# Practical work 01 – 25th February 2021
# Let's get started

---

The main objective of this Practical Work (PW) is to set up your computer environment and to bring you on track for this class on **Machine Learning and Deep Learning**.

We are going to use Python as programming language to perform some of the PW. This weeks' PW will include setting up Python on your computer and helping you to get familiar with this language.

For Python, many good libraries are available to work with data, to visualise data and for performing machine learning tasks. That's why we have chosen to use Python in this class.

For this class, we will work with iPython Notebooks which is also a good format to hand in your practical works. You can create, edit and run these notebooks from a web browser and write Python code or text (using Markdown) in so-called cells and also create nice output (e.g. tables, plots).

**Warning** : There are two "concurrent" versions of Python, versions 2 and 3 with some minor incompatible differences in language syntax. In this class, **we will use Python 3**.

### Moodle Subscription

As mentioned in the lecture, you will hand in the solutions to assignments by using Moodle.

Register on moodle at `https://moodle.msengineering.ch`. To find the class navigate to Home → Zürich → Technical scientific specialization modules → SPR21 TSM - Spring Semester 21 → TSM_DeLearn. Use subscription key : `moodlemsekey`.

### Hand-In of Practical Work

Most of the time, you'll need to submit your PW reports **by Wednesday, 10am of the following week**. However, we may override this rule. In any cases, the dates indicated in Moodle are the ones you should follow.

For this first PW, we expect you to **submit a solution for exercise 3** in form of a **iPython Notebook and a small report as pdf** (follow the instructions in the iPython Notebook).

## Exercise 1 Python Installation on Your Computer

Skip this part if you are already set up with Python, Jupyter notebooks and your favorite IDE for Python. Otherwise, we recommend the following installations :

— Jupyter notebooks with manual installation in virtual environments (**recommended**) - First install Python 3.8 from https://www.python.org. Then create a working directory and open a terminal in the directory :

a) Install the virtual env : `python3.8 -m venv venv`

b) Activate the virtual env on Mac : `source venv/bin/activate`

c) Activate the virtual env on Windows : `C:\> <venv>\Scripts\activate.bat`

d) Install Jupyter : `pip install jupyter`

e) Install needed libs : `pip install numpy matplotlib pandas scikit-learn`

f) Install tensorflow : `pip install tensorflow`

g) Launch Jupyter : `venv/bin/jupyter notebook`

Whenever you need to re-launch the notebooks, repeat steps b) and g). Whenever needed install requested libraries with `pip install LIB_NAME` once the virtual env is setup step b).

— Anaconda platform - a distribution of Python with popular data science packages that are pre-installed or easily installable. https://www.continuum.io. Select the 3.8 version of Python from https://www.anaconda.com/products/individual, as illustrated on Figure 1 below.

