

2022-2023

Big Data analytics

The Supervised Learning Workshop

Chapter 2: Exploratory Data Analysis and Visualization



Introduced to:
Malik Khalfallah

Presented by:
Paloma Puente
Amy Santisteban

INTRODUCTION

This book and chapter simplifies supervised learning for beginners with an interactive, step-by-step approach.

We'll learn when it's used with Python and understand how to automate manual tasks and the data evaluation process using Jupyter and Python libraries like pandas. We'll use data exploration and visualization techniques to develop powerful supervised learning models, before understanding how to distinguish variables and represent their relationships using scatterplots, heatmaps, and boxplots. The goal of this chapter is to gain the skills you need to work on your real-world supervised learning Python projects. Chapter 2 "Exploratory Data Analysis and Visualization". This chapter takes us through how to perform exploration and analysis on a new data set. By the end of this chapter, we will be able to explain the importance of data exploration and communicate summary statistics for a data set. We'll visualize patterns in missing values in the data and be able to replace null values appropriately. We will have the knowledge to identify continuous characteristics, categorical characteristics and visualize distributions of values through individual variables, we will analyze relationships between different types of variables using correlation and visualizations.

In the Exploratory Data Analysis and Visualization chapter there is an exploratory analysis of the data to see what the data can tell us about the relationships between the characteristics and the target variable. Knowing this data will help us interpret the model we built and identify ways we can improve its accuracy. The approach we take to achieve this is:

Exploratory Data Analysis (EDA)

- *Exploratory data analysis (EDA)*

Is the method of analyzing data sets and summarizing their main features to draw useful conclusions, often using visual methods.

The purpose of EDA is:

- Discover patterns within a data set
- Detect anomalies
- Formulate hypotheses about the behavior of the data.
- Validate assumptions

Everything from basic summary statistics to complex visualizations helps us gain an intuitive understanding of the data itself, which is very important when trying to form new hypotheses about the data and discover which parameters affect the target variable. Often finding out how the target variable varies on a single feature gives us an indication of how important a feature might be, and a variation on a combination of multiple features helps us generate ideas for new informative features to design.

To find out what our data really looks like, we use a technique known as data profiling. It is the process of examining available data from an existing information source and compiling statistics or informational summaries on that data. The goal is to ensure that you have a good understanding of your data and can identify any challenges the data may pose early in the project, which is done by summarizing the dataset and evaluating its structure, content, and quality.

Data profiling includes the collection of descriptive statistics and types of data. Common data profile commands include the ones you've seen above, including `data.describe()`, `data.head()`, and `data.tail()`. We can also use `data.info()`, which tells you how many non-null values are in each column, along with the data type of the values (non-numeric types are represented as object types).

Learning objectives

In this module, you will learn to:

- Common data exploration and analysis tasks.
- Using Python packages, such as NumPy, Pandas, Missingno, Seaborn and Matplotlib, and Json file to analyze the data Common data exploration and analysis tasks.

Data exploration with NumPy, Pandas, Matplotlib, Missingno, Seaborn and Json file

Data scientists can use various tools and techniques to explore, visualize, and manipulate data. One of the most common ways that data scientists work with data is through the Python programming language and some specific packages for data processing.

• **What is NumPy ?**

NumPy is a Python library that offers functionality comparable to math tools like MATLAB and R. Although NumPy greatly simplifies the user experience, it also offers rich math functionality.

• **What is Pandas ?**

Pandas is a well-known Python library for data analysis and manipulation. Pandas are like Python's Excel: it provides easy-to-use functionality for data tables.

- **What is Matplotlib ?**

Matplotlib is an open source library that belongs to Python in which you can create animated, static and interactive visualizations in Python.

- **What is Missingno data ?**

Missing data is probably one of the most common issues when working with real datasets. Data can be missing for a multitude of reasons, including sensor failure, data vintage, improper data management, and even human error. Missing data can occur as single values, multiple values within one feature, or entire features may be missing.

- **What is Seaborn ?**

Seaborn is a library for creating statistical graphs in Python. It is based on Matplotlib, and integrates with Pandas structures. This library allows you to capture entire data frames, and built-in semantic mapping and statistical aggregation functions allow you to turn the data into graphical displays.

- **What is Json file?**

JSON is a text format that is part of the JavaScript system and is derived from its syntax, but it is not intended to create programs, but rather to access, store, and exchange data. It is usually known as an alternative to the XML language.

