chapter-1-1

February 2, 2022

Run in Google Colab

1 Introduction to Machine Learning

with Applications in Banking and Finance

In this notebook, you will learn about the main concepts and different types of machine learning. Together with a basic introduction to the relevant terminology, we will lay the groundwork for successfully using machine learning techniques for practical problem solving.

In the following we will cover the following topics:

- The general concepts of machine learning
- The three types of learning and basic terminology
- The building blocks for successfully designing machine learning systems
- Installing and setting up Python for data analysis and machine learning

1.1 What is Machine Learning

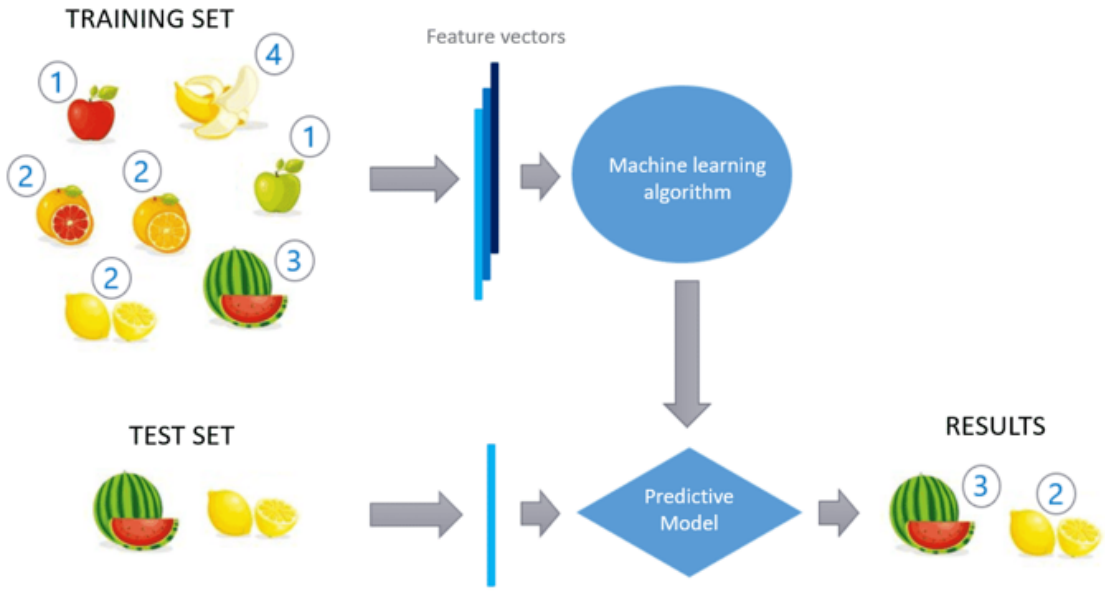
Machine learning is an artificial intelligence (AI) technology which provides systems with the ability to automatically learn from experience without the need for explicit programming, and can help solve complex problems. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

1.2 The three different types of machine learning

1.2.1 Supervised Learning

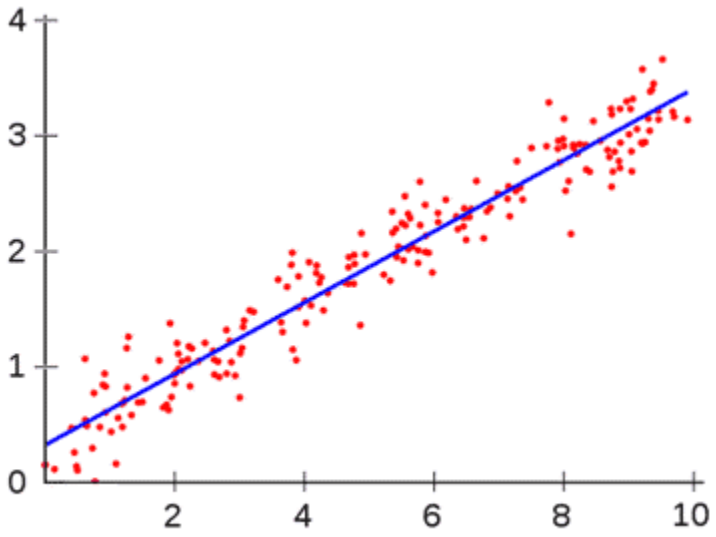
The main goal in supervised learning is to learn a model from labeled training data that allows us to make predictions about unseen or future data. Here, the term "supervised" refers to a set of **training** examples (data inputs) where the desired output signals (**labels**) are already known.

