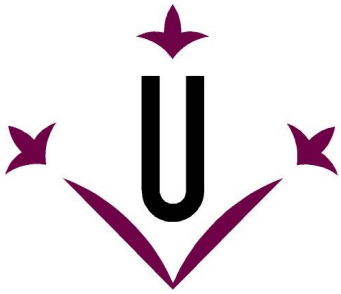

University of Lleida

Master's Degree in Informatics Engineering

Higher Polythecnic School

Big Data Project

Massive Data Processing

Mateu Piñol, Carles

Cores Prado, Fernando

Albert Pérez

Francesc Contreras

Jeongyun Lee

July 13, 2021

Table of contents

List of Figures

1 Introduction

Big Data is a term that describes the sheer volume of data, both structured and unstructured, that floods businesses every day. But it is not the amount of data that is important. What matters with Big Data is what organizations do with the data. Big Data can be analyzed for insights that lead to better decisions and strategic business moves to obtain a competitive advantage. That is why the data is so important.

This project arouses to deepen in how to analyze data, but overall, to better understand the way to draw conclusions as to the main objective.

In this way, this document is a comprehensive report concerned on examining the air quality of Lleida city, where we are currently studying, between 2019 and 2020 since the aim is to know how the pandemic situation affected.

Mainly based on premises such as an article published by BBC news, letting us know the environment in the world has been affected by the decline of human activity caused by the pandemic situation and lockdowns. Therefore, we started this project to figure out how Lleida evolved.

2 Purpose & Objective

In the first instance, the best way to increase knowledge on data analysis is to start working on projects in order to acquire a better complete vision of the possibilities in this area. Therefore, would be enlightening to carry out such a research task.

Hence, learning is the fundamental purpose on which we rely. The beauty of the computer world is that we will never stop learning, sometimes it may seem tedious, but it is definitely an infallible method of personal and professional growth. Even more, when the knowledge acquired belongs to a field of engineering that is currently emerging.

Besides that, as far as the project itself is concerned, the aim is to collect all the possible data related to air quality and generate a realistic and coherent analysis of it. The process consists of different steps: gathering, cleaning, processing, visualize, and finally, extract conclusions that are what we are looking for.

Primarily, we look for changes caused by covid-19 based on the abundance in *Micrograms per Cubic Meter of Air*, or **ug/m3**, of molecular elements in the air such as O_3[1], CO[2], NO_2[3], and SO_2[4], the capacity of which can be captured by specific sensors and thus conclude on how good the air quality is.

3 Document Structure

This section looks forward to narrating what is in each section of the document to help understand and facilitate the approach to the contents.

- **Environment:** brief explanation of the framework used to carry out the project.
- **Data Collection:** description of the process and decisions are taken when gathering and measuring information on variables of interest.
- **Data Cleaning:** narration of how was the first data analysis if any data was corrupted, incorrect, etc.
- **Data Processing:** report how the data was processed in order to get the results and finally plotting them to better visualize how all is related.
- **Results & Analysis:** complete analysis over the results obtained from the data collected, and extracting conclusions which stand as the main objective.
- **Conclusions:** complete end of the line resulting from the study and over the realization of this document.

[1] Ozone
[2] Carbon monoxide
[3] Nitrogen dioxide
[4] Sulfur dioxide

4 Environment

The environment runs upon **Python 3** interpreter, and we considered using the packed manager *Anaconda* to generate the python environment as embedding all the non-built-in python dependencies using the "environment.yml" file. To set up the environment, just run the *Makefile* and it will check all the requirements and getting all the dependencies required to run the project.

Apart from that, to set up the spark context we used the *docker* application to establish the *jupyter/all-spark-notebook* image container where runnnig isolated the notebooks requiring the python *pyspark* module to process all the data cleaned.

Furthermore, the project structure is based on the command-line utility *cookiecutter* that creates a logical, reasonably standarized, but flexible structure for doing and sharing data science works.