Explore data in a Jupyter Notebook

Jupyter Notebooks are a popular way to run basic scripts using your web browser. Typically, these notebooks are a single web page, divided into sections of text and sections of code that run on the server instead of on the local machine. This means you can get up and running quickly without the need to install Python or other tools.

- **Hypothesis testing**

Data exploration and analysis is typically an iterative process in which the data scientist takes a sample of the data and performs the following tasks to analyze it and test a hypothesis:

- ➔ Clean up data to handle errors, missing values, and other issues.
- ➔ Apply statistical techniques to better understand the data and how the sample can be expected to represent the population of real-world data, taking into account random variation.
- ➔ Visualize the data to determine relationships between variables and, in the case of a machine learning project, identify potentially predictable features of the label.
- ➔ Hypothesis review and repetition of the process.

Exploring Data with Python

A significant part of a data scientist's role is to explore, analyze, and visualize data. There are many tools and programming languages that they can use to do this. One of the most popular approaches is to use Jupyter notebook and Python.

Python is a flexible programming language that is used in a wide range of scenarios from web applications to device programming. It's extremely popular in the data science and machine learning community because of the many packages it supports for data analysis and visualization.

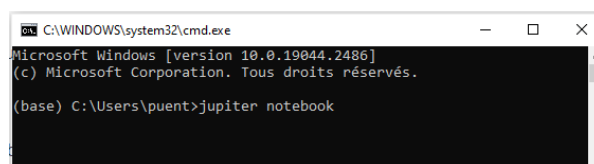
In this notebook, we'll explore some of these packages and apply basic techniques to analyze data. This is not meant to be a full Python programming exercise or even a deep dive into data analysis. Rather, it is intended to be a short course on some of the common ways that data scientists can use Python to work with data.

Before getting into the Jupyter Notebooks environment, there are a few things to keep in mind:

- Notebooks are made of cells. Some cells (like this one) contain discount text, while others (like the one below this one) contain code.
- You can run each code cell using the ► Run button. The ► Run button will appear when you hover over the cell.
- The output of each code cell will be displayed immediately below the cell.
- Although code cells can be executed individually, some variables used in the code are global to the notebook. That means it must execute all the code cells in order. There may be dependencies between cells of code, so if you skip a cell, subsequent cells may not execute correctly.

1. There are two ways to start Jupyter Notebook :

- Using the Anaconda Notice Console: In the search bar, type "Anaconda Notice" and then type "jupyter's notebook"

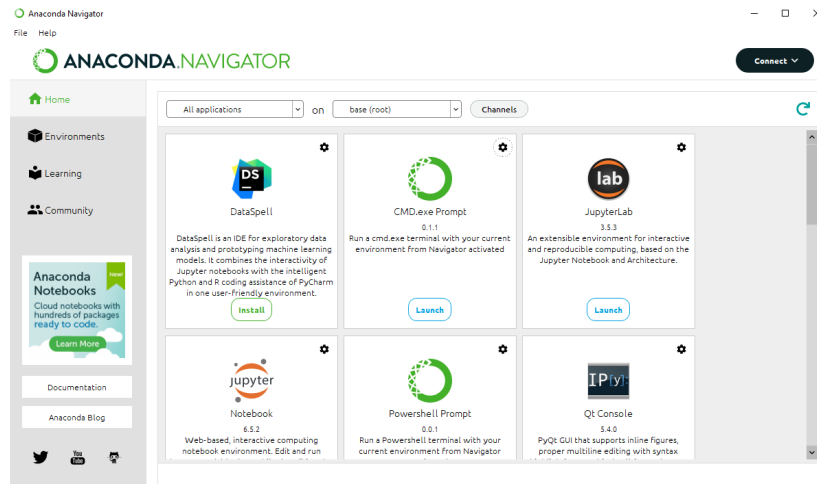


```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [version 10.0.19044.2486]
(c) Microsoft Corporation. Tous droits réservés.

(base) C:\Users\puent>jupyter notebook
```

Source : own elaborated

- Direct method: write directly in the search bar: "Anaconda Navigator". Once the interface is displayed, click start.



