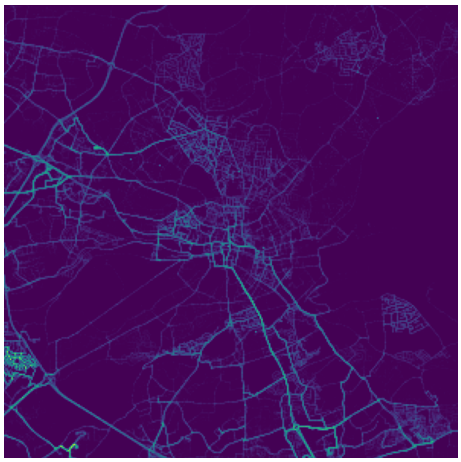


# Analysis of Air Quality in Dublin using IoT and Machine Learning

CRUVELLIER Baptiste - under the supervision of Dr Ali Intizar

january - May 2023

# Contents


**1 Introduction** **7**
- 1.1 Background . . . 7
- 1.2 Project Overview . . . 7
  - 1.2.1 Data Collection . . . 7
  - 1.2.2 Analysis and Mapping . . . 7
  - 1.2.3 Handling missing values . . . 7
- 1.3 Long-Term Goals . . . 7

**2 Visualisation of the air quality data** **8**
- 2.1 Data acquisition and preparation . . . 8
- 2.2 Front-end Design . . . 10
- 2.3 Back-end . . . 10
  - 2.3.1 SQL Solution . . . 11
  - 2.3.2 Spliting File Solution . . . 11

**3 Methodology for Missing Value Prediction** **13**
- 3.1 Data preprocessing for Machine Learning - 3D Matrix . . . 13
  - 3.1.1 Missing Values . . . 13
  - 3.1.2 Size and localisation of the matrix . . . 14
  - 3.1.3 Matrix filling algorithm . . . 16
- 3.2 Interpolation . . . 17
  - 3.2.1 Nearest Neighbour Interpolation . . . 18
  - 3.2.2 Linear interpolation . . . 18
  - 3.2.3 Cubic interpolation . . . 18
- 3.3 K-Nearest Neighbors (KNN) . . . 19
  - 3.3.1 Algorithm principle . . . 19
  - 3.3.2 KNN for missing data imputation . . . 19
  - 3.3.3 KNN - S and KNN - T . . . 20
- 3.4 Linear Regression . . . 20
  - 3.4.1 Description . . . 20
  - 3.4.2 Lasso Regression . . . 20
  - 3.4.3 Ridge Regression . . . 21
- 3.5 Machine learning models . . . 21
  - 3.5.1 Random Forest . . . 21
  - 3.5.2 Gradient Boosting Method (GMB) . . . 22
  - 3.5.3 Artificial Neural Network (ANN) . . . 22

**4 Analysis of the Correlation between Air Quality and Other Factors** **24**
- 4.1 Data Acquirement . . . 24
- 4.2 Correlation . . . 24
- 4.3 Compute correlation . . . 24
- 4.4 Analysis and Interpretation of the results . . . 26
  - 4.4.1 Numerical analysis of the results . . . 26
  - 4.4.2 Conclusion of the correlation . . . 27
- 4.5 adding the weather data to train model . . . 28

**5 Implementation of Prediction Methods** **29**
5.1 KNN KNN S/T - Implementation . . . 29
5.2 Interpolation - Implementation . . . 30
5.2.1 Interpolation T . . . 30
5.2.2 Interpolation S . . . 32
5.2.3 Interpolation S/T . . . 32
5.3 Regression and Machine Learning models - Implementation . . . 33

**6 Evaluation and Selection of the Best Model** **34**
6.1 Method of evaluating models . . . 34
6.2 Comparison of Methods . . . 35
6.2.1 Training model using the 3D matrix . . . 35
6.2.2 Training model with 3D matrix and weather data . . . 37
6.3 Visualization of the result of some models . . . 39
6.3.1 Creation of csv file from 3D matrix . . . 39
6.3.2 Visualization . . . 39
6.4 Selection of the Best Method . . . 42
6.4.1 Without weather data . . . 42
6.4.2 With weather data . . . 42

**7 Ethics** **44**

**8 Conclusion and Future Work** **45**

**9 Follow-up of the project in EE457** **46**

# List of Figures


1 Overview of the HTML interface . . . 10
2 Output splittingFile . . . 12
3 Illustration of missing spatial values . . . 13
4 Illustration of a day where there is almost no values . . . 14
5 Square for data prediction on Google Earth . . . 15
6 3D Matrix . . . 15
7 Comparaison Interpolations . . . 18
8 Illustration of KNN on a 2D map . . . 19
9 Random Forest Majority Voting . . . 21
10 Multi-layer perceptron . . . 22
11 Correlation between weather and air polution (pm2.5) . . . 26
12 Interpolation S - Necessity of boundary conditions . . . 31
13 Interpolation S - Clip Value between min and max . . . 31
14 Interpolation ST . . . 32
15 MSE, MAE and R2 values for different models without weather data . . . 36
16 MSE, MAE and R2 values for different models with weather data . . . 38
17 Adding prediction to the selector . . . 40
18 Visualization of Interpolation-ST Brut . . . 40
19 Visualization of Interpolation-ST with convolution . . . 41
20 Visualization of random forest with convolution . . . 42

**Declaration of Authorship**

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

I have read and understood the DCU Academic Integrity and Plagiarism at
https://www.dcu.ie/system/files/2020-09/1_integrity_and_plagiarism_policy_vpaa_-_v4.pdf
and EE457 referencing guidelines found at
https://loop.dcu.ie/course/view.php?id=58332

Name: *Cruvellier Baptiste* - Date: 05/05/2023

**Abstract**

In this study, we investigated the analysis and prediction of air quality in the Dublin area using data collected by sensors installed on DPD delivery vehicles. A significant correlation between meteorological data and fine particulate matter (pm2.5) concentrations was found. Several models were tested to estimate the missing values of air quality (pm2.5), and the best results were obtained with the KNN model without meteorological data, as well as with the random forest, GBM and ANN models incorporating meteorological data.

The research also created a map-based visualization of air quality in Dublin. Future work may include analyzing a larger volume of data to better understand the influence of seasonal variations on air quality, incorporating weather forecasts to improve model accuracy, and extending the current approach to predicting future air quality values. The ultimate goal is to provide a valuable tool for environmental policy and urban planning decisions.

# 1 Introduction

## 1.1 Background

Air quality is a major public health and urban ecological issue. In Dublin, the capital of Ireland, the situation is of concern due to the constant increase in population and road traffic. This innovative project aims to collect data on air quality in Dublin using DPD delivery vehicles as a means of measurement. Sensors installed on these vehicles record air quality data as they travel around the city each day, making it possible to track changes in air quality in different neighborhoods and at different times of the day.

## 1.2 Project Overview

The project will proceed in several key stages: Data Collection, Analysis and Mapping and finally Handling missing values.

### 1.2.1 Data Collection

The sensors in question are responsible for collecting the following information, which we will need to process:

- Air quality indicators (pm2.5 and pm10)
- GPS coordinates
- Date and time of data collection

### 1.2.2 Analysis and Mapping

By focusing on the air quality indicators (pm2.5 and pm10) and combining this information with GPS coordinates, dates and times, we have the opportunity to create a dynamic picture of the environmental situation in Dublin. Our first step is to develop a map that illustrates this data, providing a clear view of air quality at different times of the day.

### 1.2.3 Handling missing values

However, there will be missing values in our dataset, and this is the heart of our project. We will implement various techniques and models to predict these missing values, exploring the correlation between air quality and other sources of information, such as meteorological data. By estimating these values, we will build a more complete and reliable dataset for analysis, revealing the unknowns and correlations behind air quality variations.

## 1.3 Long-Term Goals

Our quest to understand the factors influencing these variations will lead us to a better understanding of the Dublin environment and to the discovery of valuable tools for environmental policy and urban planning decisions. Ultimately, beyond simply collecting data, this project will allow us to contribute to building a healthier future for Ireland's capital city and its residents.

Here is an access to my GitHub, where you can find all my python/html/php/js codes :
https://github.com/Remenby31/AirQualityDublin

# 2 Visualisation of the air quality data

In this section, we will discuss air quality data visualization, focusing on data acquisition and preparation, front-end design, and back-end solutions.

## 2.1 Data acquisition and preparation

We will describe data acquisition and preparation, highlighting the sources and procedures used to obtain and process air quality data.

Table 1: Mobile data structure in CSV file

| N | timestamp | sensor ID | pm2.5(ug/m3) | pm10(ug/m3) | latitude | longitude |
|---|---|---|---|---|---|---|
| 0 | 1633046403 | 869170034050501 | 1.33 | 1.33 | 53.320283 | -6.363592 |
| 1 | 1633046405 | 869170034040783 | 2.46 | 3.69 | 53.388120 | -6.171005 |
| 2 | 1633046407 | 869170034084252 | 1.42 | 1.42 | 53.320965 | -6.361845 |
| 3 | 1633046417 | 869170034040783 | 2.69 | 6.15 | 53.388120 | -6.171005 |
| 4 | 1633046418 | 869170034084252 | 1.75 | 1.75 | 53.320965 | -6.361845 |
| ... | ... | ... | ... | ... | ... | ... |

Table 2: Fixed data structure in CSV file

| N | timestamp | sensor ID | pm2.5(ug/m3) | pm10(ug/m3) |
|---|---|---|---|---|
| 0 | 1648771251 | 869170034045121 | 5.40 | 7.18 |
| 1 | 1648771253 | 869170033931073 | 4.50 | 6.57 |
| 2 | 1648771256 | 869170034081928 | 7.05 | 9.35 |
| 3 | 1648771261 | 869170033930547 | 4.95 | 5.40 |
| 4 | 1648771271 | 869170034081928 | 6.90 | 8.09 |
| ... | ... | ... | ... | ... |

Table 3: Structure of station coordinates

| N | sensor ID | latitude | longitude |
|---|---|---|---|
| 0 | 869170034086034 | 53.390659 | -6.201700 |
| 1 | 869170034045121 | 53.391046 | -6.168369 |
| 2 | 869170034052879 | 53.369354 | -6.258590 |
| 3 | 869170034063280 | 53.387039 | -6.258155 |
| 4 | 869170033897597 | 53.321163 | -6.361607 |
| ... | ... | ... | ... |