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.py           <- makes project pip installable (pip install -e .) so src can be imported
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.readthedocs.io
```

Figure 1: Project directory structure

5 Data Collection

Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes.

There are methods to collect data such as surveys, online tracking, social media monitoring, although by seeking sources, we infer the best is to use open data to obtain the required information to develop a consistent and coherent analysis of the air quality.

The first idea was to use different sources and unify all the data collected to get more precise information resulting in better and more realistic results. However, we find the platforms considered, use different models, categorizations, and formats to provide the data, for instance, JSON[5], XML[6], CSV[7], and others. So, maybe quite difficult to harmonize the data assembled as they expect to be gorgeous different.

In addition, there were platforms not providing enough or useful data, like the location, units, or incomplete values. Meaning, there was not homogeneous data, that's why the possibility to harmonize it all could be exhausting and take expensive time.

Therefore, we committed to working with only one data source, that being the OpenAQ. OpenAQ is a non-profit organization empowering communities around the globe to clean their air by harmonizing, sharing, and using open-air quality data, which means we have access to a lot of data sets unified, and accessible via web service. Further, it has different ways to obtain the data: OpenAQ API, download directly the data sets from specific selection parameters or use a web client to access the data.

In our case, we stake for building a python script called *get_data.py* to via HTTP[8] requests fetch the resources needed from the OpenAQ API and save them in JSON format.

Initially, we explored the API methods from which get the data and how they work. Even, considering what kind of data would be interesting to examine and compare. First, we focused on the countries or the biggest cities in order to compare among them their pollution indeed the pandemic situation. But, we rejected this idea because we don't have the enough computational power, at least in our home PCs, to process this considerable volume of data which extends through millions of records.

Then, it was when came up to center our attention on the city of Lleida and use the data between 2019 and 2020 provided by the sensors expressed as time series, to make a comparison of the evolution of the air quality when normal circumstances in contrast during the pandemic situation.

[5] JavaScript Object Notation
[6] Extensible Markup Language
[7] Comma-Separated Values
[8] Hypertext Transfer Protocol

What data do we collect?

- **Parameters:** to know the units and types from which select those fitting our objective saved as "parameters.json". Mainly the O_3, CO, NO_2 and SO_2 in ug/m3.
- **Locations:** containing the sensors information of the Lleida city saved as "locations.json".
- **Measurements:** for each sensor get the data collected during 2019 and 2020 saved as "measuraments_ES_<sensor_name>.json".

Note > The data files are saved on the ../../data/raw directory following the *cookiecutter* template showed up on figure 1 on page 3.

To get more information, the figure 2 presents the script usage, or you may consult the code repository, the link to it can be found in section 11 on page 14.

```
PS C:\Repositori\airQuality> python .\src\data\get_data.py -h
usage: Get data script [-h] [-p PATH] [-v]

Script to get the data about air quality of Lleida using the OpenAQ API service which collects data
 from air sensors located.