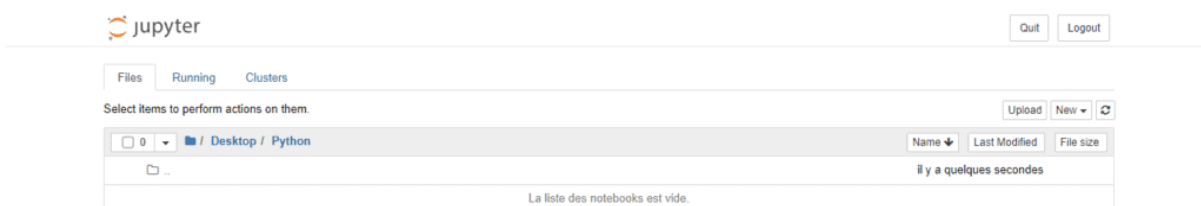
Source : own elaborated

2. After clicking Start, we see this:



Source : Own elaborated

3. You should know that it is a good idea to create a folder on your Desktop to be able to save all your projects and organize your files. To do this, click Desktop, and then browse to the folder you created. Here, we decide to name the folder in question Python:

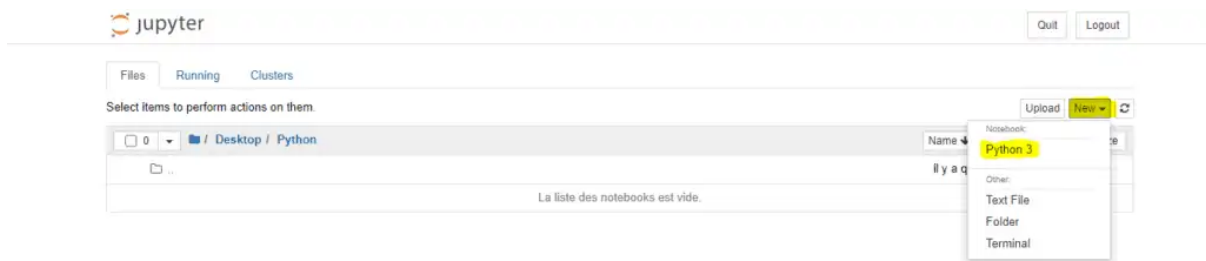


Source : Own elaborated

4. Creating a first Notebook in Anaconda for Python

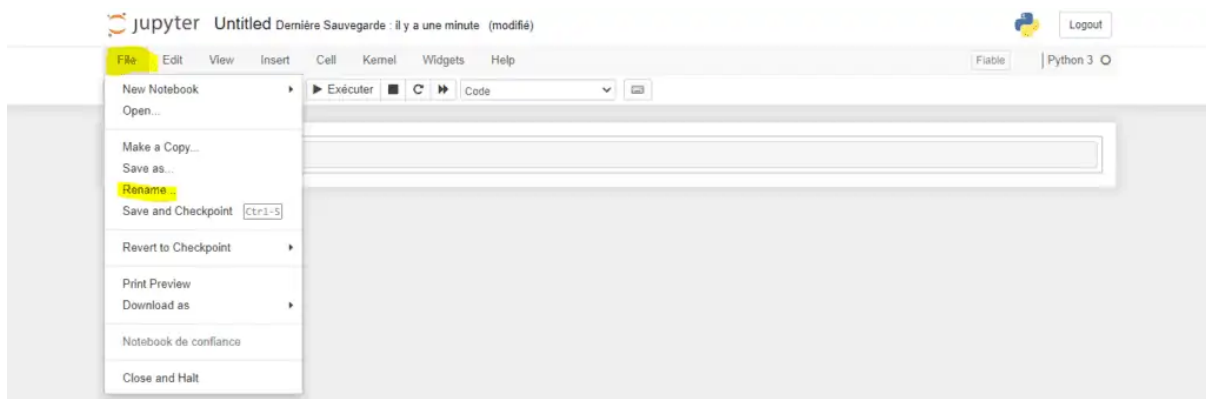
A notebook is a file in which we write the different lines of code. To do this, simply follow the steps below:

- Click New and then Python 3:



Source : Own elaborated

- Then you need to rename the file by clicking on File and then on rename:



Source : Own elaborated

- As for the name of the new file, we will choose, for example, Project1:



Source : Own elaborated

Exemple: exercise 2.01 « Summary of the statistics of our data set »

In this exercise, I'll use the summary statistics functions above to get a basic idea for creating our data set:

Once the Anaconda program is installed, I will be able to access the Jupyter notebook application and among others, which will allow me to code in Python.