In order to streamline the data processing and analysis, I developed a Python script to merge the mobile and fixed sensor data into a single CSV file. This consolidated file simplifies the data handling process, making it easier to perform analysis and visualization tasks.

```
#For each folder
for folder in folders:
    # If it's a mobile folder
    if os.path.isdir(folder) and folder.split('_')[1] == 'mobiles':
        # Folder names
        mixedFolder = folder.replace('mobiles', 'mixed')
        mobileFolder = folder
        fixedFolder = folder.replace('mobiles', 'fixed').replace('30', '31')
            # Create the mixed folder, if it doesn't existg
        if not os.path.exists(mixedFolder):
            os.makedirs(mixedFolder)

        # Get the list of files in the mobile folder
        mobileFiles = os.listdir(mobileFolder)

        # For each mobile file
        for mobileFile in mobileFiles:

            # ==================== Get the mobile file ====================
            try:
                mobileDf = pd.read_csv(mobileFolder + '/' + mobileFile, sep=';', header=None)
            except:
                print("Mobile file " + mobileFile + " not found")
                mobileDf = None

            # ==================== Get the fixed file ====================
            fixedFile = mobileFile.replace('mobile', 'fixed')
            try:
                fixedDf = pd.read_csv(fixedFolder + '/' + fixedFile, sep=';', header=None)
            except:
                print("Fixed file " + fixedFile + " not found")
                fixedDf = None

            # ==================== Merge the two files ====================
            if mobileDf is not None and fixedDf is not None:
                mixedDf = pd.concat([mobileDf, fixedDf])
            elif mobileDf is not None:
                mixedDf = mobileDf
            elif fixedDf is not None:
                mixedDf = fixedDf

            # ==================== Sort the merged file ====================
            mixedDf = mixedDf.sort_values(by=mixedDf.columns[0])

            # ==================== Save the merged file ====================
            mixedDf.to_csv(mixedFolder + '/' + mobileFile, index=False, header=False, sep=';')
```

The Python script performs the following steps to merge the data:

1. First, the script reads both mobile and fixed sensor CSV files using a library like pandas or csv.
2. The script then combines the data from both sources into a single DataFrame or equivalent

data structure. This involves appending the fixed sensor data to the mobile sensor data while ensuring that the columns align correctly. The latitude and longitude columns are added to the fixed sensor data by looking up the sensor ID in the station coordinates table.

3. Once the data is combined, the script sorts the resulting DataFrame by timestamp to ensure chronological order, making subsequent time-based analysis easier.

4. Finally, the script exports the merged data into a new CSV file, which can be used for further analysis, visualization, and prediction tasks.

By consolidating the data from mobile and fixed sensors, we create a more comprehensive dataset that facilitates more accurate analysis of air quality in the Dublin area.

## 2.2 Front-end Design

In this part, we will focus is to design a way to display air quality data (pm2.5 and pm10) on a map. After extensive research, I found that the easiest open-source way to display a map was with Leaflet. Then, using an extension called Leaflet.heat available on Github, a layer can be displayed on top of the map to represent air quality values around Dublin.

I found it logical to display the values on the map on a day-to-day basis, as it was a good compromise between reality and the limited amount of data we had, which was too limited for hourly data, for example.

As for the interface, I added:

- a **slider**, to select the day
- a **selector**, to select the values of pm2.5 or pm10
- a **play button** to automatically scroll through the days.

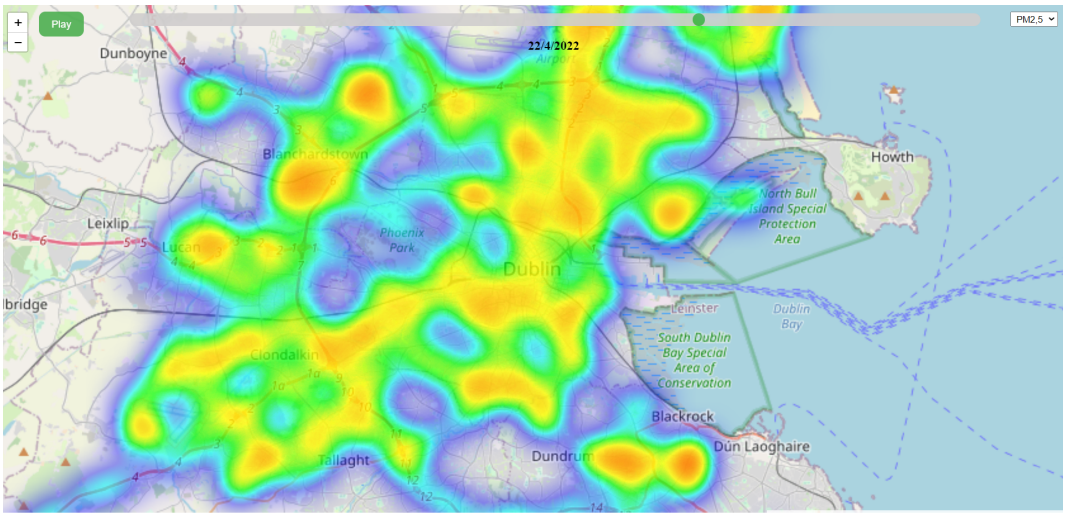



Figure 1: Overview of the HTML interface

## 2.3 Back-end

In this sub-section, back-end solutions will be discussed, looking at SQL database-based approaches and file separation.

Air quality data, collected for fine particles PM2.5 and PM10, are provided in the form of a CSV file. Given that this file is of substantial size, it is necessary to find an efficient way to store all of this data. The primary goal is to enable quick access to all available values located between two timestamps.

To achieve this goal, it is important to adopt an optimized and high-performance storage method, such as a database or an in-memory storage system.

### 2.3.1 SQL Solution

First, I chose to design an **SQL database** to manage all the information needed for our project. This database, by facilitating the selection of relevant data between two precise timestamps, will greatly simplify the analysis process. This will be made possible thanks to queries issued directly from the user interface, thus offering an efficient method to interact with the data.

For the realization and implementation of this database, I decided to use the **XAMPP** software. This choice was motivated by several reasons, notably the fact that XAMPP is open source, thus facilitating its adoption and customization. Moreover, this software offers an easy handling of **MySQL**, a significant asset to manage the database. In addition, XAMPP also integrates various valuable tools for web development, such as **Apache** and **PHP**, making this software even more suitable for our needs.

Regarding the SQL queries I set up, I made sure to specify the dates between which I wanted to retrieve the data. This allows me to extract only the relevant information corresponding to a 24-hour period, i.e. a full day. The goal of this approach is to isolate the data needed for daily air quality analysis, thus improving the efficiency of data processing operations. As a result, our ability to analyze and understand changes in air quality is enhanced, allowing for faster and more targeted corrective action.

```sql
SELECT longitude, latitude, pm25, pm10
FROM Donnees2022
WHERE timestamp > start_timestamp AND timestamp < (start_timestamp + 86400)
```

This solution is significantly more efficient than sequentially scanning the entire CSV file to select the values to display. However, it was still too slow for day-to-day scrolling on the user interface (approximately 2 to 3 seconds per query). To enhance the performance further, additional optimization techniques has to be explored.

### 2.3.2 Spliting File Solution

This brings us to this second solution for processing the data. Knowing that I wanted to display all the data for a specific day each time, I developed a Python script to split the large CSV file into smaller files, each corresponding to a single day. These smaller files are named after the dates of the respective days they represent.

This approach enables more efficient data retrieval, as it reduces the amount of data to be searched through when looking for information on a particular day. By dividing the data into smaller, more manageable chunks, the performance of the system is improved, resulting in faster response times when querying for daily air quality information. Moreover, this method simplifies data organization.

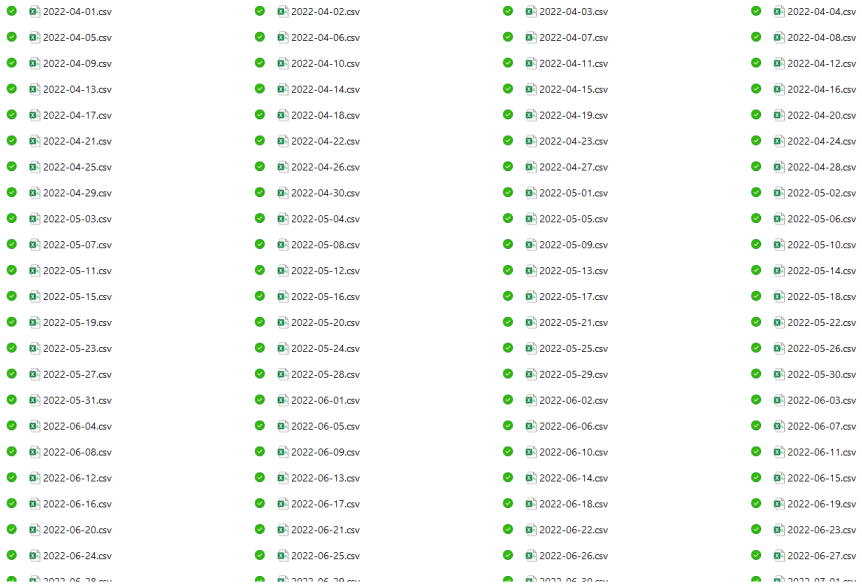



Figure 2: Output splittingFile

Then, I had to modify my php code so that instead of sending SQL requests, it simply fetches the right folder and the right file.

```php
<?php
    $timestamp = $_GET['timestamp'];
    $data_type = $_GET['data_type'];

    $dateString = date('Y-m-d', $timestamp);
    $filePath = __DIR__ . '/DataDublin/' . $data_type . '/' . $dateString . '.csv';

    if (file_exists($filePath)) {
        $fileContents = file_get_contents($filePath);
        $fileContents = str_replace("\n", "<br>", $fileContents);
        header('Content-Type: text/html');
        echo $fileContents;
      } else {
        echo "None";
      }
?>
```

Thus, knowing the name of the file, which corresponds to the specific date, it becomes much easier and faster to access and display the data for a particular day. The user interface can simply open the relevant file and retrieve the necessary information without having to search through a larger dataset. This significantly reduces query times and enhances the overall user experience when interacting with the air quality data.

# 3 Methodology for Missing Value Prediction

The methodology for predicting missing values is explored in this section, looking at data preprocessing techniques, interpolation methods, neighborhood approaches, and regression and machine learning models.

## 3.1 Data preprocessing for Machine Learning - 3D Matrix

Data preprocessing for machine learning models will be detailed, including handling missing values and creating 3D matrices for analysis.