optional arguments:
  -h, --help            show this help message and exit
  -p PATH, --path PATH  Custom save data path directory.
  -v, --verbose         Display monitoring details and create a logging file.
```

Figure 2: Get data script usage

Additionally, was built a log system when executing the script using the optional argument verbose. In this way, you will see information and debug entries on the terminal when executing but also we will be created a log file on ./logs/ directory which you may see an example on the appendix A on page 15.

6 Data Cleaning

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a data set.

In such a way to clean the data obtained, a *Jupyter* notebook called "Clean Air Quality" was created that uses the data provided by the OpenAQ API and establishes the first contact by analyzing the structure and values.

Once the data was loaded into the notebook kernel we reviewed its structure, and in general, the data does not have empty values, suspicious outliers, or other data types related problems. Likewise, all cases follow a well-designed model easy to understand with no wrong data types, but moreover, to get more deepen analysis we applied profiling reports to the data sets, thanks to the python pandas-profiling module, to pick up information related to outliers, missing values, and other non-correct data.

Nevertheless, we find some warnings, for instance, some missing values on the "maxColorValue" parameters field, but these were not important at all because they are not useful for our analysis. However, on the measurements data, it can be seen that the sensors do not worked properly or have some trouble during the period between mid-June 2019 to early December 2019. The data takes negative values, which does not make sense in this case when considering unit capacities (ug/m3). For this reason, we have considered dismissing this chunk of data.