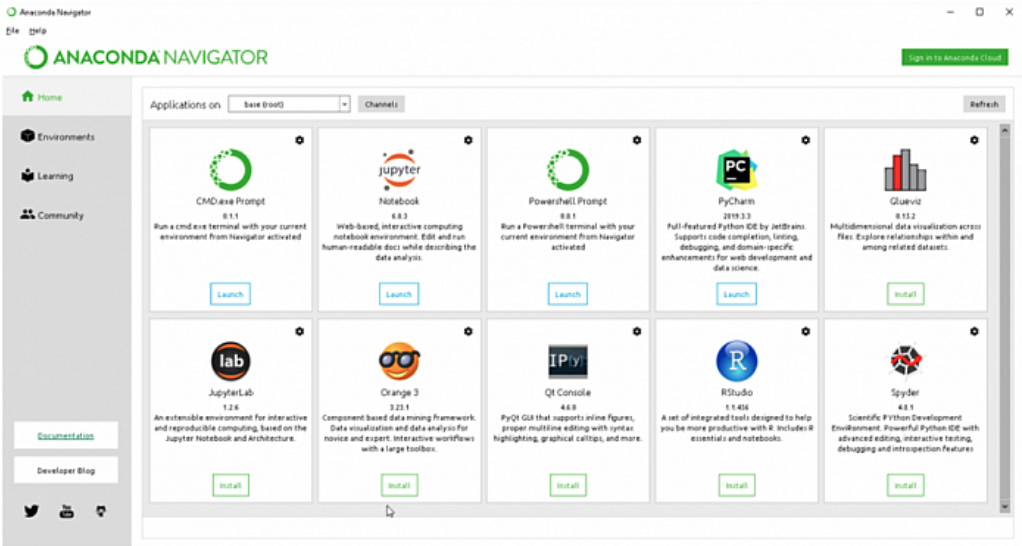


FIGURE 1 – Installing the Anaconda platform

— PyCharm IDE - an integrated development environment for Python. https://www.jetbrains.com/pycharm. You may select Professional or Community edition - the Community edition should be enough for this class. Free licenses are given to students, use your `hes-so.ch` address for the registration by JetBrains.

— If you do not want to install anything on your computer, you may use Google Colab at https://colab.research.google.com/.

## Exercise 2 Python Language in a Nutshell

We assume here that students are knowledgeable in other programming languages such as Java or C and that basic data structure concepts are known. If you know already Python and the concept of notebooks, then you can skip this exercise.

— Open Jupyter from the Anaconda Navigator. Jupyter Python notebooks run in your browser so, after launching Jupyter from Anaconda, a browser should show up. Open a new notebook from menu File and follow the User Interface Tour as illustrated in Figure 2.

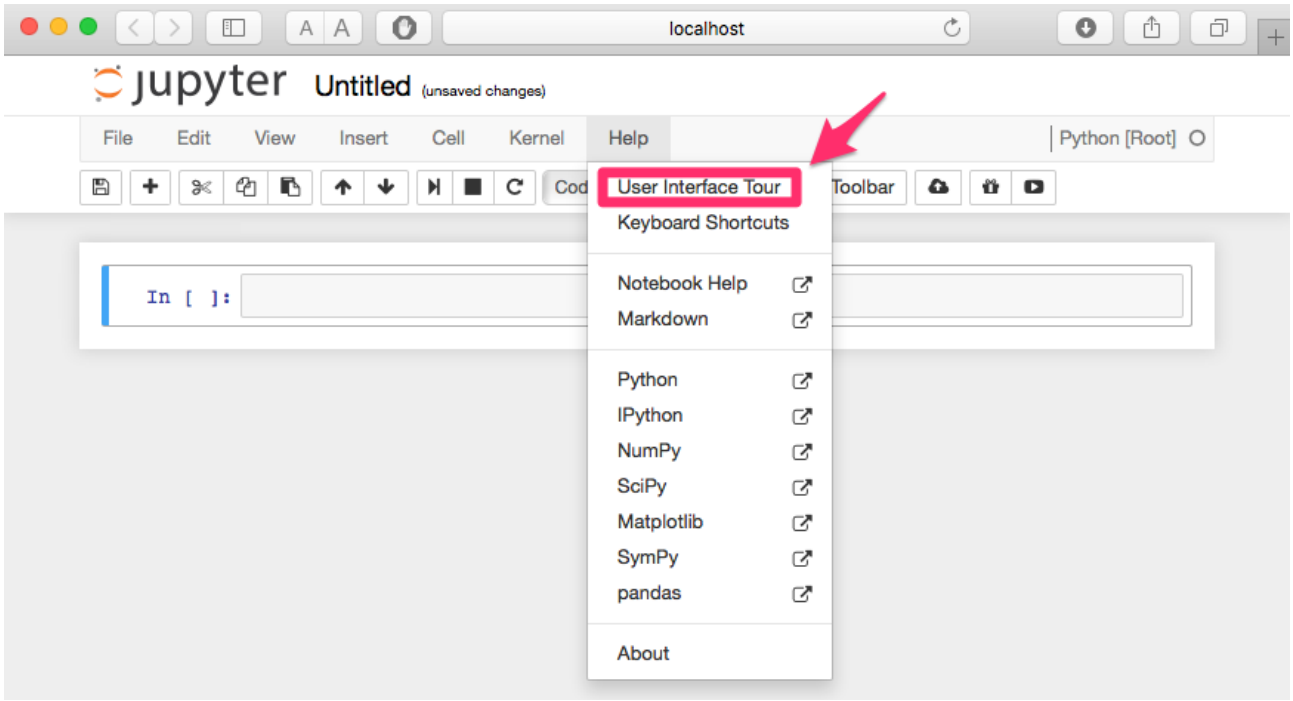


FIGURE 2 – First steps with Python notebooks.

