

# Machine Learning and Artificial Intelligence

Lab 01 – Introduction

09/03/2021

# An initial FAQ

- What will we learn in this lab? Implementations of various algorithms seen during the lectures in *Python*.
- What can we do if we have doubts regarding the arguments treated during the labs? You can contact me anytime via mail at [geri.skenderi@univr.it](mailto:geri.skenderi@univr.it)

# Brief introduction to ML

- **Machine learning:** The study of computer algorithms that improve automatically through experience.
- Starting from a set of data, we can try to understand underlying tendencies in order to then make predictions or decisions without being explicitly programmed to do so.
- Simple example: Given a set of images, we need to try and understand if a new never seen before image contains a cat or not.

# Brief introduction to ML

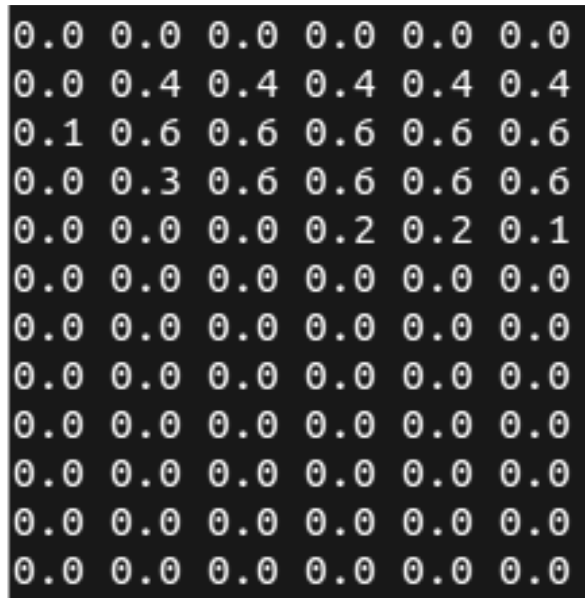
- **The above tasks are very intuitive and easy for humans:** because of mechanisms we don't fully understand yet.
- On a computing level, these tasks are not simple.