Also, there are some records with zero-values that may be related to the sensors trouble during the period mentioned but we maintained them as regarding capacities this ones may result equal to zero. Although, searching information why the sensors got in trouble or the meaning of this zero measurements, there were no source of information providing hints.

As well, we structured the data and selected only those schema fields needed for our analysis. The following figures 3, 4, and 5 on page 7, and 7 represents for each data set the final schema resulting after the filtering.

Besides, cleaning the data has helped us to better understand the structure of the data and how is modelled, to know exactly what parameters we can use, and how the measurements are presented with respect to the city of Lleida for each of the sensors.

You may check the notebook and the reports which were saved in the ./reports directory, both in JSON and HTML[9] format. In addition, the cleaned data is stored in the ./raw/clean directory for further processing.

[9] Hypertext Markup Language

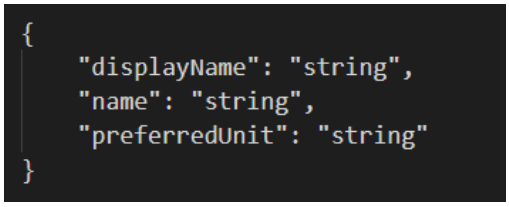

Figure 3: Resulting parameters schema

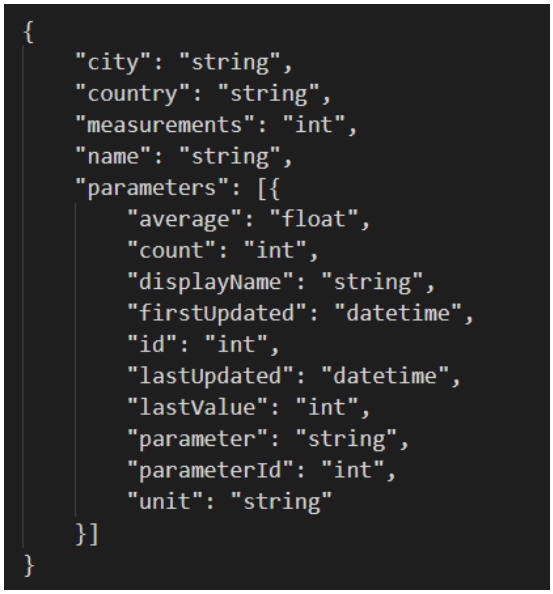

Figure 4: Resulting locations schema

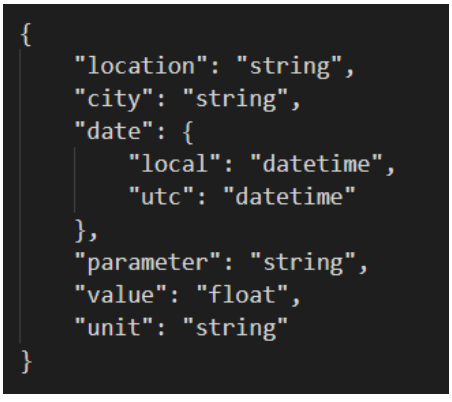

Figure 5: Resulting measurements schema

7 Data Processing

Data processing occurs when data is collected and translated into usable information. It is important for data processing to be done correctly as not to negatively affect the conclusions. Is the most important step.

Again making use of a *Jupyter* notebook to work, but, this time performing the computations inside the *docker jupyter/all-spark-notebook* container to process the data with the python *pandas* and *pyspark* modules, and finally, visualize the results generating the plots with *matplotlib*.