### 3.1.1 Missing Values

Once the HTML website was created, it became much easier to examine the amount of data collected, particularly in terms of missing values. I encountered two types of problems related to missing data:

1. **Missing spatial values**

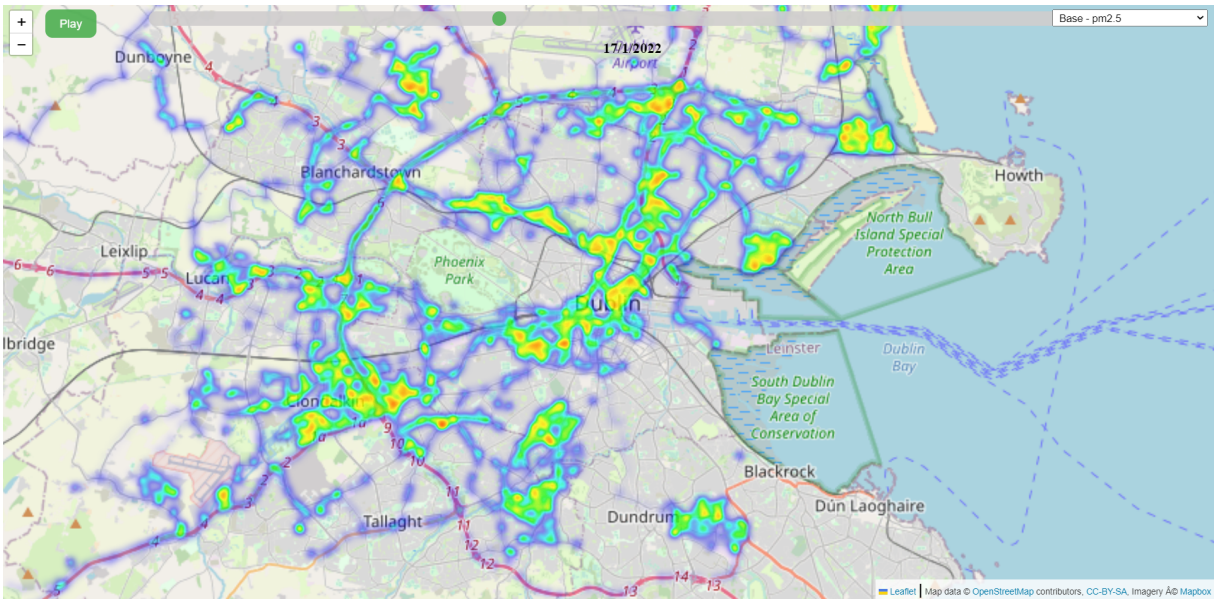



Figure 3: Illustration of missing spatial values

It turned out that the collected data were more geographically extensive than expected. Indeed, although the majority of the data come from Dublin, some extend across the whole of Ireland, or even as far as the United Kingdom. Even when focusing solely on Dublin, the best days only show values on the roads. Consequently, it is impossible to obtain accurate information about certain areas from the raw data.

2. **Missing daily temporal values**

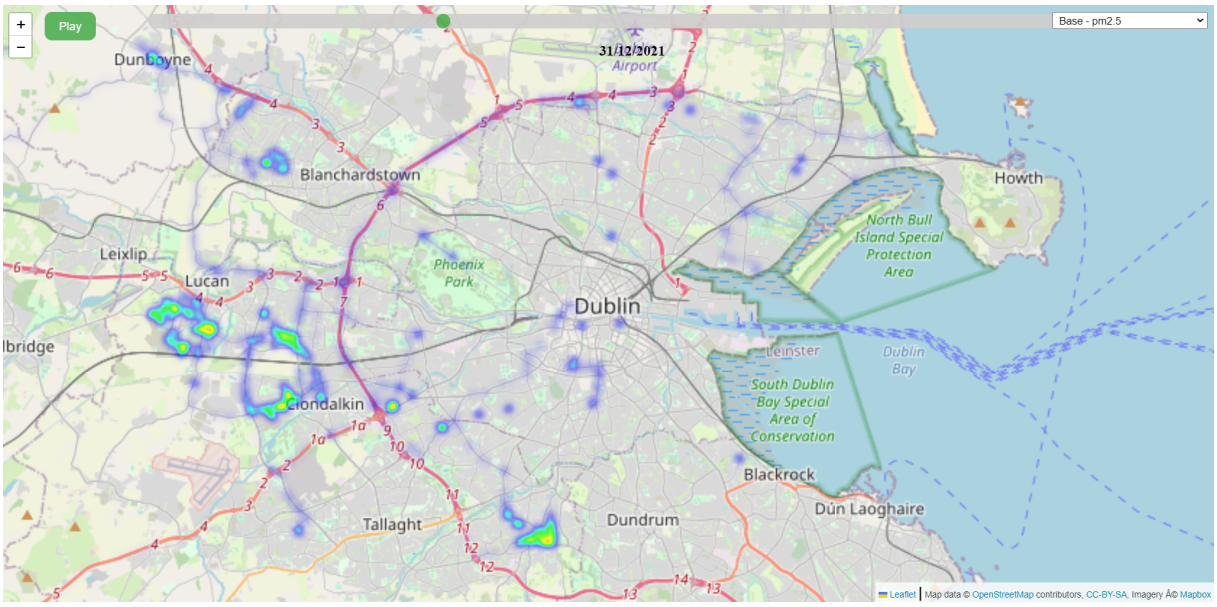



Figure 4: Illustration of a day where there is almost no values

The figue is a day where there is very little values of air quality. It shows that the data are not distributed evenly throughout the week. For example, Sunday is a day when there are significantly fewer values. Moreover, some days have very little data, with most of them corresponding to public holidays.

3. **Missing temporal values over several weeks**

   The datasets are provided in two separate files and do not overlap in time. As a result, there is a two-month period during the summer when no data was recorded.

These problems with missing values, both spatial and temporal, highlight the importance of carefully processing and analyzing data. It is essential to identify gaps and understand their implications to ensure reliable and relevant analysis results. The visualizations presented above can help to better understand the nature of the data and the challenges related to their processing.

### 3.1.2 Size and localisation of the matrix

1. **Geographical positioning of the matrix**

   In order to determine the matrix surrounding Dublin, we first established two geographical points around the city to form an encompassing square. To do this, we used Google Earth to accurately position these points while seeking to strike a balance between the area we wanted to cover and the amount of data at our disposal.


Figure 5: Square for data prediction on Google Earth

2. **Size of the matrix**

   The next step is to define the dimensions of our matrix. The goal is to find the best compromise between the desired resolution and a sufficient amount of data to minimize missing data.

   After several tests, we opted for a matrix size of 350, which means we divided the city of Dublin into 350x350 cells.

   By integrating the temporal dimension into our matrix, we obtain a 3D matrix in the form of (day, longitude, latitude).

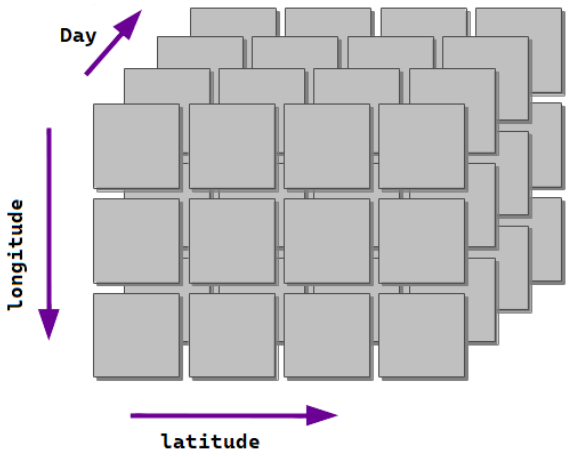



Figure 6: 3D Matrix

### 3.1.3 Matrix filling algorithm

The algorithm aims to create matrices of air pollution measurement data for a given geographic region over a specified period of time. For each day within the time period, the algorithm uses CSV files containing air pollution measurement data for each point within the geographic region to create corresponding matrices of PM2.5 and PM10. For each point within the matrix, the algorithm calculates the corresponding geographic coordinates and retrieves the air pollution measurement data from the CSV file. If data is available, the algorithm computes the average of the data and adds it to the matrix for the corresponding day. If no data is available, the corresponding value within the matrix is set to zero.

```
#For each point in the matrix
for i in range(size_matrix):

# Get the latitude range of the point
latitude_min = latitude_basDroit + i * dlatitude - 5 * dlatitude
latitude_max = latitude_basDroit + (i+1) * dlatitude + 5 * dlatitude

for j in range(size_matrix):

    # Get the longitude range of the point
    longitude_min = longitude_hautGauche + j * dlongitude - 5 * dlongitude
    longitude_max = longitude_hautGauche + (j+1) * dlongitude + 5 * dlongitude

    # Get the points in the daily file that are within the latitude and longitude range of the point
    dayDfPointLongitude = dayDf[(dayDf[5] >= longitude_min)
        & (dayDf[5] <= longitude_max)]
    dayDfPoint = dayDfPointLongitude[(dayDfPointLongitude[4] >= latitude_min)
        & (dayDfPointLongitude[4] <= latitude_max)]

    # If there are points in the daily file, compute the mean and add it to the daily matrix
    if dayDfPoint.shape[0] > 0:
        dayMatrix_pm25[i,j] = dayDfPoint[3].mean()
        dayMatrix_pm10[i,j] = dayDfPoint[4].mean()
    # Otherwise, set the value to 0
    else:
        dayMatrix_pm25[i,j] = 0
        dayMatrix_pm10[i,j] = 0
```

## 3.2 Interpolation

Various interpolation methods will be explored, including linear and cubic interpolation, for estimating missing values in the data.

Interpolation is a statistical technique for estimating missing values from existing values in a data set. In our case, we are interested in interpolation on a 3D matrix to predict missing air quality values. The 3D matrix represents air quality as a function of GPS coordinates (latitude and longitude) and time. For this purpose, we will quickly review three types of interpolation: Nearest Neighbor, Linear, and Cubic.

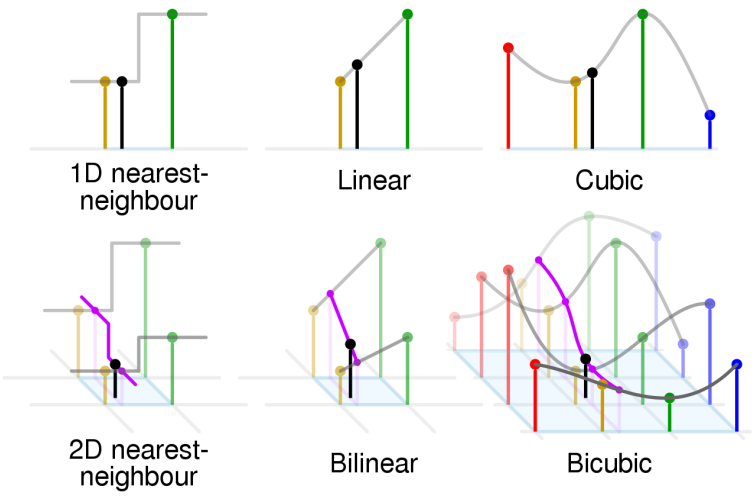



Figure 7: Comparaison Interpolations

### 3.2.1 Nearest Neighbour Interpolation

This interpolation method is the simplest and consists of assigning the value of the nearest sample in space to the missing value. Thus, the air quality for a given point will be estimated using the air quality value of the closest point in the 3D matrix. This approach is quick and easy to implement, but can result in less accurate estimates, especially when the data are highly scattered or have local variations.

