POLITECNICO MILANO 1863

Software Engineering for Geoinformatics 2023

Software Release Document

EarthTech Group SRD Presentation by

Mohammad Kordgholiabad

Fatemeh Soleimaniansomarin

Ghulam Abbas Zafari

EarthTech

Air Quality Monotoring

Deliverable: SRD

Title: Software Release Document

Authors: Mohammad Kordgholiabad, Fatemeh Soliemaniansomarin
Ghulam Abbas Zafari

Version: 1.0
Date: Jun 18, 2023

Revision history

Version	Date	Change
1.0	18/06/2023	First submitted version

Table of Contents

1 Introduction

1.1 Introduction:

This is the first release of our interactive Jupyter dashboard application! This document provides an overview of the features and functionality included in this initial release. Our application aims to provide easy access to air quality data collected from ground sensor stations in the Lombardy region. By interactive visualizations, our dashboard empowers public authorities and citizens to gain valuable insights into air pollution trends and details.

1.2 Short Overview:

Air pollution poses a significant challenge in today's world, requiring collective efforts to address its impact on the environment and human health. Our interactive Jupyter dashboard application serves as a valuable tool for accessing and analyzing air quality data in the Lombardy region. In this first release, we provide users with a stable and reliable version of the application.

With this release, users can explore air quality measurements collected throughout the year of 2022, including pollutants such as Benzene, nitrogen dioxide, sulfur dioxide, carbon monoxide, ozone, and nitrogen oxides. The dashboard offers intuitive visualizations and interactive features, enabling users to identify pollution patterns.

2 Features

2.1 Overview of Stations

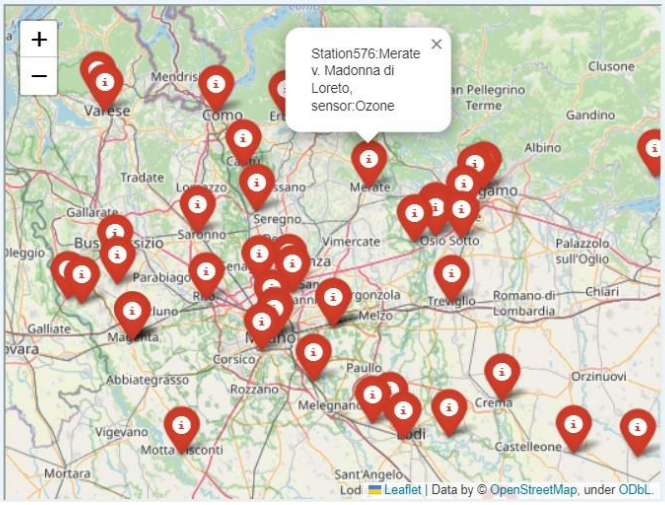
The initial section of our interactive Jupyter dashboard provides an overview of the station distribution in the Lombardi region. This section offers users two distinct maps.

In the first map, labeled "Stations and Sensor Types," users can observe the spatial distribution of stations across the Lombardi region. By interacting with the map, users have the ability to click on individual stations and retrieve information such as the station name and the corresponding sensor type. Each sensor type is responsible for monitoring a specific pollutant in the region.

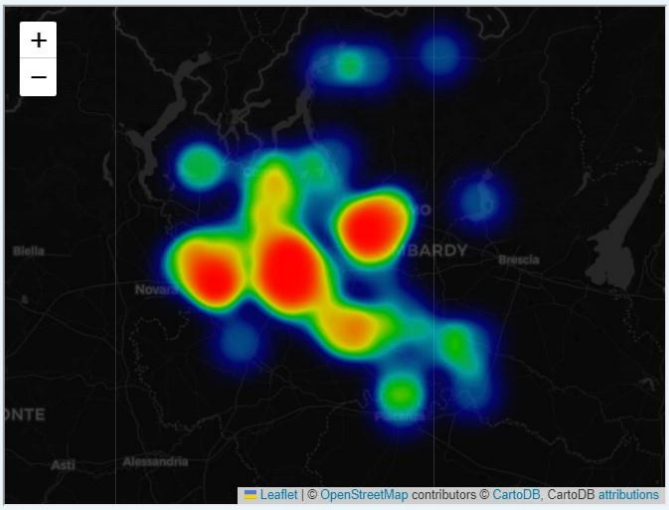
The second map in this section labeled "Station Density," presents a heat map, which visualizes the density of stations across the Lombardi region. By examining this heat map, users can gain insights into the concentration and clustering of stations in different areas of Lombardi.

These maps serve as informative tools for users to explore the station network and understand the distribution of monitoring stations and their associated sensor types within the Lombardi region.