— Download the file `intro-python-3.ipynb` from moodle and open it from Jupyter in Anaconda. You need to navigate where you stored the ipynb file. See Figure 3 below.
— Go through the content of the `intro-python-3.ipynb` notebook and play with the cells. This document should give you a quick introduction to Python assuming that you are fluent with other programming languages. You may also want to get familiar with Markdown syntax if not known already.
— If you want a more fully fledged introduction to Python, read the official tutorial from https://docs.python.org/3.8/tutorial/.

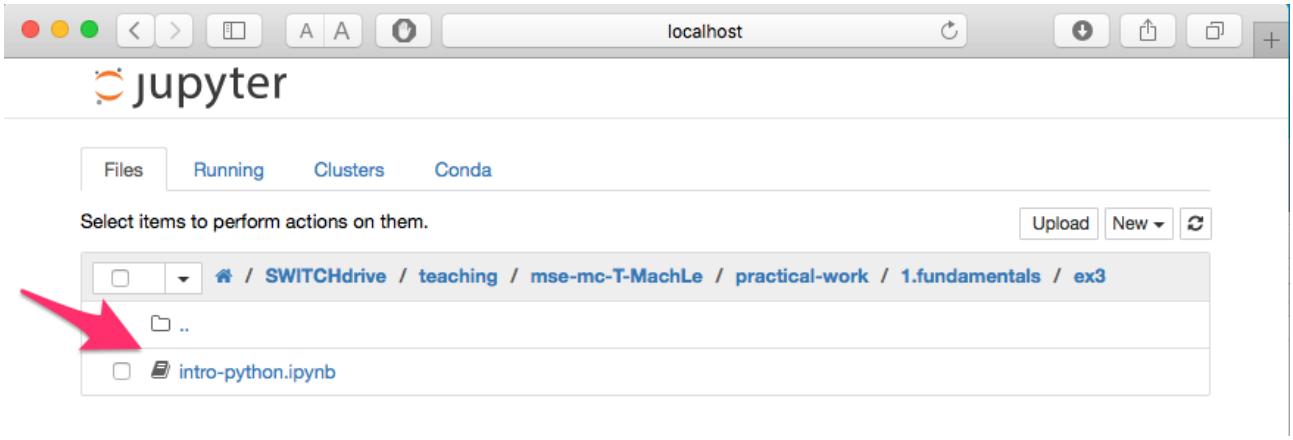


FIGURE 3 – Intro to Python language and Python notebooks.

## Exercise 3 Perceptron Learning Algorithm

The goal of this exercise is to implement the perceptron learning algorithm and to test it by applying it to data :

a) Step 1 : In a first step, you use a dummy dataset which is generated on the fly.

b) Step 2 : In a second step, you use the lightweight MNIST dataset which consists of $8 \times 8$-pixel images of handwritten digits and will explore for what digits pairs the Perceptron Learning algorithm will converge.

You will program the learning algorithm so that it finds a decision boundary - in Step 1 as indicated by the green dashed line in the Figure below that separates between the red and the blue crosses. The algorithm will perform updates on an initial (typically wrong) guess - in Step 1 as indicated by the magenta dashed line. Note that for the example in Step 2 with the $8 \times 8$-pixel images (i.e. 64-dimensional arrays) a visualization of the decision boundary (if it exists) is no longer easily possible.

— Open the notebook named `perceptron_learning_stud.ipynb` and inspect the functions defined therein. For better readability, you can use $help(function\_name)$ to inspect the documentation of what the functions should do. Carefully inspect the shapes of the numpy arrays used - you will benefit from it in later weeks!

— For some of the functions you should complete missing blocks of code that are marked as follows.

```
### START YOUR CODE ###

### END YOUR CODE ###
```

— If you manage to complete all the missing code properly it should find in Step 1 a correct decision boundary in a finite number of steps. Summarise your findings in an additional cell at the end of Step 1. In Step 2, the outcome may depend on the specific digits selected to be classified. That is something to be explored as part of the exercise.