### 3.2.2 Linear interpolation

Linear interpolation is a more sophisticated method that estimates the missing value by drawing a straight line between the two closest points in space. The air quality for a given point will be estimated by linearly interpolating between the air quality values of the two closest points in the 3D matrix. This approach gives smoother and more accurate results than the Nearest Neighbor method, but is more sensitive to local variations in the data.

According to the article from Hawthorne and Elliott (2004)[6], emphasizes the importance of solving incomplete data problems, as they can lead to different results than those obtained from a complete data set. The study specifically compares the performance of linear interpolation with mean value substitution for replacing missing values in an environmental data set. Thus, by using linear interpolation, one can expect to obtain more accurate estimates and minimize potential biases due to systematic differences between observed and unobserved data.

### 3.2.3 Cubic interpolation

Cubic interpolation is a more advanced method that uses multiple degree polynomial functions to estimate missing values. Cubics are smooth curves that pass through each data point and provide a better approximation of local variations in the data. The air quality for a given point will be estimated using Cubic interpolation based on the air quality values of the surrounding points in the 3D matrix. This method is more complex and computationally intensive than previous methods, but it can provide more accurate estimates that are better suited to local variations in the data.

In summary, Nearest Neighbor Interpolation is the simplest and fastest method, but it can provide less accurate estimates. Linear interpolation is a compromise between simplicity and accuracy, while Cubic/Cubic interpolation is the most advanced method and can provide more accurate estimates

tailored to local variations in the data. The choice of interpolation method will depend on the accuracy requirements and computational constraints for each specific application.

## 3.3 K-Nearest Neighbors (KNN)

The K-nearest neighbor (KNN) algorithm will be presented, explaining its principle, its application for imputing missing data, and the KNN-S and KNN-T variations.

### 3.3.1 Algorithm principle

The K-Nearest Neighbors (KNN) algorithm is a supervised learning method used for regression and classification. In the context of predicting air quality, it is a regression problem, as it involves predicting a continuous numerical value.

The distance between data points in the matrix must be determined using Euclidean distance. This will allow us to find the K nearest points to each data point with missing values.

Once the K nearest points are found, their values can be used to predict the missing value using a weighted average method. The prediction of the missing value is the weighted average of the values of the K nearest neighbors, weighted by their distance from the data point with the missing value.

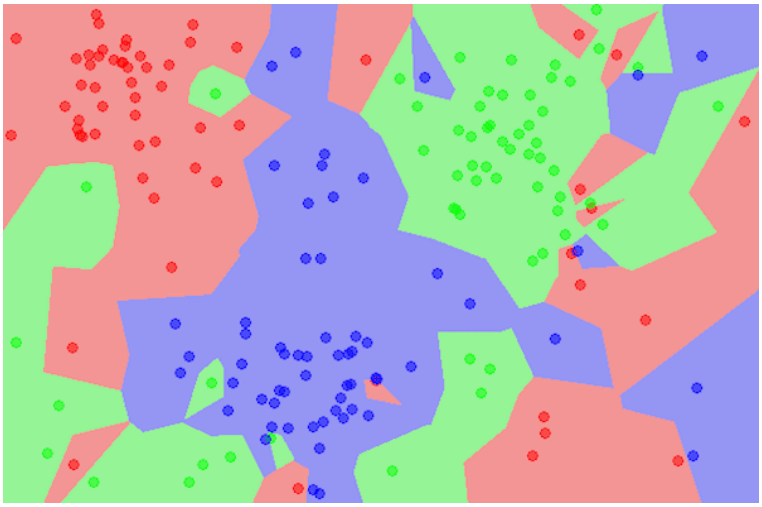


Figure 8: Illustration of KNN on a 2D map

### 3.3.2 KNN for missing data imputation

In their article [1], Ahmat Zainuri and Nuryazmin suggest that KNN methods are one of the most effective approaches for imputing missing data in air quality datasets. The K-nearest neighbors (KNN) algorithm is an effective method for predicting missing values using known values from similar data.

The KNN algorithm is a non-parametric method that does not require assumptions about the data distribution. Moreover, it can be applied to large datasets, such as 3D matrices representing air quality in different regions and at different times. One advantage of the KNN method is that it is simple to implement and understand, and can be effective even with a small number of neighbors (K) for prediction. This method can also be applied using Python.

In summary, the use of the KNN method for imputing missing data in air quality datasets can be an effective and simple approach, particularly if other imputation methods are not performing well.

### 3.3.3 KNN - S and KNN - T

KNN-S and KNN-T are variants of the KNN method that are specifically designed to handle spatio-temporal data. Both methods consider the temporal dimension in addition to the spatial dimension, but they give more or less weight to the spatial data compared to the temporal data.

1. **KNN-S** (K nearest neighbors with spatial weighting) gives more weight to the spatial data that are collected at similar times in the temporal dimension. This reduces the potential temporal bias that can affect predictions based solely on spatial data.

2. **KNN-T** (K nearest neighbors with temporal weighting) gives more weight to the temporal data that are collected at similar locations in the spatial dimension. This reduces the potential spatial bias that can affect predictions based solely on temporal data.

In conclusion, KNN-S and KNN-T are useful variants of the KNN algorithm for predicting missing values in spatio-temporal datasets. They take into account both the spatial and temporal dimensions and use weighting functions to give more or less weight to spatial or temporal data depending on their relevance.

## 3.4 Linear Regression

We will examine linear regression and its variants, Lasso and Ridge regression, to model the relationships between variables.

### 3.4.1 Description

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon \tag{1}$$

Linear regression is a statistical method commonly used to model the relationship between a continuous dependent variable (Y) and one or more continuous or categorical independent variables (X). The objective of linear regression is to find the best regression line that represents the linear relationship between the variables. This regression line is used to predict the value of Y for a given value of X. The regression line is a good choice for predicting air quality because it is simple, effective, and has good cross-validation validity[10]. The study's results show high coefficients of determination (R2), indicating a satisfactory model performance.

Regression is effective in predicting missing air quality values. According to the paper by Mahanta, Soubhik and Ramakrishnudu, T. and Jha, Rajat Raj and Tailor, Niraj [4], regression-based prediction models are used to predict the air quality index (AQI) based on pollution and meteorological data in New Delhi, India. The results of the regression analysis reveal which meteorological factors most influence the AQI and the usefulness of predictive models for air quality forecasting.

Lasso regression and Ridge regression are two variants of linear regression that were developed to improve the ability of linear regression to generalize to new data sets. Both techniques add a penalty for regression coefficients that are too large, in order to reduce the overfitting of linear regression.

### 3.4.2 Lasso Regression

Lasso regression uses an absolute value penalty for the regression coefficients, resulting in zero coefficients for variables that do not contribute significantly to the prediction of the dependent variable. This technique is particularly useful for variable selection in data sets with many independent variables.

Lasso Regression is presented as an effective solution for predicting missing air quality values in the paper written by Sethi, Jasleen Kaur and Mittal, Mamta [8]. The study proposes a Lasso-based feature selection method called CbAL Regression. The results show that carbon monoxide, sulfur dioxide, nitrogen dioxide, and ozone are the most important factors for predicting air quality. The CbAL Regression method outperforms the Lasso regression in terms of accuracy, with an average

classification accuracy of 78%. These findings were used to help improve air quality in and around Delhi.

### 3.4.3 Ridge Regression

Ridge regression uses a quadratic value penalty for the regression coefficients, which reduces the impact of larger coefficients on the prediction of the dependent variable. This technique is particularly useful for data sets with high correlations between the independent variables.

## 3.5 Machine learning models

Several machine learning models, such as random forest, gradient boosting and artificial neural networks, will be presented and discussed.

### 3.5.1 Random Forest

RandomForest is a supervised learning method for classification, regression and other data analysis tasks. It combines several decision trees (set of decision trees) to improve the accuracy and stability of prediction. The decision trees are constructed using a training data set and are used to predict the output value for new input data. In the random forest approach, multiple decision trees are created using different subsets of the training data set and using randomly selected variables at each decision node. Predictions are then made by averaging the results of each decision tree. This approach improves accuracy and reduces overfitting compared to a single decision tree.

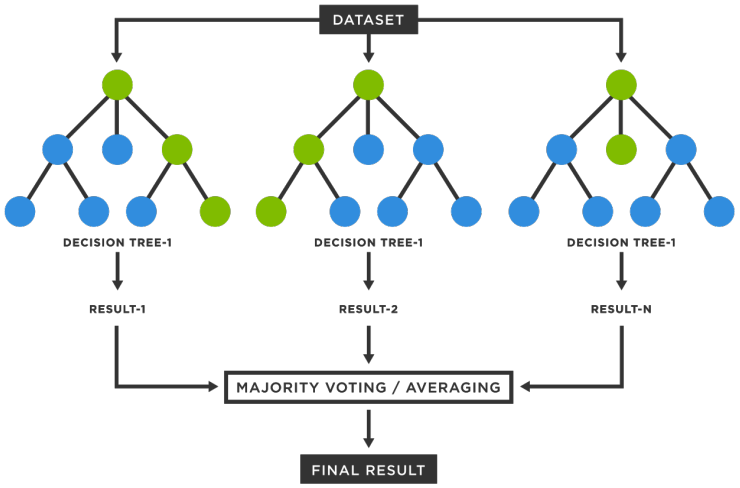



Figure 9: Random Forest Majority Voting

The paper by Ahmad R Alsaber et al. [2] describes the use of a random forest method (RandomForest) to predict missing air quality values (PM2.5 and PM10) in a Kuwait air quality monitoring dataset. They used the missForest iterative multiple imputation method, which is related to the random forest approach, to replace missing values. Climatological data were also included in the analysis to improve the estimation. The results showed that the MAR technique had the lowest RMSE and MAE, and missForest had the lowest imputation error among the other imputation methods tested.

### 3.5.2 Gradient Boosting Method (GMB)

The Gradient Boosting Method is a supervised machine learning technique that uses a set of simple base models to build a more complex prediction model by minimizing the residual error at each step. GBM is often used for classification and regression problems and can be very powerful when the data is complex. It is important to regularize the model by adjusting hyperparameters such as tree depth and learning rate to avoid overfitting.