Stations and Sensor Types



Stations Density



2.2 Statistics

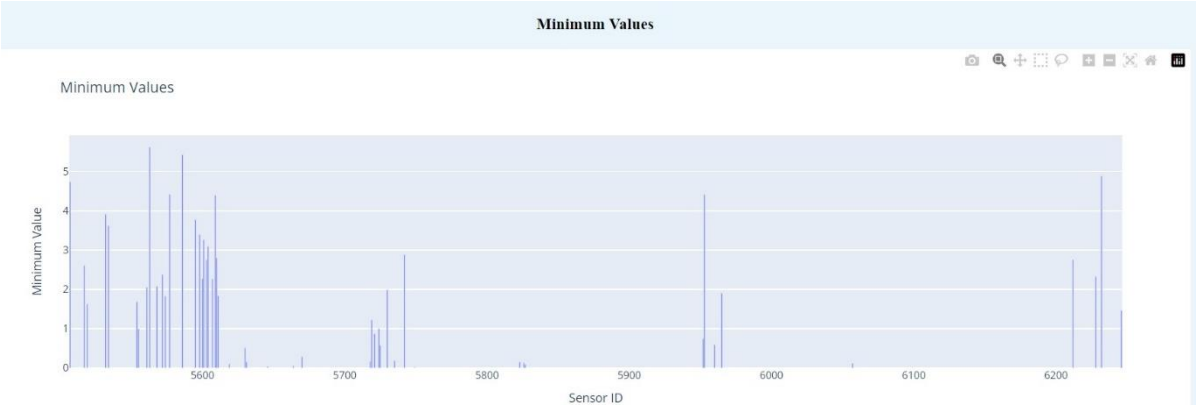
In the "Statistics" section, users are presented with three informative graphs.

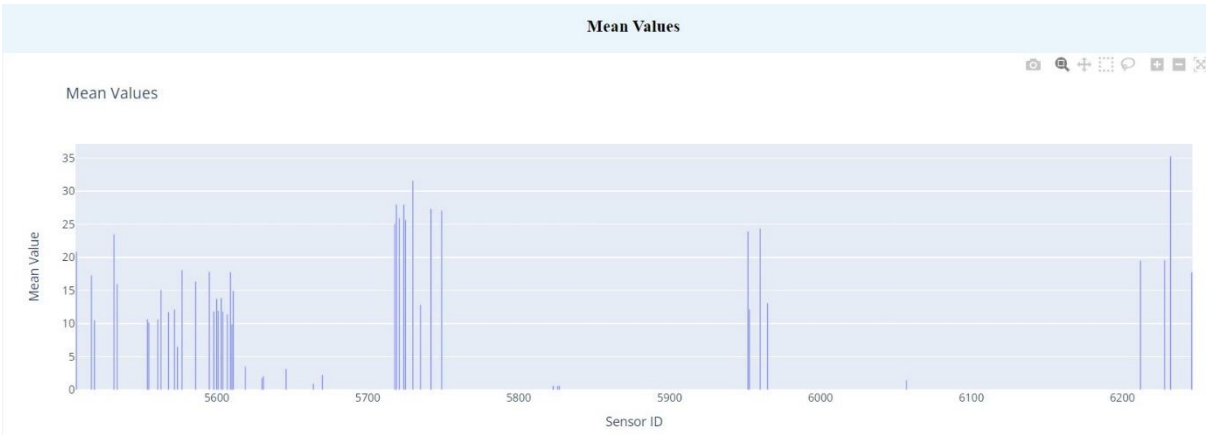
The first graph displays the minimum values observed by each sensor. This graph provides users with insights into the lowest recorded values for each sensor over a given period.

The second graph represents the maximum values observed by each sensor. By analysing this graph, users can gain an understanding of the highest recorded values for each sensor during the specified timeframe.

The third graph illustrates the mean values of each sensor. This graph enables users to comprehend the average values recorded by each sensor, thereby offering insights into the general observation patterns within the specified period.

These graphs serve as valuable tools for users to analyse and interpret the statistical characteristics of sensor observations. By examining the minimum, maximum, and mean values, users can better understand the typical observation ranges and tendencies for each sensor over the designated time period.





