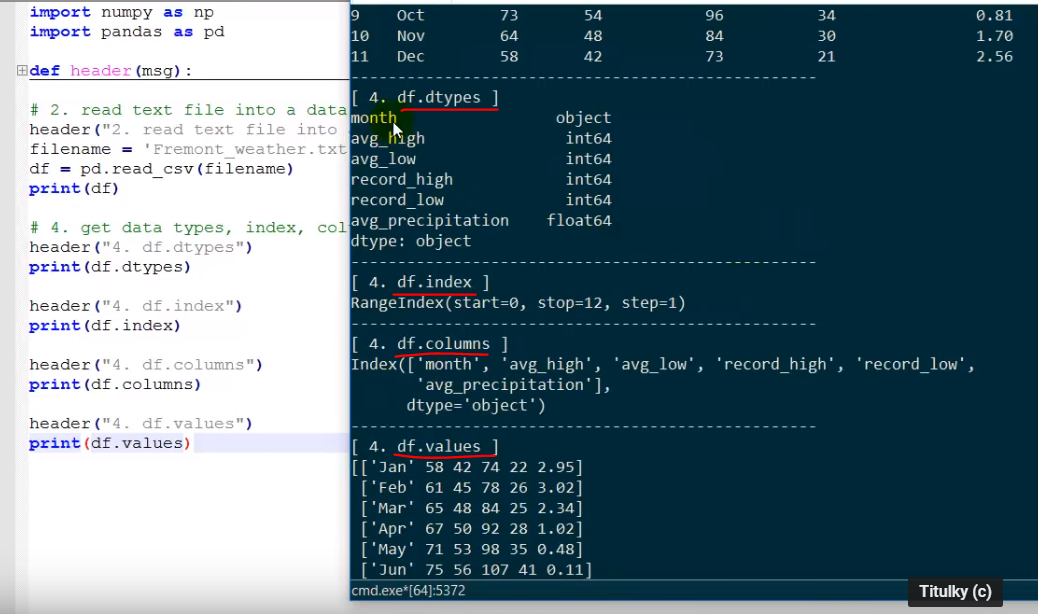
The data structure in Pandas is called DataFrame (DF)

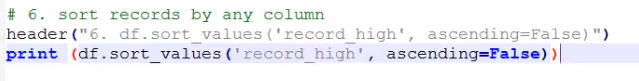
* df= pd.DataFrame()
* df = pd.read\_csv(filename)

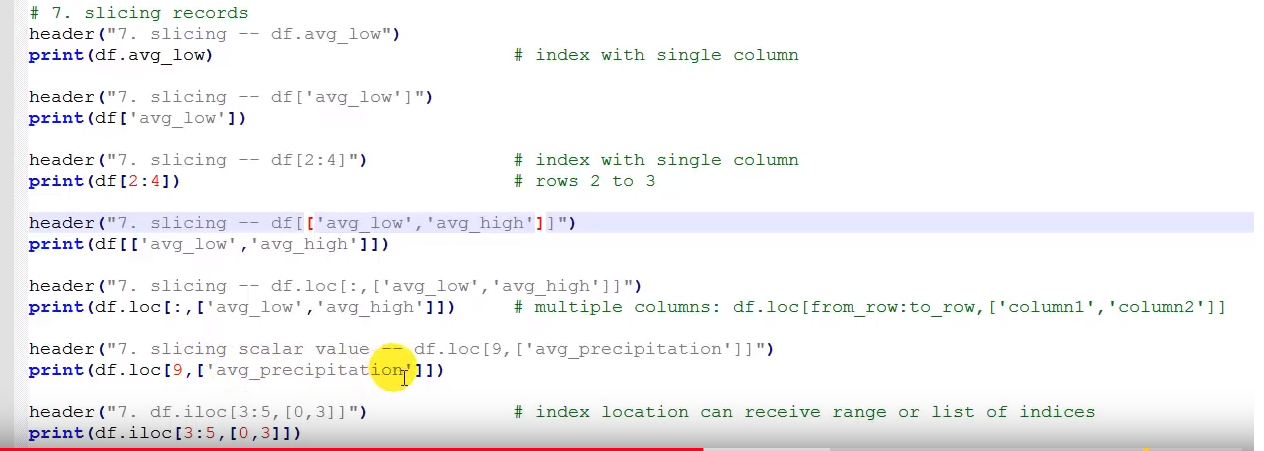
With 1 line of code we read in an entire csv file into a datafarame