The following figure summarizes a typical supervised learning workflow, where the labeled training data is passed to a machine learning algorithm for fitting a predictive model that can make predictions on new, unlabeled data inputs:



A supervised learning task with discrete class labels, such as in the previous example, is also called a **classification task**. Another subcategory of supervised learning is regression, where the outcome signal is a continuous value.

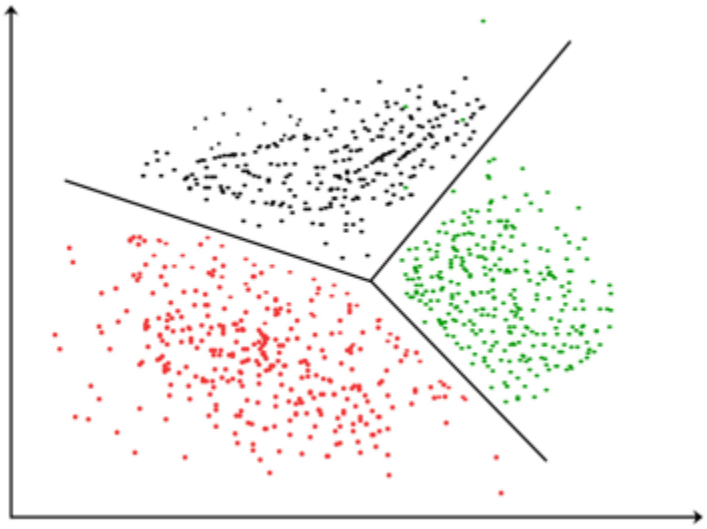
A second type of supervised learning is the prediction of continuous outcomes, which is also called **regression analysis**. In regression analysis, we are given a number of predictor (explanatory) variables and a continuous response variable (outcome), and we try to find a relationship between those variables that allows us to predict an outcome. Note that in the field of machine learning, the predictor variables are commonly called ***features***, and the response variables are usually referred to as ***target variables***.



1.2.2 Unsupervised Learning

In supervised learning, we know the right answer beforehand when we train a model. In **unsupervised learning**, however, we are dealing with ***unlabeled data*** or data of unknown structure. Using unsupervised learning techniques, we are able to explore the structure of our data to extract meaningful information without the guidance of a known outcome variable or reward function.

Clustering is an exploratory data analysis technique that allows us to organize a pile of information into meaningful subgroups (clusters) *without having any prior knowledge of their group memberships*. Each cluster that arises during the analysis defines a group of objects that share a certain degree of similarity but are more dissimilar to objects in other clusters, which is why clustering is also sometimes called unsupervised classification. Clustering is a great technique for structuring information and deriving meaningful relationships from data. For example, it allows marketers to discover customer groups based on their interests, in order to develop distinct marketing programs.



1.2.3 Reinforcement Learning

Another type of machine learning is **reinforcement learning**. In reinforcement learning, the goal is to develop a system (***agent***) that improves its performance based on interactions with the environment. Since the information about the current state of the environment typically also includes a so-called **reward signal**, we can think of reinforcement learning as a field related to supervised learning. However, in reinforcement learning, this feedback is not the correct ground truth label or value, but a measure of how well the action was measured by a reward function. Through its interaction with the environment, an agent can then use reinforcement learning to learn a series of actions that maximizes this reward via an exploratory trial-and-error approach or deliberative planning. A popular example of reinforcement learning is a chess engine. Here, the agent decides upon a series of moves depending on the state of the board (the environment), and the reward can be defined as win or lose at the end of the game.