I'll start loading the necessary libraries (Missigno, Pandas, Matplotlib, Seaborn) and the Json file into the Anaconda Prompt so the Jupyter Notebook can recognize all data types.

1. I will also have the Github repository that will help me to easily find the data.
2. I will open a new notebook in Jupyter Notebook to start writing my codes following the instructions in the book.
3. In the first three cells, I will write the three codes

```
In [1]: import json
import pandas as pd
import numpy as np
import missingno as msno
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import seaborn as sns

In [2]: with open('../dtypes.json', 'r') as jsonfile:
        dtyp = json.load(jsonfile)

In [3]: dtyp
```

The result is the following figure: Inspection of data types

```
Out[3]: {'id': 'float',
        'flag_tsunami': 'str',
        'year': 'float',
        'month': 'float',
        'day': 'float',
        'hour': 'float',
        'minute': 'float',
        'second': 'float',
        'focal_depth': 'float',
        'eq_primary': 'float',
        'eq_mag_mw': 'float',
        'eq_mag_ms': 'float',
        'eq_mag_mb': 'float',
        'intensity': 'float',
        'country': 'str',
        'state': 'str',
        'location_name': 'str',
        'latitude': 'float',
        'longitude': 'float',
        'region_code': 'str',
        'injuries': 'float',
        'injuries_description': 'str',
        'damage_millions_dollars': 'float',
        'damage_description': 'str',
        'total_injuries': 'float',
        'total_injuries_description': 'str',
        'total_damage_millions_dollars': 'float',
        'total_damage_description': 'str'}
```

4. I will write in the fourth cell the code of the "Earthquakes" database in CSV format and in the fifth cell, I will put the function data.info()

```
In [5]: data = pd.read_csv('../Datasets/earthquake_data.csv', dtype = dtyp)

In [6]: data.info()
```


The result is the following figure: Overview of the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6072 entries, 0 to 6071
Data columns (total 28 columns):
id                6072 non-null float64
flag_tsunami      6072 non-null object
year              6072 non-null float64
month             5667 non-null float64
day               5515 non-null float64
hour              4044 non-null float64
minute            3838 non-null float64
second            2721 non-null float64
focal_depth       3120 non-null float64
eq_primary        4286 non-null float64
eq_mag_mw         1216 non-null float64
eq_mag_ms         2916 non-null float64
eq_mag_mb         1786 non-null float64
intensity         2748 non-null float64
country           6072 non-null object
state             308 non-null object
location_name     6071 non-null object
latitude          6018 non-null float64
longitude         6022 non-null float64
region_code       6072 non-null object
injuries          1169 non-null float64
injuries_description 1349 non-null object
damage_millions_dollars 478 non-null float64
damage_description 4327 non-null object
total_injuries    1184 non-null float64
total_injuries_description 1357 non-null object
total_damage_millions_dollars 418 non-null float64
total_damage_description 3148 non-null object
dtypes: float64(19), object(9)
memory usage: 1.3+ MB
```

5. I print the first five and the last five rows of the dataset. The first five rows are printed as follows:

Here, I put the first code : `data.head()`

I put the second code : `data.tail()`

```
In [8]: data.head()
```

	id	flag_tsunami	year	month	day	hour	minute	second	focal_depth	eq_primary	...	longitude	region_code	injuries	injuries_description	damage_millions_dollars
0	338.0	No	1048.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	120	NaN	NaN	NaN
1	771.0	Tsu	1580.0	4.0	6.0	NaN	NaN	NaN	33.0	6.2	...	1.309	120	NaN	NaN	NaN
2	7889.0	Tsu	1757.0	7.0	15.0	NaN	NaN	NaN	NaN	NaN	...	-6.320	120	NaN	NaN	NaN
3	6697.0	Tsu	1500.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	150	NaN	NaN	NaN
4	6013.0	Tsu	1668.0	4.0	13.0	NaN	NaN	NaN	NaN	NaN	...	-71.050	150	NaN	NaN	NaN

5 rows × 28 columns