# Why is that?

What we see

3

What the computer sees



Variability



# Some more examples



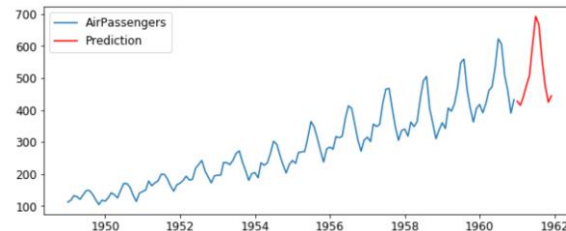
Understand that this is an apple



Distinguish different characters written by hand



Identify the most important object in a scene

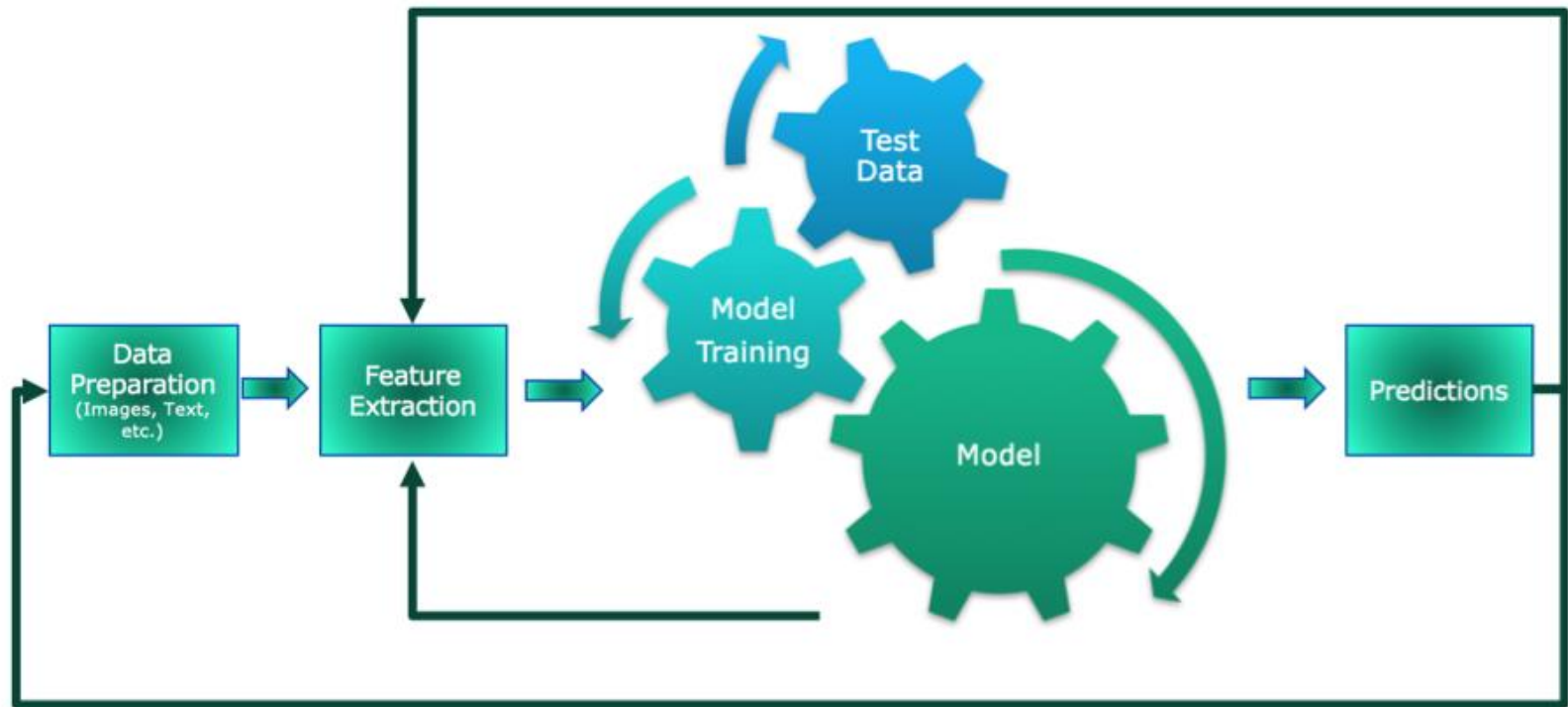


Predict the future values of a signal based on the past

# Other classic examples:

- Scene recognition:
  - <http://places2.csail.mit.edu/demo.html>
- Image understanding:
  - <https://azure.microsoft.com/it-it/services/cognitive-services/computer-vision/>
- Image generation from text queries:
  - <https://openai.com/blog/dall-e/>

# A Standard Machine Learning Pipeline



*Learning. From. Experience.*



# In this course: ML in Python



- Python is a high-level, object-oriented (often used as a scripting language), multi-purpose, cross-platform programming language.
- In recent years, it has been widely used for scientific computing.

# Python

## Weird stuff:

- Dynamic typing
- Indentation is *very* important
- Multiple differences between version 2.7 and 3+

## IDEs:

- PyCharm
- VS Code
- Bash command *python* opens an interactive shell

<https://docs.python.org/3/tutorial/>

# Python package manager

Python has an internal system for downloading, storing, and resolving packages. The most common packages (open source, third-party) are available in a repository called the Python Package Index (PyPI).

```
pip install numpy  
pip install scipy==1.2.1  
pip install --user Pillow # install only for current user
```

A requirements.txt file is handy for saving all the packages on which the project depends:

```
pip install -r requirements.txt
```

# Importing packages

Python is so popular because of its packages and libraries. Here's an import of *numpy* in the shell:

```
>> import numpy as np
```

N.B.: pip installs packages system-wide (globally). It may not always be the best solution (compatibility issues between versions, need to be superuser etc.)

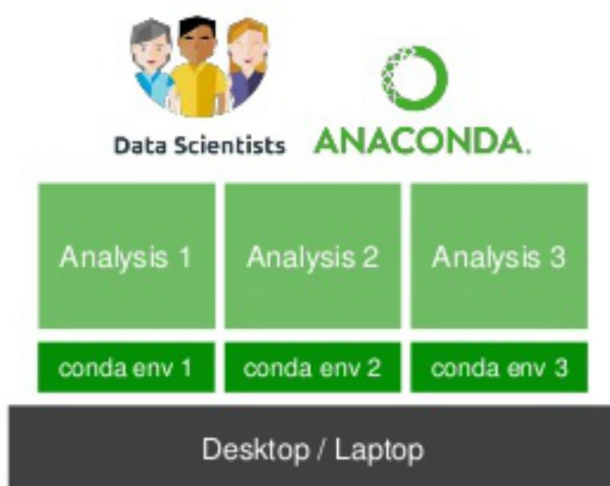
# Conda



## Create an isolated virtual environment

To install → <https://www.anaconda.com/products/individual#Downloads>

Legacy → <https://repo.anaconda.com/archive/>



Creating a virtual env

```
conda create -n <name> python=3.8
```

Activating and deactivating the env

```
conda activate <name>
```

```
conda deactivate
```

# Python: Lists, vectors and matrices

```
>> a = 5                # scalar
>> b = [2, 3]           # list, len(b): 2
>> c = [[1,2],[3,4]]    # list of lists. len(c): 2
```

```
>> import numpy as np
>> d = np.array([0])     # scalar. len(d): 1, d.shape: (1,)
>> e = np.array([2, 3])  # vector (array). len(e): 2, e.shape: (2,)
>> F = np.array([[1,2],[3,4]]) # matrix. len(F): 2, F.shape: (2,2)
```

# Python: matrix operations

Are of great importance in Machine Learning.

```
>> B = A + A
>> C = A ** 2 # power of 2
>> C = A * A # element-wise
>> C = np.dot(A,A)
>> D = np.divide(A,A) # element-wise
>> D = np.dot(A, np.linalg.inv(A))
>> np.sum(A)
>> np.sum(A, axis=0) # col wise
>> np.mean(A, axis=1) # row wise
```

Matrix product:  $c_{i,j} = \sum_k A_{i,k} B_{k,j}$

Matrix division:  $D = AA^{-1}$

# Extracting data from matrices

```
>> a = [10,20,30,40]
>> b = a[0:2] # b = [10,20]
>> A = np.array([[10,20,30],[40,50,60]])
>> B = A[:2,:2]
```



# Python as a scripting language

- A script is a set of python commands that are interpreted
- Scripts are simple text files with a .py extension
- To execute a script: `python <name>.py`
- Debugging: depends on the IDE
- A lot of useful information can be found in the python cheatsheet attached to the lesson

# Visualization

The most popular libraries for visualization are:

1. Matplotlib: for the visualization and plotting of

<https://matplotlib.org/contents.html>

2. Pillow: for the visualization of images

<https://pillow.readthedocs.io/en/latest/>

# Jupyter Notebook

- Unique set-up for programming: open-source web-based application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

<https://jupyter.org/>

- Very useful for sharing and interactive output functionalities.
- Files end with a .ipynb extension.

# Google Colab

- Free platform for developing Python code on cloud based on Jupyter Notebooks.
- PROs:
  - Most scientific Python libraries already installed
  - Access to Google GPUs and TPUs
  - Ability to share code

<https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c>

## Exercises

On the course's Moodle page