First of all, the cleaned data is read and a dictionary is created that uses as a key the available sensors. Then, for each sensor, we consider saving a *Spark Dataframe* and aggregate the data by month, thus, allowing us to compare for each month the values during the years. Afterward, all the data frames are joined, performing an union all, and filtered by the selected parameters (NO_2, CO, $S0_2$, O_3) saving them to a new dictionary, using the parameters name as a key.

To sum up, the plan is to visualize for each parameter how unfold the data but also comparing the sensors' measurements. In this way, we will get a comparison not only in the evolution of the molecular elements over the months, thus, resulting in how affected the pandemic situation, but also, about the precision or differentiation among the sensors, for further analysis, as they are located in different parts of the city.

It is true, at the beginning, we considered to carry out an aggregation by averaging the sensors values, but as seen, they may provide significant different measurements, even though they all follow the same tendency, it is also good to visualize this contrast. Then, it would not be accurate nor fair to make an average of the values collected in the same period of time.

But that is not all, we wanted to go further even visualizing clearly how the pandemic affected, we considered comparing explicitly 2019 and 2020 maximum absolute values. Why max? Simply because these values are real measures, and even recording lower values do not imply the reality of others. Finally, we generated some bar plots comparing the max values for each month to see how the absolute values evolve.

8 Results

This section presents the resulting plots from which understand the measurements collected. First, you may find the plots regarding what data collected the sensors and how they evolve during the period selected.

And then, there is the maximum measurements comparison using bar plots to visualize better the comparison for each month.