2.3 Sensors Map

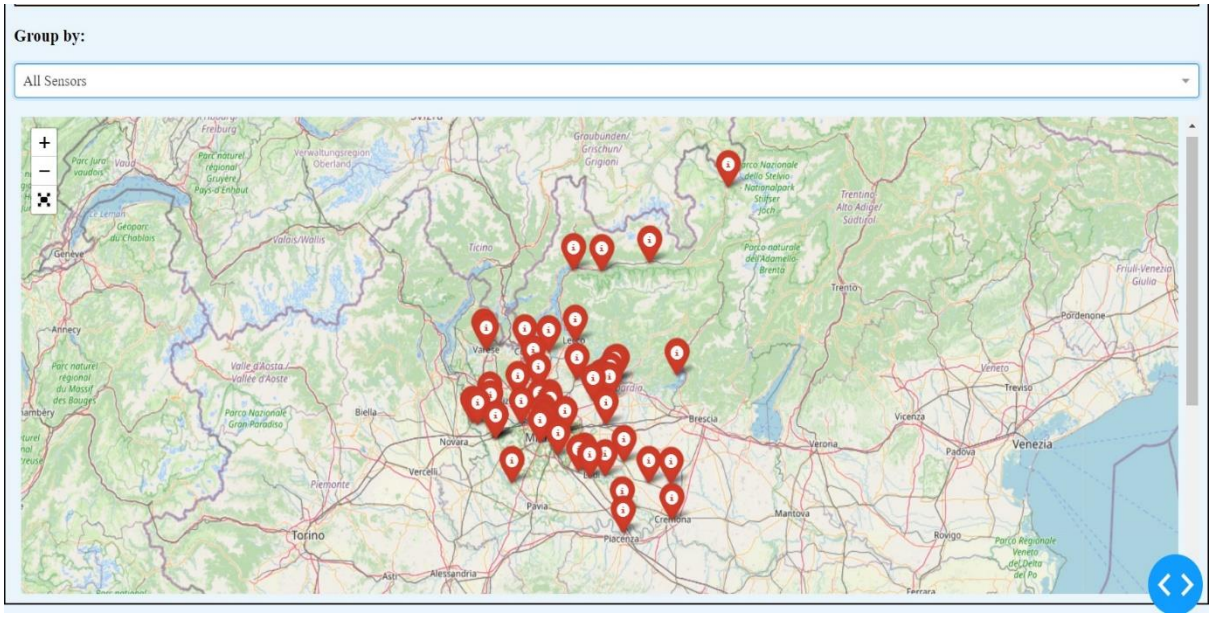
This section is designed and implemented to provide users with detailed information about the sensors. It offers several key features to enhance user experience and facilitate the exploration of sensor data.

To begin with, users can utilize a dropdown menu to sort sensors based on the province they are located in or the type of pollution they measure. This sorting functionality allows users to quickly identify and focus on specific sensors of interest. Additionally, there is an option labeled "All sensors" which enables users to view all sensors on the map simultaneously, providing a comprehensive overview.

By clicking on individual sensors, users can access a popup window that displays detailed information about each sensor, including the sensor ID, Sensor Type, Sensor Province, and Comune. This feature enables users to gain insights into the specific attributes and locations of the sensors.

Once users have selected their desired sensor, they can utilize the subsequent feature to plot the data associated with that sensor. These two features complement each other, allowing users to visualize and analyse the data collected by their chosen sensor. The mapping component of this feature is powered by Folium, which provides additional functionalities such as zoom control, full-screen mode, and a scale bar.

Overall, this section offers a user-friendly interface that facilitates the exploration and understanding of sensor details. The combination of sorting options, detailed sensor information, and data plotting capabilities empowers users to effectively analyze the sensor data within the Lombardi region.