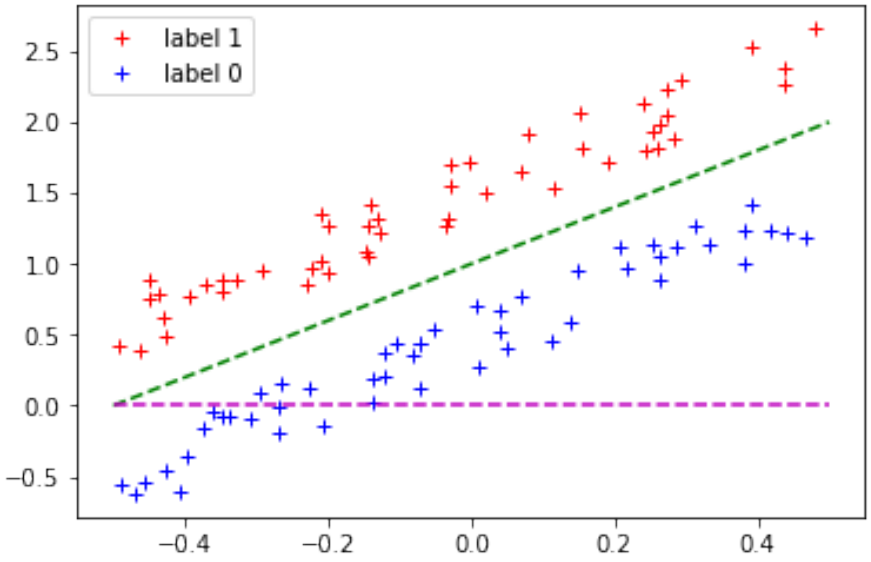


FIGURE 4 – The linearly separable example data used to train a classifier with the perceptron learning rule. The ideal decision boundary would be the green dashed line (the max margin classifier) - the magenta dashed line will be the first trivial guess which is bad since the number of mis-classifications is high.

Summarise your findings in an additional cell at the end of Step 2. For further details see `perceptron_learning_stud.ipynb`.

— (Optional) Finally, try to generate a dataset (by modifying the $prepare_data$-function in Step 1), so that the Perceptron learning algorithm no longer converges.

## Exercise 4 Optional : Data Visualization

To train a bit yourself, we propose you to do a visualisation workout with the Iris dataset https://en.wikipedia.org/wiki/Iris_flower_data_set. Download the file `iris.txt` from moodle or from http://www.statlab.uni-heidelberg.de/data/iris/. Put it in a Python data structure and attempt to reproduce a plot close to the one in Figure 5. Advice : start by working with individual plots and then move to a grid of plots with `subplot` **or** take the shortcut of using the library `pandas`.

The iris species classification task is a classical one. The goal is to distinguish three kinds of iris : setosa, versicolor and virginica. In the data set, a botanist has extracted 4 *features* : sepal length, sepal width, petal length and petal width.

What can you say regarding class separation ? One class seems easier to distinguish from the others, which one ? How would you separate the other two ?