According to some article such as [9], GBM is a good choice for studying the evolution of air pollutants during events because it is efficient, accurate, and adaptable for modeling complex relationships between variables.

### 3.5.3 Artificial Neural Network (ANN)

An artificial neural network is a mathematical model of machine learning inspired by the functioning of the human brain. It is composed of layers of interconnected neurons that process inputs, generate outputs and adjust the weights of the connections between neurons to improve prediction accuracy. ANNs are widely used in classification, regression, image recognition and time series prediction.

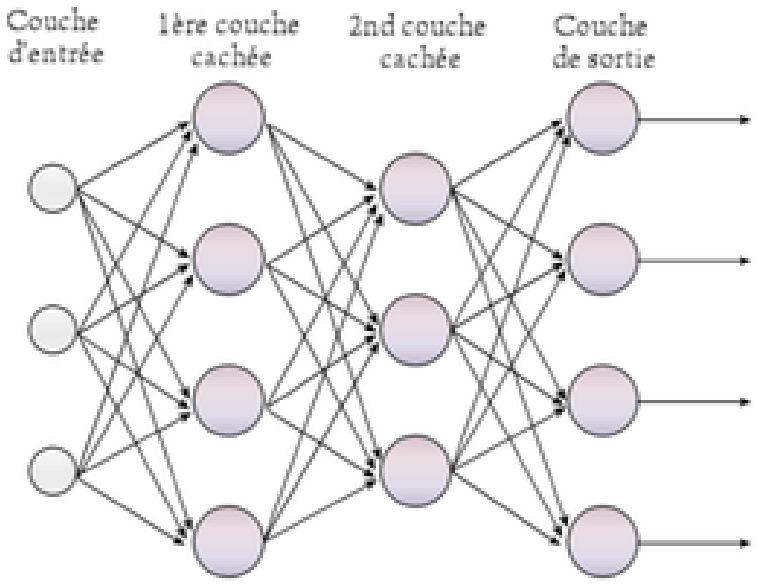



Figure 10: Multi-layer perceptron

According to Heikki Junninen and Harri Niska, [3], neural networks, such as the multilayer perceptron, performed slightly better than univariate methods for imputing air quality data in simulated missing data schemes. However, other factors such as model complexity and computational time must also be considered before deciding to use neural networks for air quality data imputation.

The use of neural networks (NN) to handle missing values has significant advantages. In a study by[7], Rumaling, Muhammad Izzuddin and Chee, Fuei pien and Dayou, Jedol and Chang, Jackson and Kong, Steven and Sentian, J., NNM and EM methods are compared to impute missing air quality data. NN can take into account complex and non-linear relationships, which allows for more accurate estimates. The results show that NNM performs better than EM at some monitoring stations. In addition, the accuracy evaluation shows that NNM is more accurate in most cases. By using neural

networks, it is possible to capture complex patterns and provide accurate estimates to improve data analysis.

# 4 Analysis of the Correlation between Air Quality and Other Factors

Here we analyze the correlation between air quality and other factors, discussing data acquisition, correlation methods, analysis and interpretation of results, and the use of these data to train models.

## 4.1 Data Acquirement

To obtain the meteorological data needed for my analysis, I consulted the website *visualcrossing.com*. This site provides free access to a wide range of meteorological information for Dublin, covering the time period corresponding to my air quality data set. Once this meteorological data was retrieved, I proceeded to extract the daily average air quality from my matrix. This step allowed me to obtain a representative value of air pollution for each day studied, thus providing a solid basis for the analysis of the relationship between weather and air quality.

Finally, I set out to examine the relationships between the various meteorological variables and the concentration of fine particles PM2.5. To do this, I calculated the correlation coefficient between each column of the CSV file containing the meteorological data and the PM2.5 concentration values. This approach allowed me to quantify the impact of meteorological factors on air quality and to identify the most significant variables for understanding and predicting air pollution in Dublin.

## 4.2 Correlation

Correlation is a statistical measure that assesses the linear relationship between two variables. In other words, correlation measures how closely the values of two variables are related and how they vary together. The correlation equation is usually denoted by the letter "r" and represents the correlation coefficient. This coefficient can range from -1 to +1, where -1 indicates a perfect negative correlation, +1 a perfect positive correlation, and 0 indicates that there is no linear correlation between the variables.

The correlation equation can be expressed mathematically as follows:

$$r_{xy} = \frac{\operatorname{cov}(x,y)}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2}$$

In other words, correlation is calculated by dividing the covariance of variables by the product of their standard deviations. The covariance measures how much two variables vary together, while the standard deviation measures the dispersion of a variable's values around its mean. Using the correlation equation, you can calculate the correlation coefficient between two variables to assess the strength and direction of their linear relationship.

## 4.3 Compute correlation

This subpart will discuss the calculation of correlation coefficients and associated techniques.

In this code, I worked with weather and pollution data to explore the relationship between different variables.

```
# Load the 3D matrix from "matrix3D_pm25.npy"
matrix = np.load("matrix3D_pm25.npy")

# Load the CSV file "Dublin 2021-10-01 to 2022-07-31.csv"
df = pd.read_csv("Dublin 2021-10-01 to 2022-07-31.csv", delimiter=";")

def determine_correlation(matrix, X):
    # Calculate the average temperature per day
    average_temperature_per_day = np.mean(matrix, axis=(1, 2))

    # Calculate the correlation coefficient between average_temperature_per_day and X
    correlation_coefficient = np.corrcoef(average_temperature_per_day, X)[0, 1]

    return correlation_coefficient

L = []

# Iterate over columns in the DataFrame
for i in range(df.shape[1]):
    try:
        # Calculate the correlation score for each column in the DataFrame
        score = determine_correlation(matrix, np.array(df[df.columns[i]].values))

        # Append the score and column name to the list L if the score is not NaN
        if not math.isnan(score):
            L.append([score, df.columns[i]])
    except:
        pass

# Sort the list L in descending order based on the correlation scores
L.sort(reverse=True)

# Display the correlation scores as a bar graph
scores, labels = zip(*L)
y_pos = np.arange(len(labels))
plt.bar(y_pos, scores, align='center', alpha=0.5)
plt.xticks(y_pos, labels, rotation='vertical')
plt.ylabel("Correlation scores")
plt.title("Correlation between weather and pollution data (pm2.5)")

plt.tight_layout()
plt.show()
```

The data was stored in a 3D matrix, which probably represents meteorological and pollution measurements over a period of time. This matrix contains information that I want to correlate with other variables. To analyze the data, I loaded a CSV file that probably contains additional information related to weather and pollution. This file may include different columns representing different variables such as temperature, humidity, wind speed or pollution levels. To understand the correlation between the variables in the matrix and the columns in the CSV file, I implemented a correlation calculation function. This function calculates the correlation coefficient, which measures

the strength and direction of the linear relationship between two variables. I created an empty list to store the correlation scores and corresponding column names. By iterating over each column in the CSV file, I calculated the correlation score between the average temperature per day (extracted from the 3D matrix) and each column in the file. This allowed me to assess how well each variable in the CSV file correlates with the average temperature. To identify the columns with the highest correlations, I sorted the correlation scores in descending order. This sorting helped me understand which variables have the strongest relationship with the mean temperature.

Finally, I visualized the correlation scores using a bar graph. This graph displays the correlation scores as bars, with the column names represented on the x-axis and the correlation scores on the y-axis. This visualization makes it easier to interpret and compare the correlation strengths between the different variables.

Overall, this code allowed me to explore and analyze the relationship between meteorological and pollution variables by calculating correlation scores and visualizing the results, as shown bellow

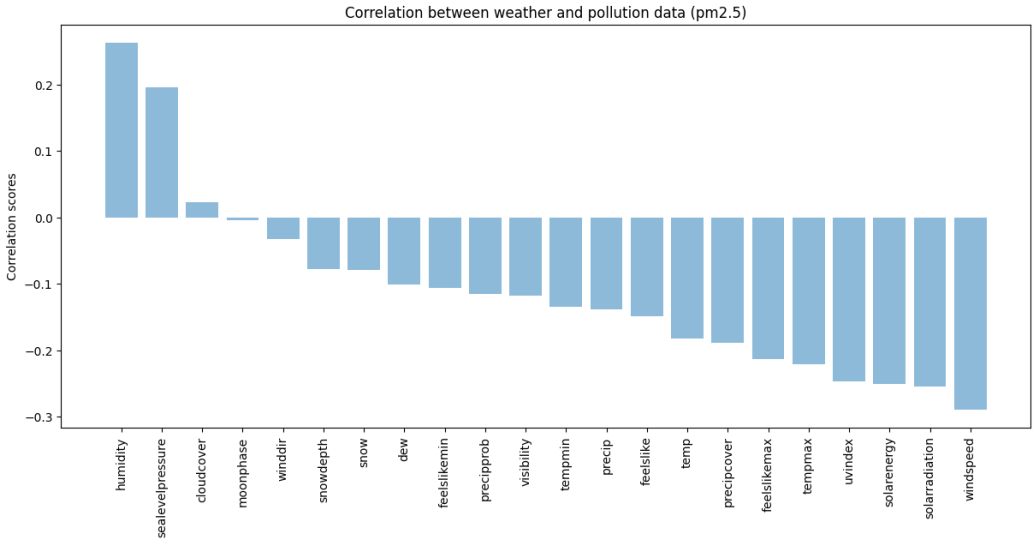



Figure 11: Correlation between weather and air polution (pm2.5)

## 4.4 Analysis and Interpretation of the results

### 4.4.1 Numerical analysis of the results

The analysis and interpretation of the correlation results will be discussed, focusing on the numerical analysis and conclusions drawn from the results.

I compute a code to evaluate the correlation between air pollution (pm 2.5) and different meteorological data. Here is an analysis of the results obtained:

- **Temperatures** (tempmax, tempmin, and temp): The correlation coefficients are negative for maximum, minimum, and mean temperatures (-0.22, -0.13, and -0.18, respectively). This means that there is an inverse relationship between air pollution and these temperatures: as temperature increases, air pollution tends to decrease.