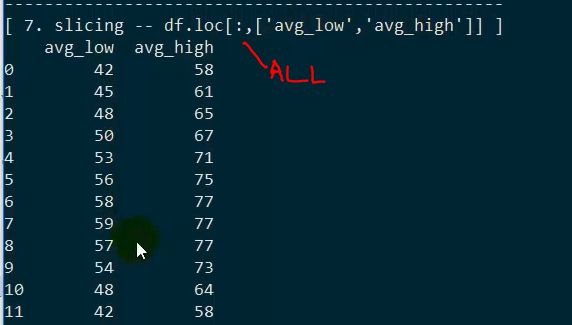
* df.describe()

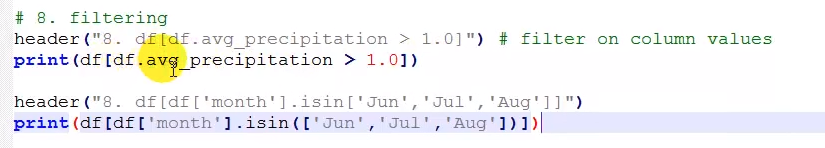




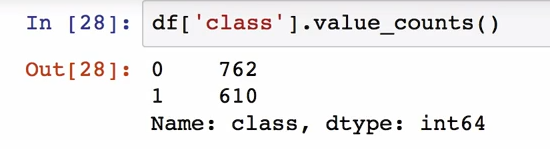








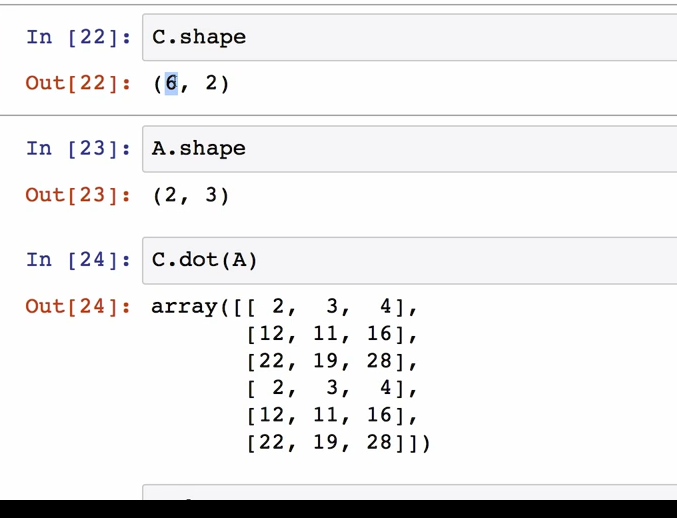
* df.head()
* df.tail(3)



* + 1. NumPy

*Like python lists but with superpower*

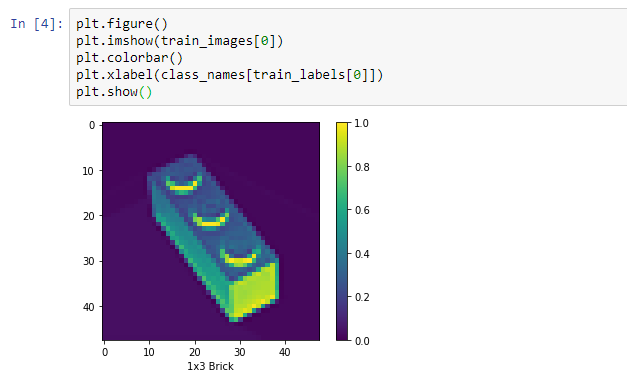
* Core lib for scientific computing in Python
* It provides a high-performance multidimensional array object, and tools for working with these arrays

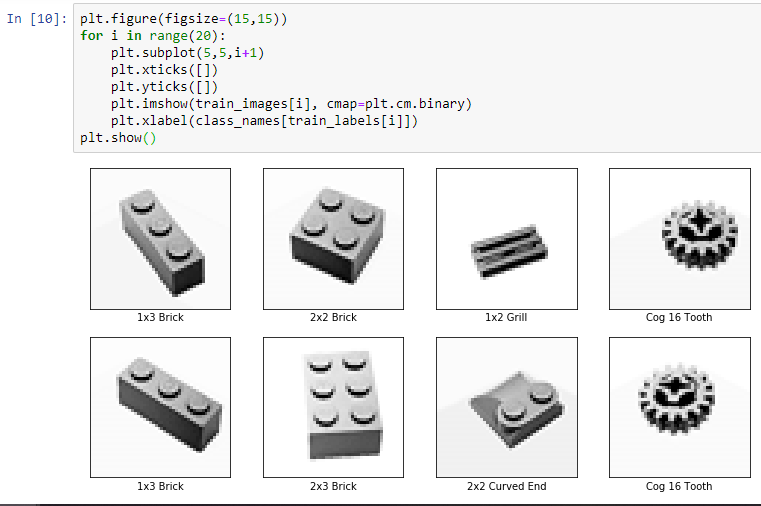


Multiplying matrixes – the last dimension of the first and the first dimension of the second matrixes must be equal.