Take a look, that there exist a lack of data during the period between mid-June 2019 and early December 2019, therefore, we can not make comparison for this period of time.

8.1 Sensors measurements

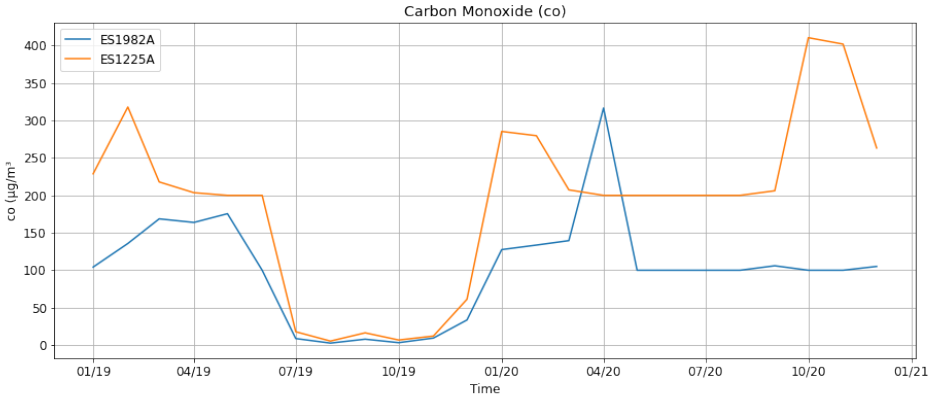

Figure 6: *CO* Lleida city sensors values 2019-2020

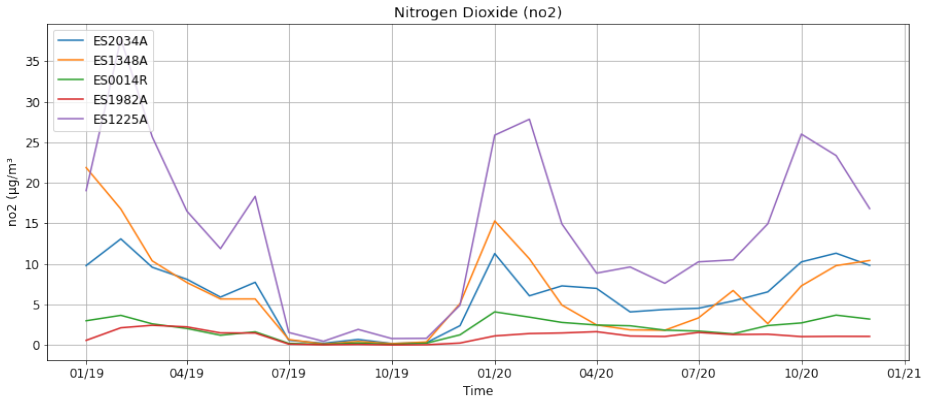

Figure 7: NO_2 Lleida city sensors values 2019-2020

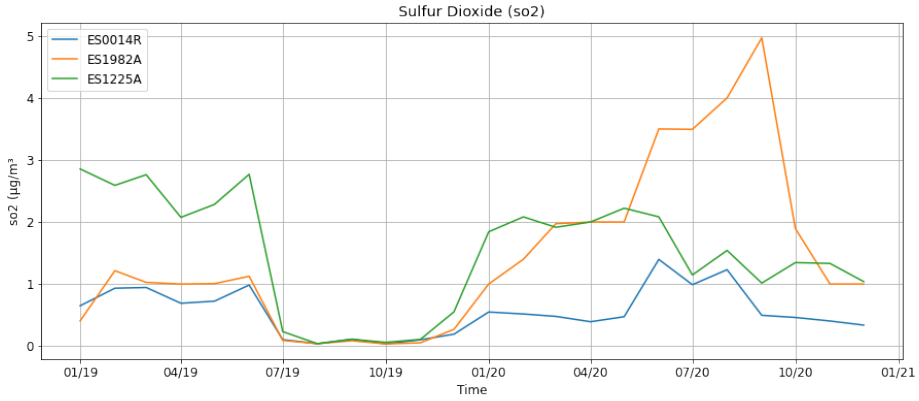

Figure 8: SO_2 Lleida city sensors values 2019-2020

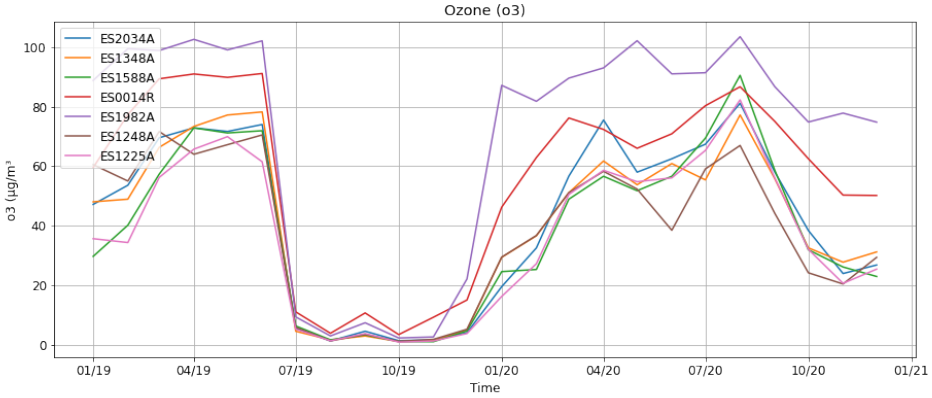

Figure 9: O_3 Lleida city sensors values 2019-2020

8.2 Maximum measurements per month

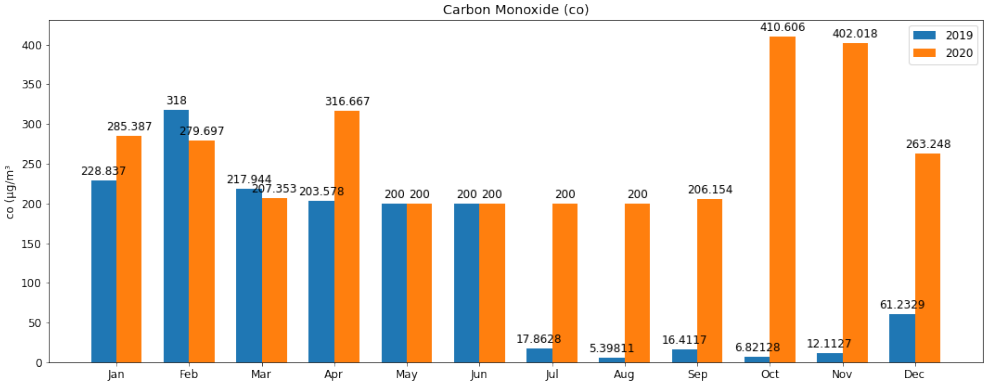

Figure 10: Max. *CO* Lleida city value 2019-2020

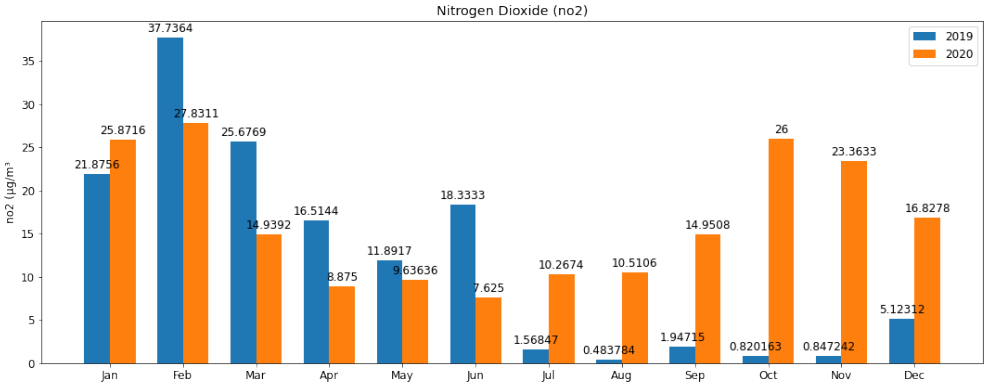

Figure 11: Max. NO_2 Lleida city value 2019-2020

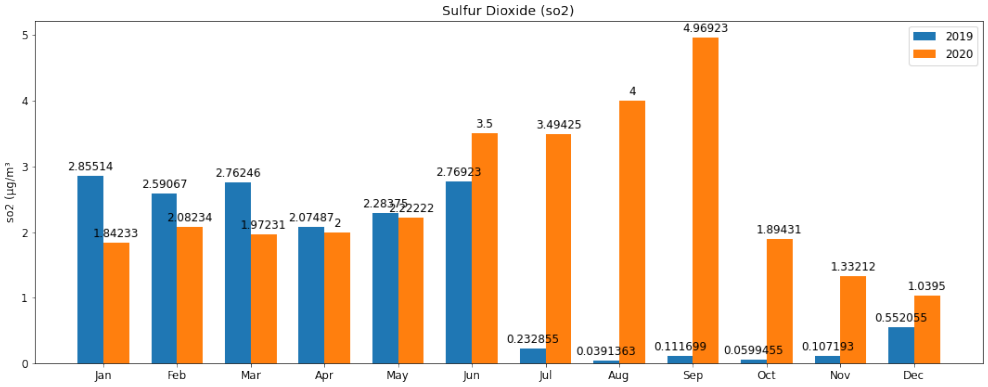

Figure 12: Max. SO_2 Lleida city value 2019-2020

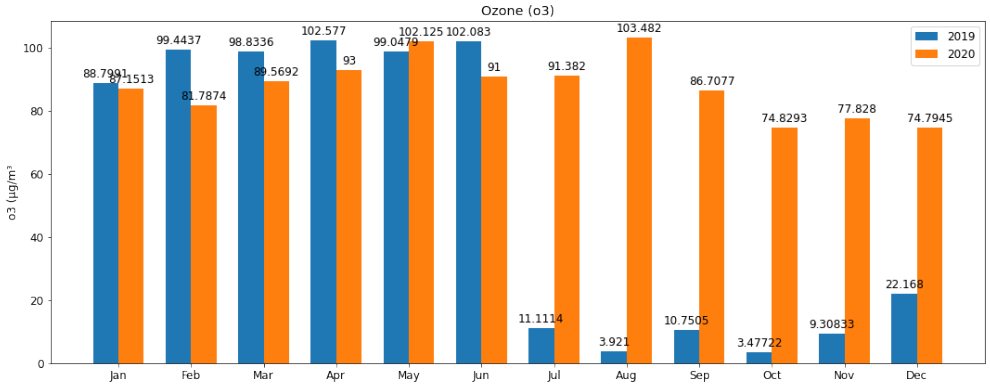

Figure 13: Max. O_3 Lleida city value 2019-2020

9 Analysis

This is focused on observing what changes have occurred during this period by comparing monthly values for each component based on the obtained result from 2019 to 2020.

As we can see O_3, SO_2, NO_2 and CO decreased during the lockdown period (February 15th 2020 to June 21st 2020), meaning that the quality of the air improved. This is clearly seen in the carbon monoxide and nitrogen dioxide plots (figures 6 and 7) where there is a considerable drop down in the values which directly affects the quality of air.

In addition, the sulfur dioxide and the ozone have low down tendencies compared to the 2019 first half period but the measures are not as clear as the other molecular elements.

Also, regarding the bar plots, the first half of the year we can observe how the values are relatively lower. For instance, on February there is a significant difference about the nitrogen dioxide, but it is also clear that all measures are relatively lower compared to 2019 which fits perfectly our expectations.