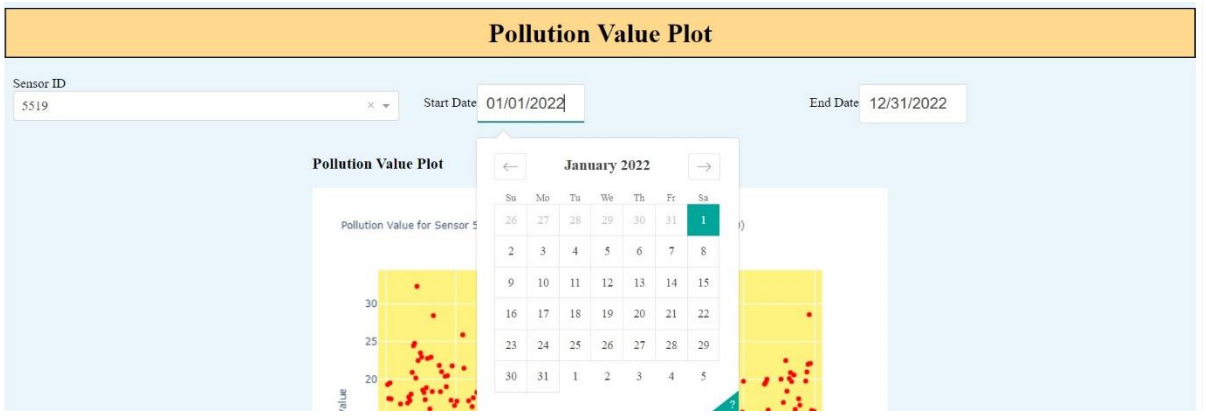
2.4 Pollution Value Plot

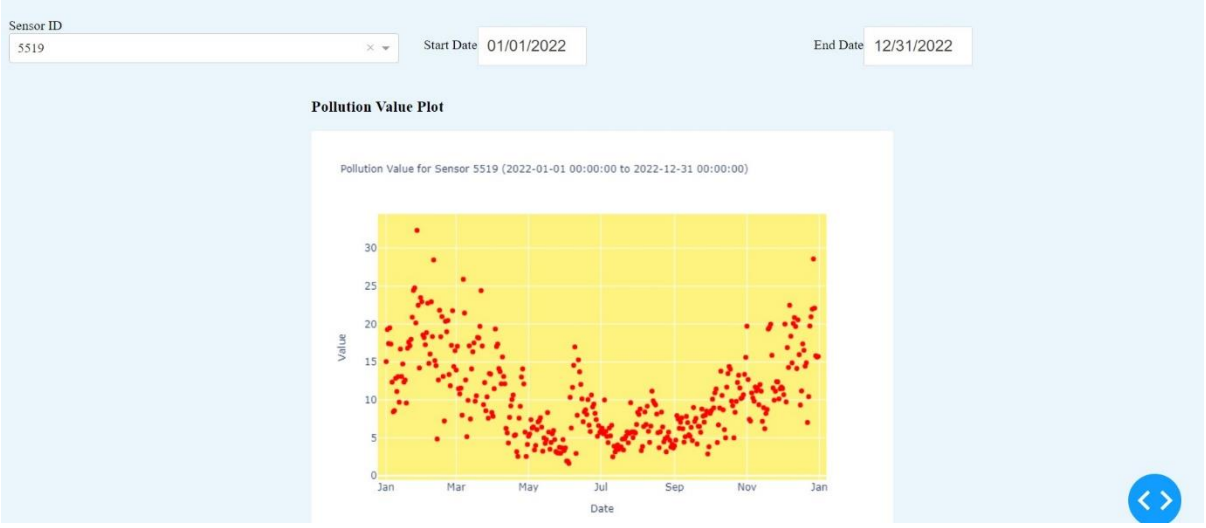
This section consists of three distinct parts. Firstly, users can reference the information about their desired sensors from the previous section. They can select the specific sensor ID from a drop-down menu provided.

Next, users are given the option to choose a start date and end date for their selected sensor's observations. This is facilitated through the use of date pickers, allowing users to specify a time period within the year 2022.

Once the users have chosen their desired sensor and selected the appropriate dates, a plot will be generated on the dashboard. This plot will visually represent the observations recorded by the chosen sensor, displaying them as red dots on the graph. The plot will accurately reflect the observations made during the specified time period.

This feature provides users with the ability to delve deeper into the sensor data by focusing on a specific sensor and examining its observations within a defined timeframe. The interactive nature of the dashboard enhances the user experience and enables users to gain valuable insights into the sensor's measurements over time.





3 Limitations

The primary limitation of our dashboard is the availability of data. Currently, the dashboard exclusively utilizes data from the year 2022. As a result, users are restricted to selecting dates solely within that particular year using the date pickers.

It is important to note that while the dashboard offers various features and functionalities, such as sensor sorting, detailed sensor information, and data plotting, these capabilities are limited to the data from the year 2022. Developers have designed and implemented the dashboard with the intention of making all features fully functional, given the availability of data from other years.

In future iterations or updates of the dashboard, it is possible for additional data from different years to be incorporated. This would allow users to choose dates beyond the constraints of the year 2022, providing them with a broader range of temporal analysis and insights.

The current version of the dashboard serves as a foundation, offering a glimpse of its potential and the value it can provide to users. However, its limitations regarding data availability should be taken into consideration when utilizing and interpreting the provided features and functionalities.

4 Installation Guide

The dashboard we have developed is an offline application that operates on the user's local host. To access and utilize the dashboard on their PC, users can follow these steps:

To utilize the dataset provided, users should have access to two Excel files: "Data.xls" and "Sensors.xlsx". These files contain the essential data required for the proper functioning of the dashboard.