1.3 Features and Labels

The data for supervised learning contains whare are referred to as **features** and **labels**. The **labels** are the values of the target that is to be predicted. The **features** are the variables from which the predictions are to be made. For example when predicting the price of a house the **features** could be the swuare meters of living space, the number of bedrooms, the number of bathrooms, the size of the garage and so on. The **label** would be the house price.

The data for unsupervised learning consists of features but no labels because the model is being used to identify patterns not to forecast something.

1.4 Type of Data

There are two types of data:

- Numerical

- Categorical

Numerical data consists of numbers. Categorical data is data which can fall into a number of different categories, for example data to predict a house price might categorize driveways as asphalt, concrete, grass, etc. Categorical data must be converted to numbers for the purposes of analysis.

The standard way of dealing with categorical features is to create a dummy variable for each category. The value of this variable is 1 if the feature is in the category and 0 otherwise. For example in the situation in which individuals are categorized as male or female, we could create two dummy variables. For man the first dummy variable would be 1 and the second would be 0. The opposite for women. This procedure is appropriate when there is no natural ordering between the feature values.

When there is a natural ordering, we can reflect this in the numbers assigned. For example if the size of an order is classified as small, medium or large, we can replace the feature by a numerical variable where *small = 1*, *medium = 2* and *large = 3*.

1.5 Cost Functions

1.5.1 Linear Cost Function

In Machine Learning a cost function or loss function is used to represent how far away a mathematical model is from the real data. One adjusts the mathematical model, usually by varying parameters within the model, so as to minimize the cost function.

Let's take for example the simple case of a linear fitting. We want to find a relationship of the form

$$y = \theta_0 + \theta_1 x \tag{1}$$

where the θs are the parameters that we want to find to give us the best fit to the data. We call this linear function $h_\theta(x)$ to emphasize the dependence on both the variable x and the two parameters θ_0 and θ_1.

We want to measure how far away the data, the $y^{(n)}$s, are from the function $h_\theta(x)$. A common way to do this is via the quadratic *cost function*

$$J(\grave{}) = \frac{1}{2N} \sum_{n=1}^{N} \left[h_\theta \left(x^{(n)} \right) - y^{(n)} \right]^2 \tag{2}$$

This is called *Ordinary Least Squares*.

In this case, the minimum is easily find analitically, differentiate (2) with respect to both θs and set the result to zero:

$$\begin{aligned} \frac{\partial J}{\partial \theta_0} &= \sum_{n=1}^{N} \left(\theta_0 + \theta_1 x^{(n)} - y^{(n)} \right) = 0 \\ \frac{\partial J}{\partial \theta_1} &= \sum_{n=1}^{N} x^{(n)} \left(\theta_0 + \theta_1 x^{(n)} - y^{(n)} \right) = 0 \end{aligned} \tag{3}$$

The solution is trivially obtained for both θs

$$\begin{aligned}\theta_0 &= \frac{(\sum y)\left(\sum x^2\right)-(\sum x)(\sum xy)}{N(\sum x^2)(\sum x)^2} \\ \theta_1 &= \frac{N(\sum xy)-(\sum y)(\sum x)}{N(\sum x^2)(\sum x)^2}\end{aligned} \tag{4}$$

1.5.2 Gradient Descent

> We can describe the main idea behind gradient descent as climbing down a hill until a local or global cost minimum is reached. In each iteration, we take a step in the opposite direction of the gradient, where the step size is determined by the value of the **learning rate**, as well as the slope of the gradient.

The scheme works as follow: start with an initial guess for each parameter θ_k. Then move θ_k in the direction of the slope:

$$\theta_k^{new} = \theta_k^{old} + \beta \frac{\partial J}{\partial \theta_k} \tag{5}$$

Update all θ_k simultaneously and repet until convergence. Here β is a *learning factor* that governs how far you move. if β is too small it will take a long time to converge, if too large it will overshoot and might not converge at all.

The loss function J is a function of all of the data points. In the above description of gradient descent we have used all of the data points simultaneously. This is called *batch gradient* descent. But rather than use all of the data in the parameter updating we can use a technique called *stochastic gradient descent*. This is like batch gradient descent except that you only update using *one* of the data points each time. And that data point is chosen randomly.