- **Felt temperatures** (feelslikemax, feelslikemin, feelslike): The correlation coefficients are also negative for maximum, minimum, and average felt temperatures (-0.21, -0.11, and -0.15, respectively), indicating a similar relationship to actual temperatures.
- **Dew point** (dew): The correlation coefficient is slightly negative (-0.10), suggesting a weak inverse relationship between air pollution and dew point.
- **Humidity**: The correlation coefficient is positive (0.26), implying that air pollution is directly related to humidity. In other words, when humidity increases, air pollution also tends to increase.
- **Precipitation** (precip, precipprob, precipcover): The correlation coefficients are negative for precipitation, precipitation probability, and precipitation cloud cover (-0.14, -0.12, and -0.19 respectively). This indicates an inverse relationship between air pollution and these factors: when precipitation increases, air pollution tends to decrease.
- **Snow** (snow, snowdepth): The correlation coefficients are weakly negative for snow and snow depth (-0.08 for both). This suggests a weak inverse relationship between air pollution and snow.
- **Wind** (windspeed, winddir): The correlation coefficient is negative for wind speed (-0.29), indicating that air pollution decreases as wind speed increases. In contrast, the correlation coefficient for wind direction is small and negative (-0.03), suggesting a very weak relationship with air pollution.
- **Sea level pressure** (sealevelpressure): The correlation coefficient is positive (0.20), indicating a direct relationship between air pollution and sea level pressure. As sea level pressure increases, air pollution tends to increase as well.
- **Cloud cover**: The correlation coefficient is small and positive (0.02), suggesting a very weak relationship between air pollution and cloud cover.
- **Visibility**: The correlation coefficient is negative (-0.12), indicating an inverse relationship between air pollution and visibility. As visibility increases, air pollution tends to decrease.
- **Solar radiation** and solar energy: The correlation coefficients are negative for solar radiation and solar energy (-0.25 for both), suggesting an inverse relationship between air pollution and these factors. As solar radiation and solar energy increase, air pollution tends to decrease.
- **UV Index** (uvindex): The correlation coefficient is negative (-0.25), indicating an inverse relationship between air pollution and UV index. When the UV index increases, air pollution tends to decrease.
- **Moon phase** (moonphase): The correlation coefficient is very low and negative (-0.004), suggesting an almost insignificant relationship between air pollution and moon phase.

Concerning columns such as sunrise and sunset or meteorological conditions, errors occurred during the analysis, as these were not numerical data, which prevented me from correlating these variables with air pollution.

### 4.4.2 Conclusion of the correlation

In conclusion, our analysis of correlations between air pollution (PM 2.5) and various meteorological data reveals complex relationships between these variables. Several meteorological factors show significant correlations with air pollution, with potential implications for understanding and managing air quality.

Of the factors studied, **wind speed** shows the strongest and inverse correlation with air pollution. This suggests that stronger winds may help disperse air pollutants, thereby improving air quality. **Humidity** and **sea level pressure** show positive correlations with air pollution, indicating that wetter conditions and higher atmospheric pressure can promote the accumulation of pollutants.

**Solar radiation**, **solar energy** and UV index show notable inverse correlations with air pollution. This may be due to the effect of photodissociation, which breaks down some pollutants under the influence of sunlight, as well as the formation of ground-level ozone under high UV exposure.

**Temperature** and **precipitation** also have inverse correlations with air pollution, although less pronounced. Higher temperatures may facilitate the dispersion of pollutants, while precipitation may remove airborne particles.

The other factors studied, such as wind direction, cloud cover, dew point, visibility, snow and moon phase, show weaker relationships with air pollution. Nevertheless, these factors could be relevant in specific local or seasonal contexts.

The results of my study are consistent with the air quality correlating factors identified in the article from Ng, Kar Yong and Awang, Norhashidah [5]. Variables such as humidity, temperature, wind speed a significant role in variations in PM10 concentrations. In addition, the MLR2 model, which includes the lagged variables and PM10 past concentrations, shows similar performance to the RTSE model in terms of forecast accuracy. These results reinforce the consistency between the two studies.

It is important to note that the correlations identified do not necessarily demonstrate causal relationships between meteorological factors and air pollution. Further research, including longitudinal studies and modeling, is needed to further our understanding of the underlying mechanisms and to assess the effectiveness of potential interventions to improve air quality.

## 4.5 adding the weather data to train model

We thus observe a logical correlation between air quality and various environmental factors. Factors with a positive impact contribute to poorer air quality, such as humidity. Conversely, factors with a negative impact are associated with better air quality in Dublin, such as wind speed or solar intensity.These relationships highlight the influence of different meteorological parameters on air pollution. For example, high wind speed can disperse pollutants and reduce their concentration, while solar intensity can promote the dispersion of fine particles. Conversely, high humidity can promote particle formation and air quality deterioration.

By understanding these correlations, we can better analyze and predict air quality in Dublin, taking into account weather variations and their impact on pollution levels.

# 5 Implementation of Prediction Methods

This section discusses the implementation of prediction methods, focusing on KNN approaches, interpolation, and regression and machine learning models.

## 5.1 KNN KNN S/T - Implementation

The implementation of the KNN, KNN S/T approaches will be presented in detail.

Following the KNN aglorithm method, my code estimates and replaces the missing values by taking into account the spatial and temporal proximity of the neighbors.

However, to implement more or less weight to the spatial or temporal data, I introduced a new *alpha* parameter.

```
# Find the indices of the missing and existing values
missing = np.argwhere(missing_values_mask)
existing = np.argwhere(~missing_values_mask)
values = data[~missing_values_mask]

# Find the k nearest neighbors for each missing value
nbrs = NearestNeighbors(n_neighbors=n_neighbors, algorithm='auto').fit(existing)
distances, indices = nbrs.kneighbors(missing)

# Estimate the missing values using weighted averages of the nearest neighbors
estimated = np.zeros(missing.shape[0])
for i, idx in enumerate(indices):
    same_day = missing[i][0] == existing[idx, 0]
    weights = np.where(same_day, alpha, 1-alpha)
    weights /= weights.sum()
    estimated[i] = np.dot(weights, values[idx])

# Replace the missing values with the estimated values
data[missing_values_mask] = estimated
```

- If *alpha* is close to 1, space is only considered.
- If *alpha* is close to 0, time is only considered.

Now, we can adjust the weight of space / time, by changing *alpha* from 0 to 1.

## 5.2 Interpolation - Implementation

Interpolation techniques, including T-interpolation, S-interpolation, and S/T-interpolation, will be discussed in the context of their implementation.

### 5.2.1 Interpolation T

For the implementation of temporal interpolation, I first identified the dimension corresponding to time and then applied spline interpolation on this dimension. Spline interpolation was preferred over linear interpolation because it was clear that the relationship between the data was not linear.

To better understand this process on a physical level, let's consider each point in the Dublin mesh, which has 350x350 points. For each of these points, we consider all the data available for that point throughout the year. Then, we use only this data to perform the spline interpolation, which allows us to predict the missing values for the days where we have no information. Thus, spline interpolation helps us fill temporal gaps in our data set by providing accurate estimates of missing values.

**Data problems**

- **Lack of data in some locations**

  *Solution Considered*: Replace the missing data with the average of the surrounding points. However, this approach is not appropriate due to the large differences between points. A correlation with the spatial data will be required and discussed later.

- **Interpolation problems**

  1. **Insufficient points for effective spline interpolation (less than 20 points)**
     *Solution*: Calculate the average of the available points and replace all missing values with this average.

2. **Boundary interpolation problems**

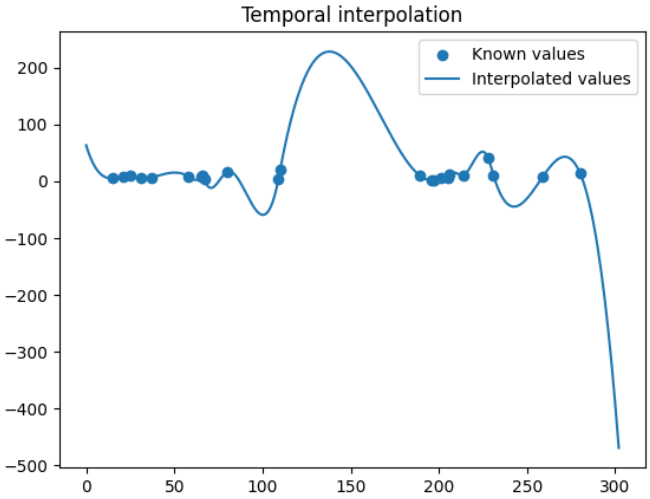



Figure 12: Interpolation S - Necessity of boundary conditions

Looking at the image, we can see that the interpolation leads to extreme negative values for predicting the last days. There are also other cases where the interpolation leads to extremely high and positive values.

*Solution*: Add boundary conditions by repeating the points at the edges twice, to create a zero derivative at the edges.

3. **Interpolation generates extreme or impossible values** Solution: Limit the interpolated values.**
   *Solution*: Limit the interpolated values by clipping them with the minimum and maximum value of the available data for each point.

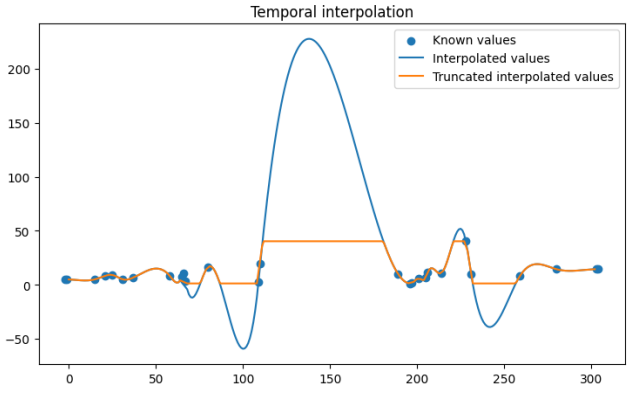



Figure 13: Interpolation S - Clip Value between min and max

### 5.2.2 Interpolation S

The implementation of the interpolation is done as follows:

1. Browse each layer of the 3D matrix.
2. **Apply interpolation** to the missing points using a specific interpolation method.
3. Adjust the interpolated values to be **between the minimum and maximum** values of each layer (to avoid the same problem as with S-interpolation)
4. R**eplace the missing values** in the 3D matrix by the interpolated values

Concerning the interpolation on a 2D matrix, the 2D spline interpolation is not available in Python. Therefore, there are two options: NN (Nearest Neighbors) and linear. However, since the quality data do not show linearity on any axis, the NN method should be preferred to perform spatial interpolation.

### 5.2.3 Interpolation S/T

In the current context, the T-interpolation does not have a sufficient number of points to completely fill the matrix. To remedy this situation, it is possible to combine the T-interpolation with the S-interpolation. Thus, by applying the S-interpolation after the T-interpolation, it is possible to obtain a more accurate and better defined S-interpolation, since more points will be available.

By combining the two interpolations, the T-interpolation first provides an initial estimate of the missing values, and then the S-interpolation refines these estimates by taking advantage of the additional points obtained from the T-interpolation. This approach improves the accuracy of the final interpolation and generates a more accurate and consistent completed matrix.