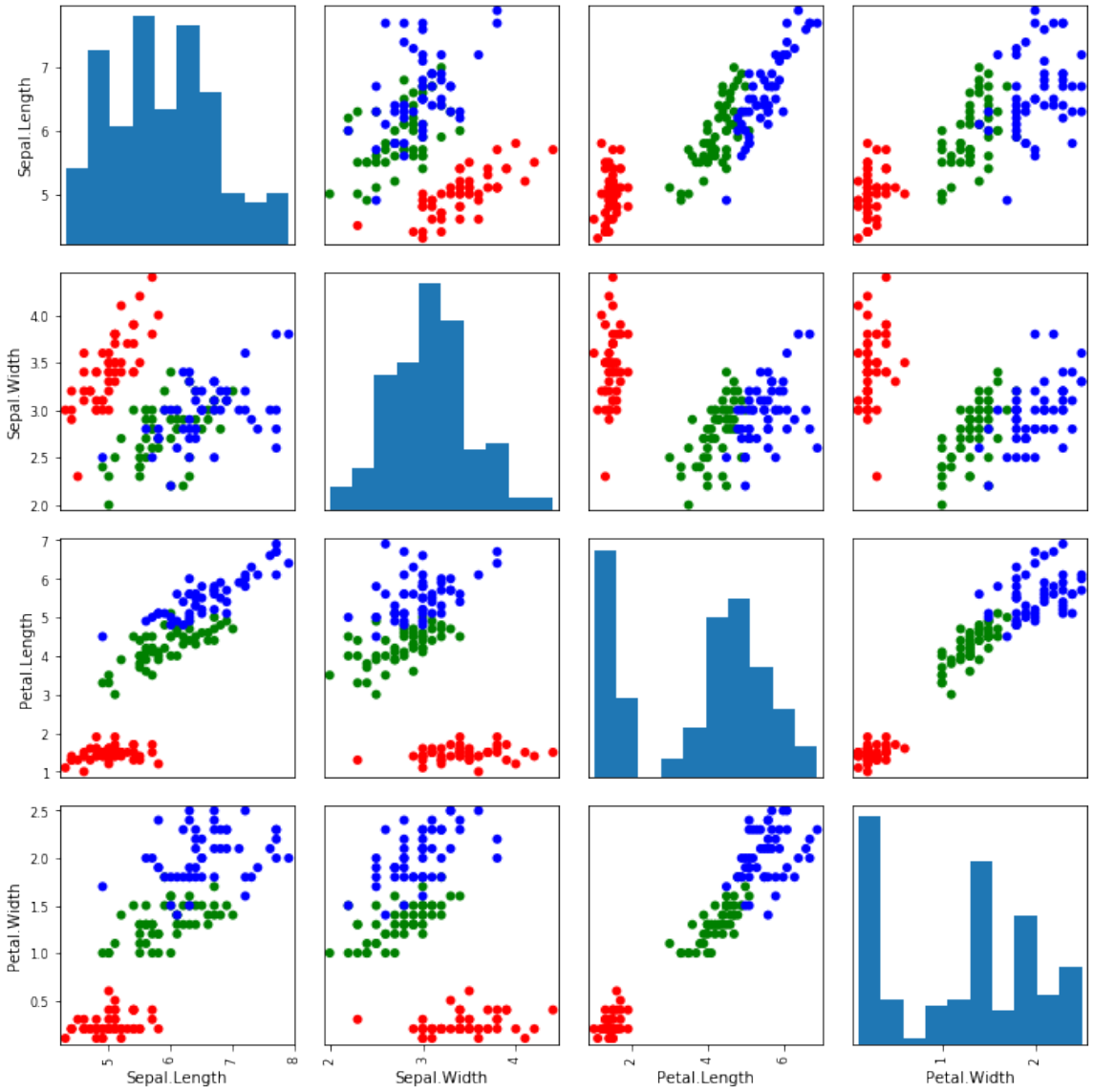


FIGURE 5 – Visualization of the Iris data as a pairwise scatter plot. The diagonal plots the marginal histograms of the 4 features. The other cells contain scatterplots of all possible pairs of features. Red circle = setosa, green diamond = versicolor, blue star = virginica.

## Exercise 5 Optional : Review Questions

### 1) Supervised vs. unsupervised systems

Of the following examples, which one would you address using a supervised or an unsupervised learning algorithm ? Give some explanations for your answers.

a) Given email labeled as spam/not spam, learn a **spam filter**.

b) Given a set of news articles found on the web, group them into sets of **related articles**.

c) Given a database of customer data, automatically discover **market segments** and group customers into different market segments.

d) Given a dataset of patients diagnosed as either having **glaucoma** or not, learn to classify new patients as having glaucoma or not.

### 2) Classification vs. regression systems

Can we transform a regression problem into a classification problem ? What would be the benefits of doing so ?

### 3) Perceptron

a) For what kind of problems are Perceptrons suited ?

b) For what kind of problems will the Perceptron Learning Algorithm converge ?

c) What kind of models can be trained by the Perceptron Learning Algorithm ?

d) Give an example for which the Perceptron Learning Algorithm will not converge. Explain why not.

## Exercise 6 Optional - Reading Assignments

— Read the first chapter of Murphy's book "Machine Learning".

Build your own summary of these chapters by doing a taxonomy of machine learning problems. You can find the pdfs on Moodle.