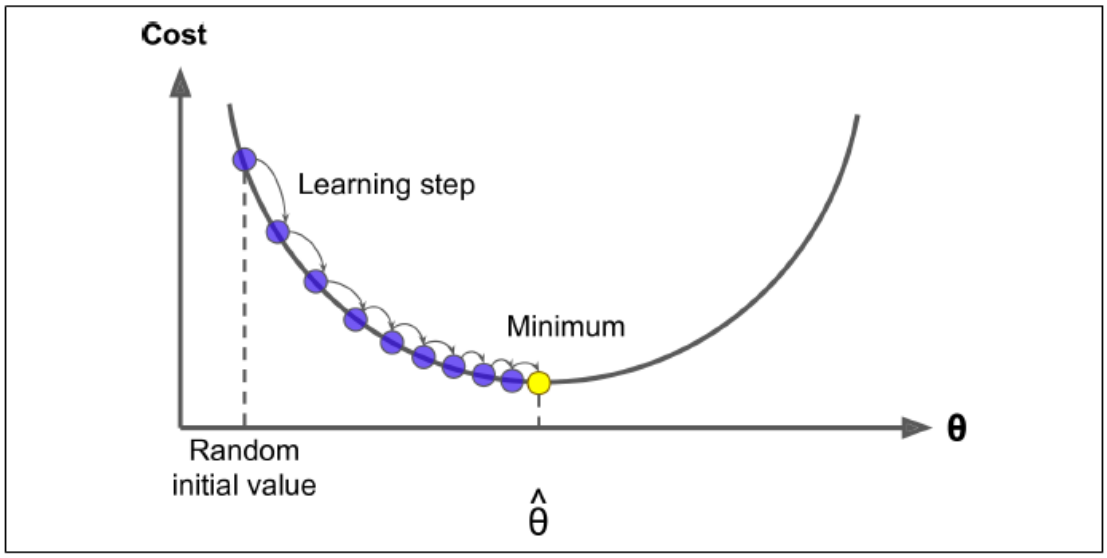
$$J(\grave{}) = \sum_{n=1}^{N} J_n(\grave{}) \tag{6}$$

Stochastic gradient descent means pick an n at random and then update according to

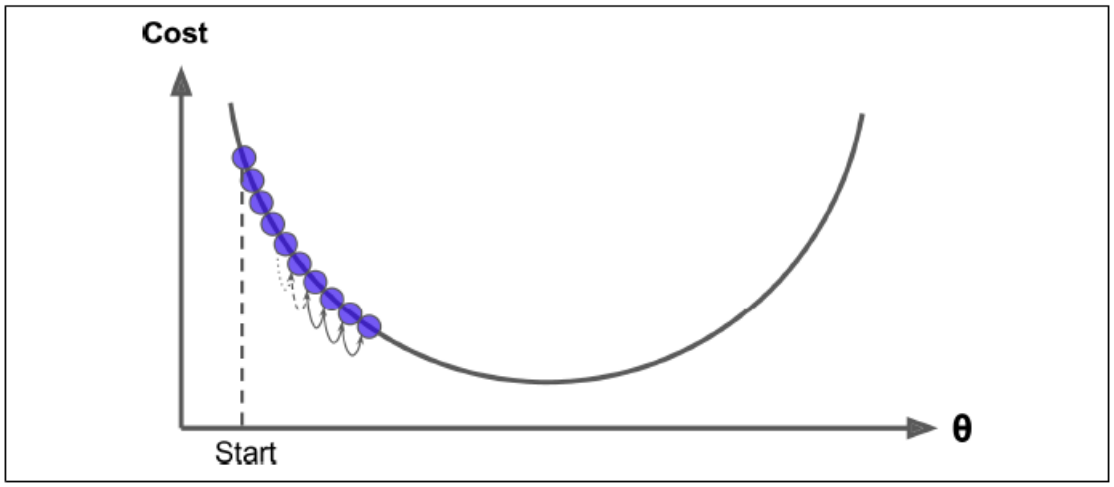
$$\theta_k^{new} = \theta_k^{old} + \beta \frac{\partial J_n}{\partial \theta_k} \tag{7}$$

Repeat, picking another data point at random, etc.