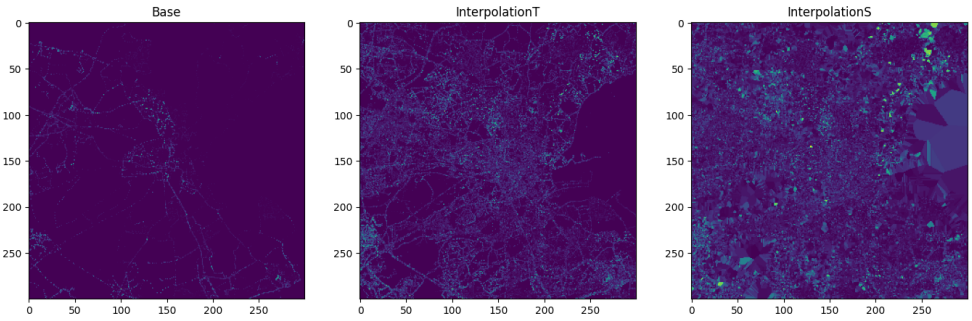



Figure 14: Interpolation ST

## 5.3 Regression and Machine Learning models - Implementation

The implementation of regression and machine learning models for missing value prediction will be discussed.

In order to optimize the performance of regression models, such as Random Forest, GBM and ANN, it is wise to use all available and relevant values for these models. As noted earlier, there is a correlation between air quality and various meteorological parameters. Thus, by incorporating this information, the models should be able to improve the accuracy of their predictions.

The input data for these models will therefore include the following:

- Longitude
- Latitude
- Day of the year
- Weather parameters (wind, temperature, humidity, etc.)

In order to measure the impact of the inclusion of these additional data on the performance of the models, we will train the models both with and without these additional variables. This approach will allow us to compare the results obtained in both cases and to evaluate the importance of this additional information to improve the quality of the predictions provided by the regression models.

# 6 Evaluation and Selection of the Best Model

The evaluation and selection of the best model are presented in this section, discussing the evaluation methods, the comparison of methods, and the selection of the best model based on the results obtained.

## 6.1 Method of evaluating models

Methods for evaluating prediction models will be presented, explaining the criteria and performance indicators used.

The method consists in evaluating the quality of the predictions of a model by manipulating and comparing 3D matrices. It proceeds in three main steps.

First, it removes points located inside a cube, defined by a given radius, around central points randomly chosen in the original 3D matrix. This process is repeated a number of times, determined by the user, to remove several areas from the matrix.

```python
def delete_random_points(matrix_3d, Nbpoints, radius=3):
    # Initialize the number of deleted points to 0
    Nb_points_deleted = 0
    # Keep deleting points until the desired number of points have been deleted
    while Nb_points_deleted < Nbpoints:
        # Select a random point within the bounds of the matrix
        point = (np.random.randint(0, matrix_3d.shape[0]),
                 np.random.randint(0, matrix_3d.shape[1]),
                 np.random.randint(0, matrix_3d.shape[2]))
        # Check if the selected point is not already deleted
        if matrix_3d[point] != 0:
            # Delete the points around the selected point within the given radius
            delete_points_arround(matrix_3d, point, radius)
            Nb_points_deleted += 1

def delete_points_arround(matrix_3d, point, radius):
    # Iterate through each point within the given radius
    for i in range(-radius, radius + 1):
        for j in range(-radius, radius + 1):
            for k in range(-radius, radius + 1):
                # Check if the point is within the bounds of the 3D matrix
                if point[0] + i >= 0 and point[0] + i < matrix_3d.shape[0] and point[1] + j >= 0 and
                    # Set the value of the point to 0
                    matrix_3d[point[0] + i, point[1] + j, point[2] + k] = 0
```

Then, the model predictions are computed on the modified 3D matrix, resulting in an estimated 3D matrix.

Finally, the method evaluates the quality of the predictions by calculating three performance metrics between the original 3D matrix and the estimated 3D matrix: the Mean Squared Error (MSE), the Mean Absolute Error (MAE) and the coefficient of determination ($R^2$). These metrics are calculated by taking into account only the non-zero points of the original 3D matrix.

In sum, this method allows us to assess the ability of the model to make predictions on missing or suppressed data by comparing the results obtained with the real values of the original 3D matrix.

## 6.2 Comparison of Methods

A comparison of the prediction methods will be made using training matrices and meteorological data.

### 6.2.1 Training model using the 3D matrix

I tested several models to predict missing air quality values without using weather data. Below is a detailed analysis of the results obtained for each model, using the following metrics: mean square error (MSE), mean absolute error (MAE), and coefficient of determination R2.

| Models | MSE | MAE | R2 |
|---|---|---|---|
| KNN | 31.175012080248365 | 1.4893224290995637 | 0.8534314882366963 |
| KNN_S | 36.1125449389733 | 1.6214324663449375 | 0.8302178053992089 |
| KNN_T | 27.20830717099467 | 1.3264911825957326 | 0.8720808486172769 |
| InterpolationST | 139.7965826559224 | 4.305014873767917 | 0.34274998781937993 |
| RandomForest | 36.10587340699423 | 1.756906844942814 | 0.8302491714339947 |
| Régression linéaire | 36.106064114204486 | 1.757034779337526 | 0.8302482748289856 |
| Régression Lasso | 36.106064114204486 | 1.757034779337526 | 0.8302482748289856 |
| Régression Ridge | 36.106064114204486 | 1.757034779337526 | 0.8302482748289856 |
| GBM | 36.106064114205715 | 1.7570347793373 | 0.8302482748289798 |
| ANN | 36.11893108829481 | 1.7643557597875892 | 0.8301877810835955 |

Table 4: Performance comparison of the different models - without weather data

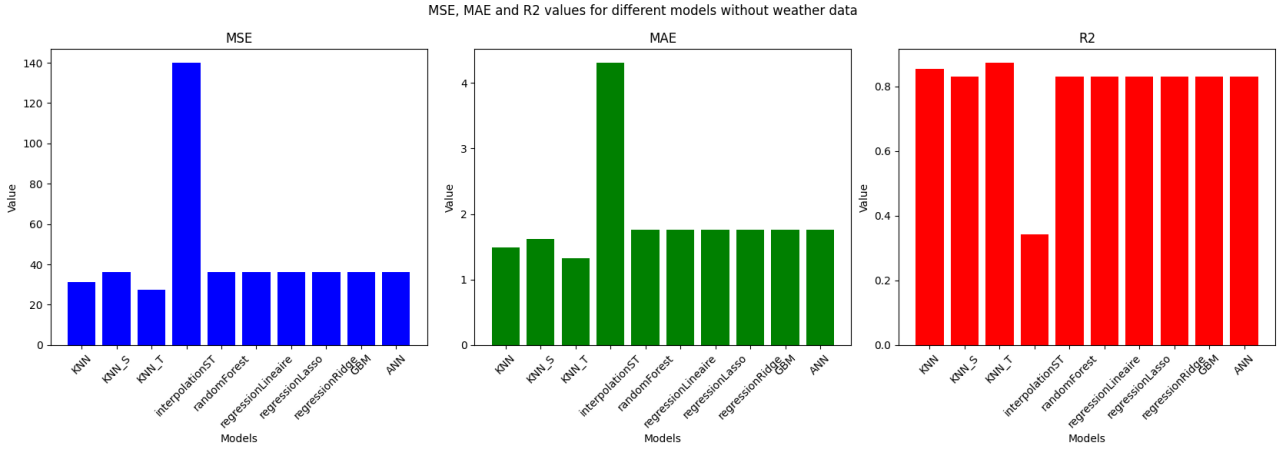



Figure 15: MSE, MAE and R2 values for different models without weather data

Among the tested models, KNN_T (k-Nearest Neighbors with time weighting) stands out for its superior performance in predicting missing air quality values without using meteorological data. With an MSE of 27.20830717099467, MAE of 1.3264911825957326, and R2 of 0.8720808486172769, KNN_T demonstrates improved ability to estimate missing values with increased precision and accuracy compared to the other models tested. Time weighting allows the model to take into account temporal variations in the data and adjust weights accordingly, which likely improves the performance of the KNN_T model.

In contrast, the ST interpolation (space-time interpolation) has the worst performance among all models, with a high MSE of 139.7965826559224 and a low R2 of 0.34274998781937993. These values suggest that interpolationST fails to effectively capture the spatiotemporal trends and variations in air quality, resulting in poor performance for this type of problem. It is possible that interpolation ST is less suitable for predicting missing values in situations where meteorological data are not available.

The other models, such as RandomForest, the different regressions (linear, Lasso, and Ridge), and ANN (artificial neural networks), have similar and relatively close results to each other in terms of MSE, MAE, and R2. However, they all perform worse than KNN_T. These models may not be able to capture the nuances and complexities of the air quality data as effectively as KNN_T, resulting in inferior performance.

In conclusion, it is recommended that the KNN_T model be used for this project, as it has demonstrated a better ability to predict missing air quality values without using meteorological data. Adoption of KNN_T would result in more accurate and reliable predictions, which would contribute to a better understanding of air quality and more informed air quality decision making.

### 6.2.2 Training model with 3D matrix and weather data

First, I decided to include meteorological parameters because as discussed above, there are often correlations between meteorological conditions and air quality. For example, meteorological conditions such as temperature, humidity, wind speed and precipitation can influence the dispersion of air pollutants. By incorporating this information into my models, I am able to capture these relationships and use this data to estimate missing air quality values more accurately.

In addition, weather forecasts are usually available in advance, while air quality measurements can take longer to collect and process. By using meteorological parameters as input to my models, I can get faster estimates of air quality for periods when data is missing. This allows me to have better continuity in my data and to fill in the gaps using available meteorological information.

In terms of dealing with missing values, using meteorological parameters allows me to adopt more sophisticated imputation techniques. I can use regression-based imputation methods, for example, to estimate missing values based on existing meteorological parameters and air quality data. This gives me the ability to replace missing values more accurately by taking into account the relationships between weather and air quality.