Before proceeding, users must ensure that they have set up a PostgreSQL environment. They should modify the "Data Base.py" code accordingly. Within this code, users need to update the data directory path to point to the location where the "Data.xls" and "Sensors.xlsx" files are stored. Additionally, users should set the appropriate password for their PostgreSQL environment. By running this modified code, the necessary tables will be created, utilizing the data from the provided Excel files.

Once the dataset is ready, users can proceed to execute the "Backend.py" code within their environment. Prior to running this code, it is crucial to have the required libraries installed. These libraries include pandas, Flask and jsonify and request (from flask), create_engine, and text (from SQLAlchemy).

Upon successful execution of the backend code, users will observe the following message displayed in the console:

Running on http://127.0.0.1:5005.

In the final step, users should have an Anaconda environment installed on their system. They can then run the "Dash.ipynb" file. Running this file will generate a link as the output. By clicking on this link, users will be redirected to a new tab, which serves as the dashboard interface. From there, users can fully explore and utilize the features and functionalities of the dashboard.

Please ensure that you have the "Data.xls" and "Sensors.xls" files accessible and correctly located in your system. Also, make sure to follow the instructions provided in the "Data Base.py", "Backend.py", and "Dash.ipynb" codes accurately, including modifying the necessary paths and setting up the PostgreSQL environment as instructed.

5 Test Execution and Reporting

This report indicates the test set applied to version 1.0 of the air quality interactive dashboard.

The report defines for each test case of the test set:

- The date of test execution.
- The set of inputs given to the system.
- The initial state of the system at the time of the test case execution.
- The expected output.
- The actual output.
- Whether the test has been successful or not.

The following tables are the report of each test case of the set.

Test Case	Date:18.06.23		
	Input(s):Sensors.xls		
test_AllSensors_data	Expected Outputs	Actual Outputs	Condition
	Users send request to get the data about the sensors and the endpoint return the table of sensors data.	Users send request to get the data about the sensors and the endpoint return the table of sensors data.	Successful

Test Case	Date: 18.06.23		
	Input(s): Data.xls		
test_AllAQdata	Expected Outputs	Actual Outputs	Condition
	Users request to get the detailed observation data that observed by sensors and the endpoint return that data.	Users request to get the detailed observation data that observed by sensors and the endpoint return that data.	Successful

Test Case	Date: 18.06.23		
	Input(s): sensor and data table		
test_AllData	Expected Outputs	Actual Outputs	Condition
	Users send request to merge the sensors and data tables and the endpoint return the merged AllData table.	Users send request to merge the sensors and data tables and the endpoint return the merged AllData table.	Successful

Test Case	Date: 18.06.23		
	Input(s): "station_name": "Milano v.Senato"		
test_station_filter	Expected Outputs	Actual Outputs	Condition
	This station name as sample provided. The endpoint should return the station with this name, and it has json format and it is not empty.	This station name as sample provided. The endpoint should return the station with this name and it has json format and it is not empty.	Successful

Test Case	Date: 18.06.23		
	Input(s): "sensor_type": "Benzene"		
test_sensor_type_filter	Expected Outputs	Actual Outputs	Condition
	This sensor type as sample provided. The endpoint should return the sensors with this type, and it has json format and it is not empty.	This sensor type as sample provided. The endpoint should return the sensors with this type, and it has json format and it is not empty.	Successful

Test Case	Date: 18.06.23		
	Input(s): "comune": "Como"		
test_comune_filter	Expected Outputs	Actual Outputs	Condition
	This Comune as sample provided. The endpoint should return the sensors from this comune, and it has json format and it is not empty.	This Comune as sample provided. The endpoint should return the sensors from this comune, and it has json format and it is not empty.	Successful

Test Case	Date: 18.06.23		
	Input(s): "Province": "MI"		
test_province_filter	Expected Outputs	Actual Outputs	Condition
	This province as sample provided. The endpoint should return the sensors from this province, and it has json format and it is not empty.	This province as sample provided. The endpoint should return the sensors from this province, and it has json format and it is not empty.	Successful

6 Bibliography

Atlassian's "Release Notes Guide"

PostgreSQL Documentation. (2022). PostgreSQL: The World's Most Advanced Open-Source Relational Database. Retrieved from https://www.postgresql.org/docs/

Flask Documentation. (2022). Flask: A Microframework for Python. Retrieved from https://flask.palletsprojects.com/

https://cordis.europa.eu/docs/projects/cnect/3/609023/080/deliverables/001-MYWAYD21RequirementSpecificationAndAnalysiscnectddg1h520142537023.pdf

Robert L. Lux and C. Arden. Pope. Air quality index. U.S. Environmental Protection Agency, 2009.