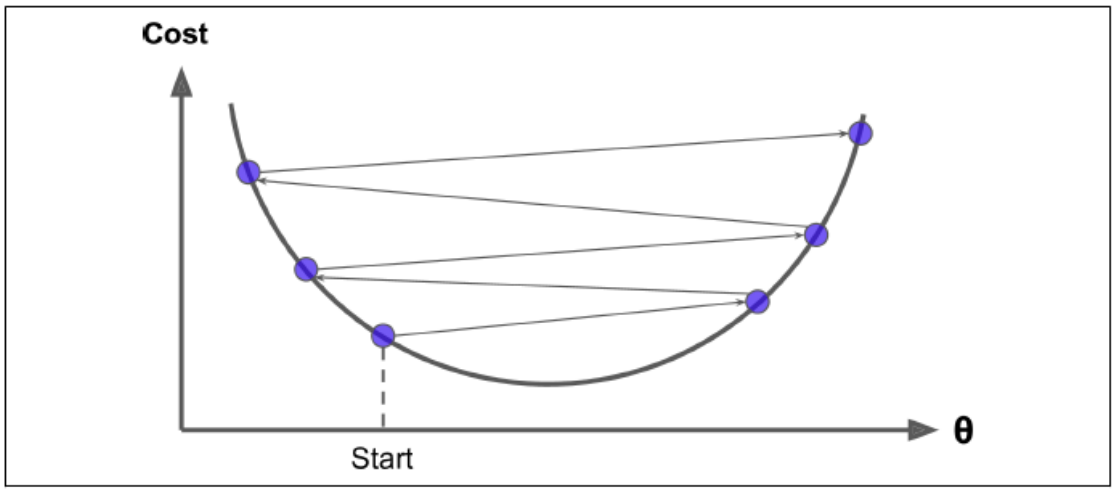
An important parameter in Gradient Descent is the size of the steps, determined by the **learning rate** hyperparameter.



If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time…



… on the other hand, if the learning rate is too high, you might jump across the valley. This might make the algorithm diverge failing to find a good solution.



Gradient descent is one of the many algorithms that benefit from feature scaling.

1.5.3 Stochastic Gradient Descent

In the previous section, we learned how to minimize a cost function by taking a step in the opposite direction of a cost gradient that is calculated from the whole training dataset; this is why this approach is sometimes also referred to as batch gradient descent. Now imagine that we have a very large dataset with millions of data points, which is not uncommon in many machine learning applications. Running batch gradient descent can be computationally quite costly in such scenarios, since we need to reevaluate the whole training dataset each time that we take one step toward the global minimum.

A popular alternative to the batch gradient descent algorithm is stochastic gradient descent (SGD), which is sometimes also called iterative or online gradient descent. Instead of updating the weights based on the sum of the accumulated errors over all training examples, we update the weights incrementally for each training example:

$$\eta \left(y^{(i)} - \phi \left(z^{(i)} \right) \right) \mathbf{x}^{(i)}$$

Although SGD can be considered as an approximation of gradient descent, it typically reaches convergence much faster because of the more frequent weight updates. Since each gradient is calculated based on a single training example, the error surface is noisier than in gradient descent, which can also have the advantage that SGD can escape shallow local minima more readily if we are working with nonlinear cost functions.

[ ]:

1.6 Learning Tools

1.6.1 Using Python for machine learning

Python is one of the most popular programming languages for data science and thanks to its very active developer and open source community, a large number of useful libraries for scientific computing and machine learning have been developed. Although the performance of interpreted languages, such as Python, for computation-intensive tasks is inferior to lower-level programming languages, extension libraries such as **NumPy**, **Matplotlib** and **Pandas**, among the others, have been developed that build upon lower-layer Fortran and C implementations for fast vectorized operations on multidimensional arrays. For machine learning programming tasks, we will mostly refer to the **scikit-learn** library, which is currently one of the most popular and accessible open source machine learning libraries. In the later chapters, when we focus on a subfield of machine learning called deep learning, we will use the latest version of the **Keras** library, which specializes in training so-called deep neural network models very efficiently.