However, we do not know if the quality of the air decreases or increase when the lock-down ends, this is because we have a lack of data during mid-June 2019 to early December 2019, and therefore, we cannot make comparisons for this period of time. As said in previous sections, we do not know why this happened even seeking for answers, but the important periods, luckily, were recorded.

In conclusion, the data collected shows quite good how the pandemic situation reduced the overall pollution in the air, meaning a better air quality during the lock-downs. Also, is so good that this results match what we expected to find based on multiple news and articles supporting the kind of theories, but now, facts.

10 Conclusions

Carrying out this study has been really enriching for our professional careers as data scientists. Just like an introduction to this fascinating area.

We were looking for specific results, as when reading, even may be obvious and logical, that the pandemia affected on how the world works, and reflecting it, just like in this case, in the improvement of the air quality.

The great point is that finally, even some lack of information, we obtained valid results confirming our thoughts. But remarking that having consistent and decent quality data without lack of information and a good model helps a lot when it comes to analyzing and treating amounts of information. Because when there empty values, there is no possible comparison or conclusions to extract. Is the same as having no data.

Endlessly, to conclude, we wanted to emphasize how this work allowed us to put the worthy collection of knowledge acquired during the course into practice.

References

[1] OpenAirQuality API

[2] Cookiecutter github repository

[3] Cookiecutter documentation

[4] Covid BBC Article: Then and now: Pandemic clears the air

[5] El Confidencial: Lockdown Article

11 Code repository

BigData project: AirQuality

Appendices

A Get data log example