* + 1. Matplotlib

Plot.imshow()





* 1. TensorFlow

TensorFlow uses a **dataflow graph** to represent your computation in terms of the dependencies between individual operations. This leads to a low-level programming model in which you first define the dataflow graph, then create a TensorFlow **session** to run parts of the graph across a set of local and remote devices. The session starts and will be automatically cleaned up. Sessions are good because it cleans after himself.

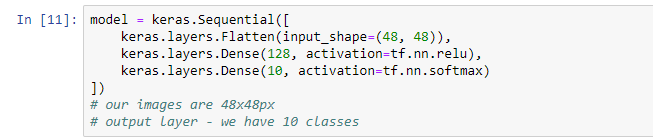
Build up a graph by defining constants, placeholders and operations, then run it.

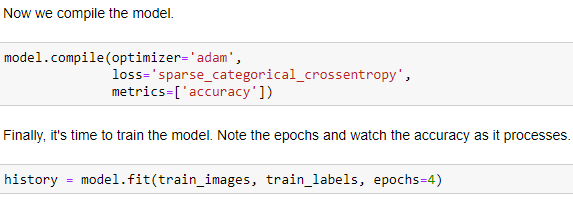
With tf.Session as sess:

Result = sess.run ()

* 1. Keras

It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to ding good research. Running on top of TensorFlow, CNTK or Theano.





3 lines (input, hidden, output) suggests 3 layers of the NN.

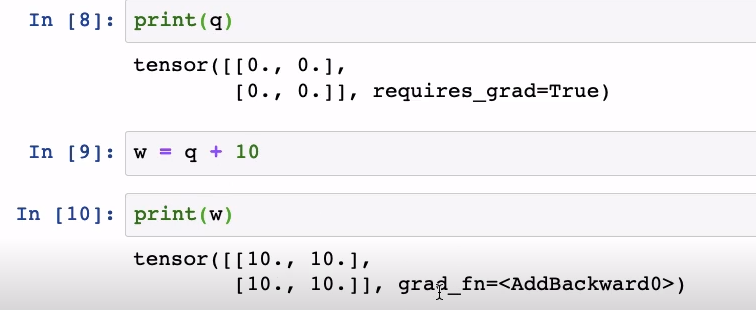
* 1. PyTorch

Natively Python. Better to doing come types of NN – e.g. recurrent

Import torch

Create a tensor (torch array) and you can perform operations on that – still nothing special with NumPy would not be able to do.

Magic happens here:



With TensorFlow you define the graph upfront and then you run it, w PyTorch you are creating the graph as you go along – more dynamic.

* 1. MXNet

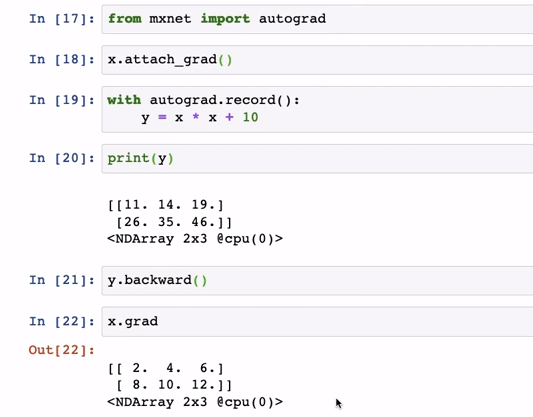
Import mxnet as mx

From mxnet import nd

By default uses CPU but w parameter you can set it to use GPU.



We want mxnet to record the operations, whats happening to array – so when comes to backpropagation it will be good (?).



Similar to pytorch, the way it handles the creation of the graph.

1. One-class classification (OOC)

Several approaches are have been proposed to solve one-class classification (OCC). The approaches can be distinguished into three main categories:

1. density estimation,
2. boundary methods,
3. and reconstruction methods.
   1. PU learning