1.6.2 Installing Python and Packages

To set up your python environment, you'll first need to have a python on your machine. There are various python distributions available and we have chosen one that works very well for data science: **Anaconda**. Anaconda comes with its own Python distribution which will be installed along with it.

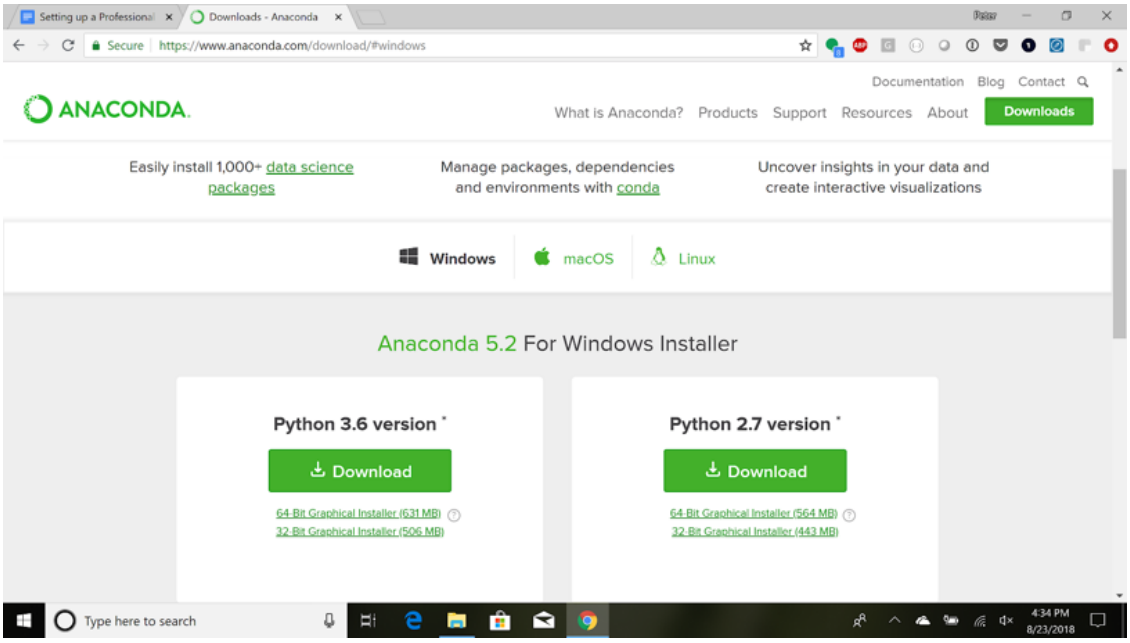
Data Science often requires you to work with a lot of scientific packages like scipy and numpy, data manipulation packages like pandas and IDEs and interactive Jupyter Notebook.Now, you don't need to worry about any python package most of them come pre-installed and if you want to install a new package, you can do that simply by using conda or via the pip installer program, which has been part of the Python Standard Library since Python 3.3. More information about pip can be found here. After we have successfully installed Python, we can execute pip from the terminal to install additional Python packages:

pip install SomePackage

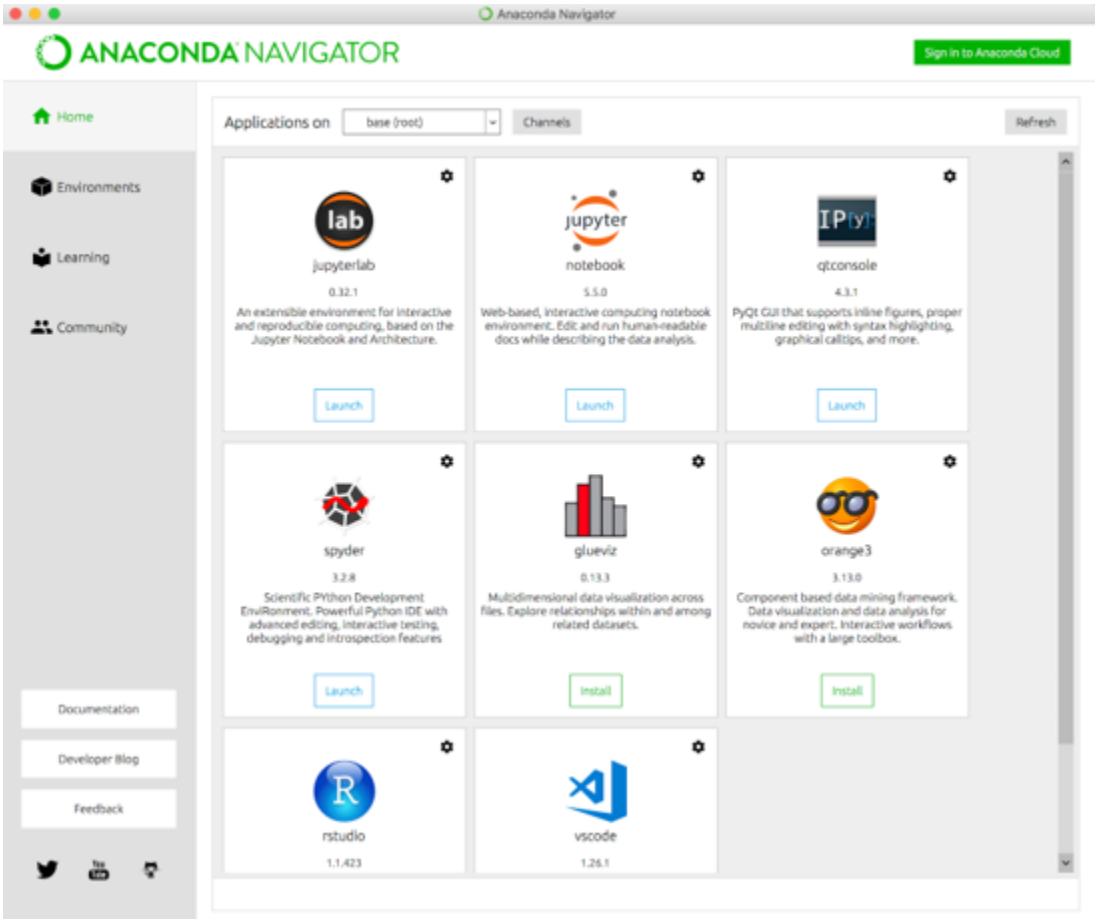
Already installed packages can be updated via the –upgrade flag:

pip install SomePackage –upgrade

To download an Anaconda distribution, you can use the official download page and you can select your platform and then choose the installer. For this, you can choose which version you want and whether 32-bit or 64-bit.



To test your installation, on Windows, click on Start and then Anaconda Navigator in the program list (or search for Anaconda in the search bar and select Anaconda Navigator). On a Mac, open up the finder, and in the Applications folder, double click on Anaconda-Navigator.



Package Managers

Anaconda will give you two package managers- **pip** and **conda**. When some packages aren't available with conda, you can use pip to install them. Note that using pip to install packages also available to conda may cause an installation error.

Jupyter Notebook

A notebook is a document like this one! A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media.

In other words: it's a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable. As part of the open source Project Jupyter, Jupyter Notebooks are completely free. You can download the software on its own, or as part of the Anaconda data science toolkit.

Data Files

INSERIRE UNA DESCRIZIONE PRECISA DEI FILE DI DATI CHE VERRANNO UTILIZZATI DOVE RECUPERARLI E DOVE TENERLI

1.6.3 Google Colab

Although it is not essential to work in a colab environment (all the course notebooks are in fact designed to be able to run without problems locally on your pc), it is useful to know some basic elements of the interaction with colab. In particular, in the cells below you will find two examples for the use of external files. In the first case it is shown how to load a text file from your local PC into the google virtual machine. The second example relates to the opposite operation: let's create a simple pandas dataframe into the colab environment and export it in csv format to the local machine.