```
2021-07-03 11:44:10,169 INFO Starting...
2021-07-03 11:44:10,169 INFO Checking path...
2021-07-03 11:44:10,190 INFO Path checked.
2021-07-03 11:44:10,191 INFO Getting countries...
2021-07-03 11:44:10,194 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:10,813 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/countries?limit=200&order_by=country&sort=asc HTTP/1.1" 200 4493
2021-07-03 11:44:10,821 INFO Countries got.
2021-07-03 11:44:10,828 INFO countries.json saved.
2021-07-03 11:44:10,829 INFO Getting parameters...
2021-07-03 11:44:10,831 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:11,378 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/parameters?limit=100000&sort=asc HTTP/1.1" 200 746
2021-07-03 11:44:11,384 INFO Parameters got.
2021-07-03 11:44:11,386 INFO parameters.json saved.
2021-07-03 11:44:11,386 INFO Getting locations...
2021-07-03 11:44:11,388 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:11,965 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/locations?limit=1000&page=1&country_id=ES&city=Lleida HTTP/1.1" 200 1194
2021-07-03 11:44:11,986 INFO locations_ES_Lleida.json saved.
2021-07-03 11:44:11,987 INFO Locations found: ['ES1248A', 'ES1348A', 'ES1225A', 'ES1588A', 'ES1982A', 'ES2034A', 'ES0014R']
2021-07-03 11:44:11,988 INFO Getting [ES1248A] measuraments...
2021-07-03 11:44:11,991 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:15,352 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/measurements?limit=100000&page=1&country_id=ES&date_from=2019-01-01&date_to=2020-12-31&city=Lleida&location=ES1248A HTTP/1.1" 200 76430
2021-07-03 11:44:16,218 INFO measurements_ES_ES1248A.json saved.
2021-07-03 11:44:16,227 INFO Getting [ES1348A] measuraments...
2021-07-03 11:44:16,229 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:20,141 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/measurements?limit=100000&page=1&country_id=ES&date_from=2019-01-01&date_to=2020-12-31&city=Lleida&location=ES1348A HTTP/1.1" 200 121919
2021-07-03 11:44:21,775 INFO measurements_ES_ES1348A.json saved.
2021-07-03 11:44:21,793 INFO Getting [ES1225A] measuraments...
2021-07-03 11:44:21,795 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:30,399 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/measurements?limit=100000&page=1&country_id=ES&date_from=2019-01-01&date_to=2020-12-31&city=Lleida&location=ES1225A HTTP/1.1" 200 195212
2021-07-03 11:44:33,407 INFO measurements_ES_ES1225A.json saved.
2021-07-03 11:44:33,464 INFO Getting [ES1588A] measuraments...
2021-07-03 11:44:33,467 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:36,243 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/measurements?limit=100000&page=1&country_id=ES&date_from=2019-01-01&date_to=2020-12-31&city=Lleida&location=ES1588A HTTP/1.1" 200 77994
2021-07-03 11:44:37,155 INFO measurements_ES_ES1588A.json saved.
2021-07-03 11:44:37,165 INFO Getting [ES1982A] measuraments...
2021-07-03 11:44:37,167 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:44,317 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/measurements?limit=100000&page=1&country_id=ES&date_from=2019-01-01&date_to=2020-12-31&city=Lleida&location=ES1982A HTTP/1.1" 200 188333
2021-07-03 11:44:47,397 INFO measurements_ES_ES1982A.json saved.
2021-07-03 11:44:47,432 INFO Getting [ES2034A] measuraments...
2021-07-03 11:44:47,442 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:51,024 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/measurements?limit=100000&page=1&country_id=ES&date_from=2019-01-01&date_to=2020-12-31&city=Lleida&location=ES2034A HTTP/1.1" 200 121677
2021-07-03 11:44:52,853 INFO measurements_ES_ES2034A.json saved.
2021-07-03 11:44:52,869 INFO Getting [ES0014R] measuraments...
2021-07-03 11:44:52,872 DEBUG Starting new HTTPS connection (1): u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443
2021-07-03 11:44:59,244 DEBUG https://u50g7n0cbj.execute-api.us-east-1.amazonaws.com:443 "GET /v2/measurements?limit=100000&page=1&country_id=ES&date_from=2019-01-01&date_to=2020-12-31&city=Lleida&location=ES0014R HTTP/1.1" 200 196795
2021-07-03 11:45:02,149 INFO measurements_ES_ES0014R.json saved.
2021-07-03 11:45:02,179 INFO Completed.
--------------------------------------------------------------------------------------------
```

Figure 14: Get data script log