```
In [9]: data.tail()
```

	id	flag_tsunami	year	month	day	hour	minute	second	focal_depth	eq_primary	...	longitude	region_code	injuries	injuries_description	damage_millions_dollars
6067	5360.0	Tsu	1993.0	8.0	8.0	8.0	34.0	24.9	59.0	7.8	...	144.801	170	48.0	1	250.
6068	5009.0	No	1983.0	12.0	22.0	1.0	2.0	2.4	26.0	6.4	...	151.868	170	NaN	NaN	25.
6069	10307.0	No	2018.0	2.0	25.0	17.0	44.0	43.0	23.0	7.5	...	142.768	170	300.0	3	61.
6070	5498.0	No	1998.0	7.0	9.0	5.0	19.0	7.3	10.0	6.2	...	-28.626	130	100.0	2	72.
6071	5459.0	No	1997.0	4.0	22.0	9.0	31.0	23.2	5.0	6.7	...	-60.892	90	2.0	1	25.

5 rows × 28 columns

The result is the following figure: The first and last five rows

We can see in these outputs that there are 28 columns, but not all of them are displayed. Only the first 10 and last 10 columns are displayed, with the ellipses representing the fact that there are columns in between that are not displayed.

6. I'll use `data.describe()` to find the summary statistics for the data set. And I run `data.describe().T`:

Here, `.T` indicates that we're taking a transpose of the DataFrame to which it is applied, that is, turning the columns into rows and vice versa. Applying it to the `describe()` function allows us to see the output more easily with each row in the transposed DataFrame now corresponding to the statistics for a single feature.

```
In [10]: data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
id	6072.0	4658.426219	2924.650010	1.000	2142.75000	4608.5	6475.25000	10378.000
year	6072.0	1802.307477	377.924931	-2150.000	1818.00000	1927.0	1986.00000	2018.000
month	5667.0	6.510852	3.450167	1.000	4.00000	7.0	9.00000	12.000
day	5515.0	15.734361	8.752862	1.000	8.00000	16.0	23.00000	31.000
hour	4044.0	11.308605	7.033485	0.000	5.00000	11.0	17.00000	23.000
minute	3838.0	28.855915	17.151545	0.000	14.00000	30.0	44.00000	59.000
second	2721.0	29.740243	17.132196	0.100	14.80000	29.7	44.50000	59.900
focal_depth	3120.0	41.680769	71.258782	0.000	11.00000	26.0	40.00000	675.000
eq_primary	4286.0	6.471419	1.043968	1.600	5.70000	6.5	7.30000	9.500
eq_mag_mw	1216.0	6.526562	0.937869	3.600	5.80000	6.5	7.20000	9.500
eq_mag_ms	2916.0	6.574451	0.989850	2.100	5.80000	6.6	7.30000	9.100
eq_mag_mb	1786.0	5.797592	0.716809	2.100	5.30000	5.8	6.30000	8.200
intensity	2748.0	8.325328	1.800089	2.000	7.00000	8.0	10.00000	12.000
latitude	6018.0	22.537909	22.787934	-62.877	9.87175	32.2	38.77825	73.122
longitude	6022.0	37.985633	86.726852	-179.984	-8.00000	43.3	115.50000	180.000
injuries	1169.0	2293.579127	27095.202227	1.000	10.00000	42.0	200.00000	799000.000
damage_millions_dollars	478.0	1715.606259	12157.409978	0.013	3.62500	20.9	204.35000	220000.000
total_injuries	1184.0	2510.967061	28273.298405	1.000	10.00000	42.5	200.00000	799000.000
total_damage_millions_dollars	418.0	1978.743206	12988.187606	0.010	4.31000	28.0	300.00000	220085.456

The result is the following figure: Summary statistics

Notice here that the `describe()` function only shows the statistics for columns with numerical values. This is because we cannot calculate the statistics for the columns having non-numerical values

CONCLUSION

In summary, in this module, we have learned how to use Python to explore, visualize, and manipulate data. Data exploration is the foundation of data science and a key element in data analysis and machine learning.

Machine learning is a subset of data science that deals with predictive modeling. In other words, machine learning uses data to create predictive models, in order to predict unknown values. You could use machine learning to predict the amount of food a supermarket should order, or to identify plants in photos.

What machine learning does is identify relationships between data values that describe the properties of something, its characteristics, such as the height and color of a plant, and the value to be predicted, the label, such as the species of flowers. These relationships are built into a model through a training process.

Thanks to Python, which is a very useful programming language for data analysis, and also thanks to its powerful libraries and specialized tools. Using NumPy, Pandas, Missingno, Seaborn and Matplotlib, we can load, clean, process, and visualize data in a simple and efficient way, allowing us to get a variable from working presentations like graphs, charts, and images.