How Upload a File on Google Colab

[1]:
```
if 'google.colab' in str(get_ipython()):
    from google.colab import files
    uploaded = files.upload()
    path = ''
else:
    path = './data/'
```

[2]:
```
with open(path + "carroll-alice.txt", "r") as f:
    alice = f.read()

alice[:392]
```

[2]:
```
"[Alice's Adventures in Wonderland by Lewis Carroll 1865]\n\nCHAPTER I. Down the
Rabbit-Hole\n\nAlice was beginning to get very tired of sitting by her sister on
the\nbank, and of having nothing to do: once or twice she had peeped into
the\nbook her sister was reading, but it had no pictures or conversations
in\nit, 'and what is the use of a book,' thought Alice 'without pictures
or\nconversation?'"
```

How Download a File on Google Colab

[3]:
```
import pandas as pd

cars = {'Brand': ['Honda Civic','Toyota Corolla','Ford Focus','Audi A4'],
        'Price': [22000,25000,27000,35000]
        }

df = pd.DataFrame(cars, columns= ['Brand', 'Price'])
```

[4]:
```
if 'google.colab' in str(get_ipython()):
    # if we run in google environment first we save in virtual machine...
    df.to_csv ('export_dataframe.csv', index = False, header=True)
    # ...then we download to local machine
    from google.colab import files
    files.download("export_dataframe.csv")
```

```
else:
    # if we are working in local we save directly with the usual method
    df.to_csv ('./data/export_dataframe.csv', index = False, header=True)
```

1.6.4 Packages for scientific computing

Throughout this course, we will use **NumPy**'s multidimensional arrays to store and manipulate data. We will make use of **Pandas**, which is a library built on top of NumPy that provides additional higher-level data manipulation tools that make working with tabular data even more convenient. To augment your learning experience and visualize quantitative data, which is often extremely useful to make sense of it, we will use the very customizable **Matplotlib** library.

1.7 Summary

In this lesson, we explored machine learning at a very high level and familiarized ourselves with the big picture and major concepts that we are going to explore in the following chapters in more detail.

We learned that:

- **Supervised learning** is composed of two important subfields: **classification** and **regression**. While classification models allow us to categorize objects into known classes, we can use regression analysis to predict the continuous outcomes of target variables;
- **Unsupervised learning** offers useful techniques for discovering structures in unlabeled data;
- **How to set up a Python environment** and installed and updated the required packages to get ready to see machine learning in action.

1.8 Textbooks Reference

These introductory lessons assume a basic level of statistical and mathematical knowledge. No previous knowledge of Machine Learning is assumed. For this reason I have decided to use ... basic texts for the preparation of these lessons that you can consult for further details on the topics we are going to deal with.

- John C. Hull, **Machine Learning in Business, An Introduction to the World of Data Science**, Amazon (2019)
- Paul Wilmott}, **Machine Learning, An Applied Mathematics Introduction**, Panda Ohana Publishing (2019)

1.9 Appendix A: Machine Learning Terminology

Machine learning is a vast field and also very interdisciplinary as it brings together many scientists from other areas of research. As it happens, many terms and concepts have been rediscovered

or redefined and may already be familiar to you but appear under different names. For your convenience, in the following list, you can find a selection of commonly used terms and their synonyms that you may find useful when reading this book and machine learning literature in general:

- **Training example**: A row in a table representing the dataset and synonymous with an observation, record, instance, or sample (in most contexts, sample refers to a collection of training examples).
- **Training**: Model fitting, for parametric models similar to parameter estimation.
- **Feature**, abbrev. x: A column in a data table or data (design) matrix. Synonymous with predictor, variable, input, attribute, or covariate.
- **Target**, abbrev. y: Synonymous with outcome, output, response variable, dependent variable, (class) label, and ground truth.
- **Loss function**: Often used synonymously with a **cost function**. Sometimes the loss function is also called an error function. In some literature, the term "loss" refers to the loss measured for a single data point, and the cost is a measurement that computes the loss (average or summed) over the entire dataset.