I left the same parameters for the evaluation method to be able to compare well. Here are the results:

| **Modèle** | **MSE** | **MAE** | **R2** |
|---|---|---|---|
| RandomForest | 19.99275484273342 | 1.131653445830308 | 0.9060045809828384 |
| Régression linéaire | 29.535271552051398 | 1.5270662862517748 | 0.8611406858555185 |
| Régression Lasso | 31.14275274111297 | 1.5765545014817817 | 0.8535831546840219 |
| Régression Ridge | 29.53528600974346 | 1.5270663178279755 | 0.8611406178830538 |
| GBM | 22.13968247175029 | 1.2593188008240142 | 0.8959108563472709 |
| ANN | 22.906498333180533 | 1.2066824050772347 | 0.8923056914377266 |

Table 5: Performance comparison of the different models - without weather data

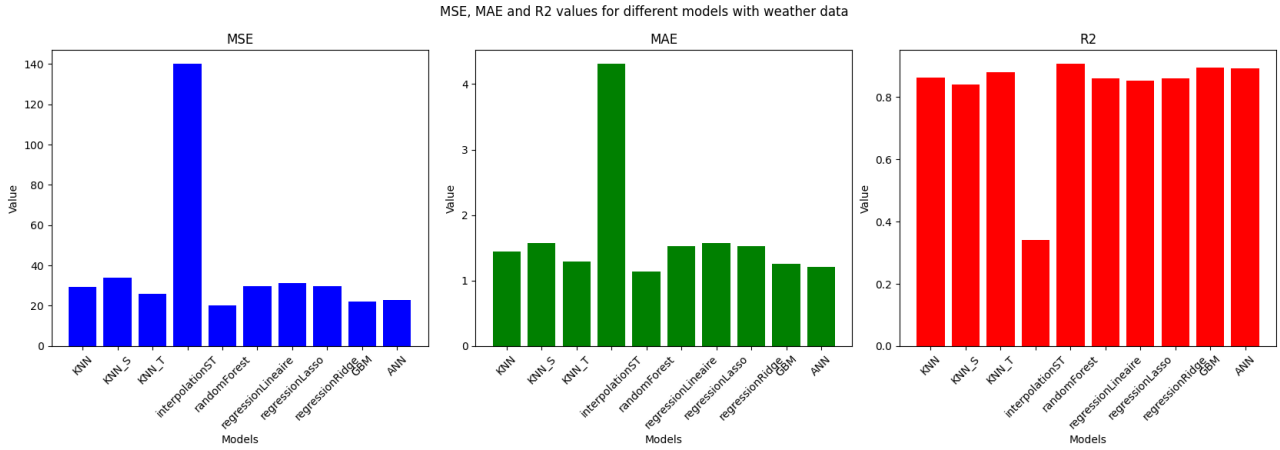



Figure 16: MSE, MAE and R2 values for different models with weather data

After integrating weather data into my Machine Learning project, I was able to observe noticeable improvements in model performance. Here is a detailed analysis of the results obtained by comparing the models without weather data and with weather data.

- **RandomForest**: The integration of weather data greatly improved the performance of the RandomForest model. The MSE decreased from 36.11 to 19.99, while the MAE also decreased from 1.76 to 1.13. The coefficient of determination R2 increased from 0.8302 to 0.9060, indicating better predictive ability of the model.
- **Linear regression**: The addition of weather data also improved the performance of the linear regression. The MSE decreased from 36.11 to 29.54, the MAE decreased from 1.76 to 1.53, and the R2 increased from 0.8302 to 0.8611.
- **Lasso Regression**: The Lasso regression showed a slight improvement with the weather data. MSE decreased from 36.11 to 31.14, MAE from 1.76 to 1.58, and R2 increased from 0.8302 to 0.8536.
- **Ridge Regression**: The performance of the Ridge regression was similar to that of the linear regression. The MSE decreased from 36.11 to 29.54, the MAE from 1.76 to 1.53, and the R2 increased from 0.8302 to 0.8611.
- **Gradient Boosting Machines (GBM)**: The integration of weather data improved the performance of the GBM model. The MSE decreased from 36.11 to 22.14, the MAE from 1.76 to 1.26, and the R2 increased from 0.8302 to 0.8959.
- **Artificial neural networks (ANN)**: ANN model performance also improved with the addition of weather data. MSE decreased from 36.12 to 22.91, MAE from 1.76 to 1.21, and R2 increased from 0.8302 to 0.8923.

In conclusion, the integration of meteorological data into the Machine Learning models significantly improved their performance in predicting missing air quality values. The RandomForest, GBM, and ANN models showed the most significant improvements in terms of MSE, MAE, and R2.

These results highlight the importance of meteorological variables for air quality modeling, as they directly and indirectly impact air pollutant concentrations. By considering these variables, we can develop more accurate and robust models for air quality prediction, which can facilitate air pollution monitoring and management.

## 6.3 Visualization of the result of some models

### 6.3.1 Creation of csv file from 3D matrix

```
def export_csv(matrix_pm25, matrix_pm10, name):

# Export the matrix as CSV files, one for each day in the matrix.
for i in range(np.shape(matrix_pm25)[0]):
    # Calculate the date corresponding to the current iteration.
    date = pd.to_datetime(date_min, format='%Y-%m-%d') + pd.Timedelta(days=i)

    # Open a new CSV file for writing.
    # The file name is based on the date in the format 'YYYY-MM-DD.csv'.
    with open(name + "/" + date.strftime("%Y-%m-%d") + ".csv", 'w') as f:
        # Iterate over each point in the matrix.
        for j in range(np.shape(matrix_pm25)[1]):
            for k in range(np.shape(matrix_pm25)[2]):
                # Calculate the timestamp for the current date and time (12:00 PM).
                timestamp = (pd.to_datetime(date_min, format='%Y-%m-%d')
                    + pd.Timedelta(days=i) + pd.Timedelta(hours=12)).timestamp()
                id = 0
                # Calculate the longitude and latitude based on the position in the matrix.
                longitude = longitude_hautGauche + k *
                    abs((longitude_hautGauche - longitude_basDroit) / size_matrix)
                latitude = latitude_basDroit + j *
                    abs((latitude_hautGauche - latitude_basDroit) / size_matrix)
                pm25 = matrix_pm25[i, j, k]
                pm10 = matrix_pm10[i, j, k]
                # Write the data to the CSV file in the format:
                # timestamp;id;pm25;pm10;latitude;longitude
                f.write(
                      str(timestamp) + ";"
                    + str(id) + ";"
                    + str(pm25) + ";"
                    + str(pm10) + ";"
                    + str(latitude) + ";"
                    + str(longitude) + "\n")
```

The function export_csv takes as parameters a matrix_pm25, a matrix_pm10 and a name. It performs the following actions:

1. Exports the matrix_pm25 and matrix_pm10 data as CSV files. Each CSV file represents a different day.
2. The CSV files are created with the same format as the original file, with the following columns: timestamp, id, pm25, pm10, latitude and longitude.
3. The values of the matrices are written to the corresponding CSV files in the format mentioned above.

### 6.3.2 Visualization

In this section, we will examine the visual results of several models by displaying them on the website we developed for this project. This will allow us to evaluate their performance and relevance in representing air quality in Dublin.

To begin this step, we proceeded to add multiple functionalities to the HTML site selector, in order to be able to integrate the predictions generated by the different models. In addition, it was necessary to make changes to the request sent to the PHP code so that it could select the appropriate data directory in its database.

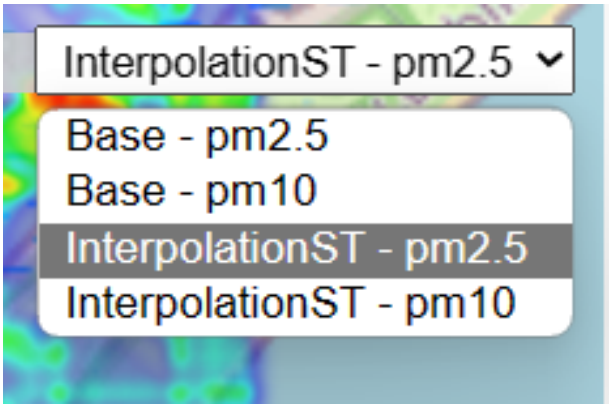



Figure 17: Adding prediction to the selector

Then, let's analyze the results obtained with the InterpolationST model:

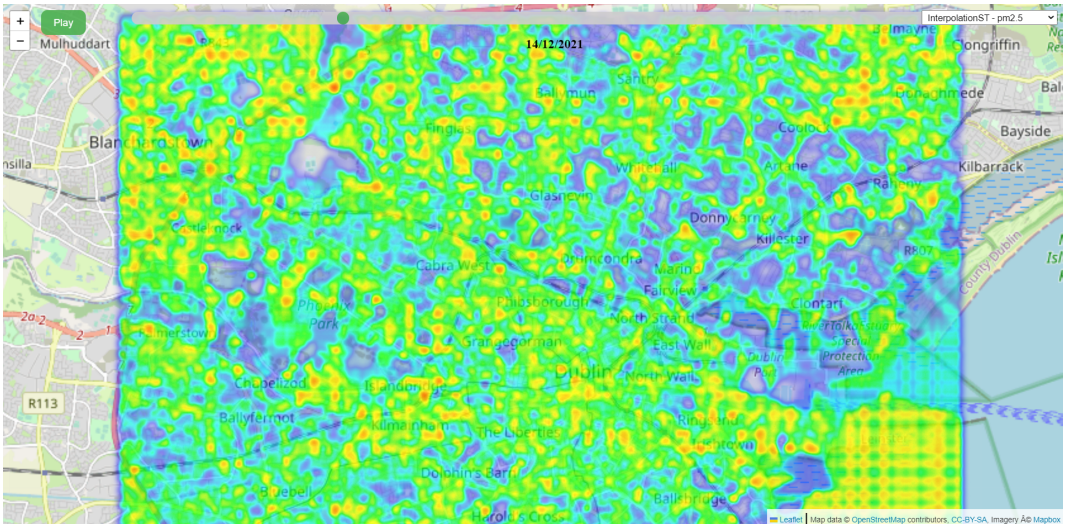



Figure 18: Visualization of Interpolation-ST Brut

At first glance, we see that the data is noisy and has very fine details that seem inconsistent with reality. To remedy this problem, I decided to apply a low-pass filter to smooth the values. To do this, I performed a convolution on the data matrix using a uniform kernel of dimensions 20x20.

$$(I * K)(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(m,n) \cdot K(i-m, j-n)$$

In this equation, I is the input matrix and K is the convolution matrix, also called the kernel. The convolution is computed by summing the element-by-element products between the elements of the input matrix and the corresponding kernel elements, after an appropriate shift. The result of

the convolution is a new matrix that represents the combination of the two matrices through the convolution kernel.

$$\begin{bmatrix} \frac{1}{400} & \frac{1}{400} & \frac{1}{400} & \cdots & \frac{1}{400} \\ \frac{1}{400} & \frac{1}{400} & \frac{1}{400} & \cdots & \frac{1}{400} \\ \frac{1}{400} & \frac{1}{400} & \frac{1}{400} & \cdots & \frac{1}{400} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{400} & \frac{1}{400} & \frac{1}{400} & \cdots & \frac{1}{400} \end{bmatrix}$$

Here is the output:

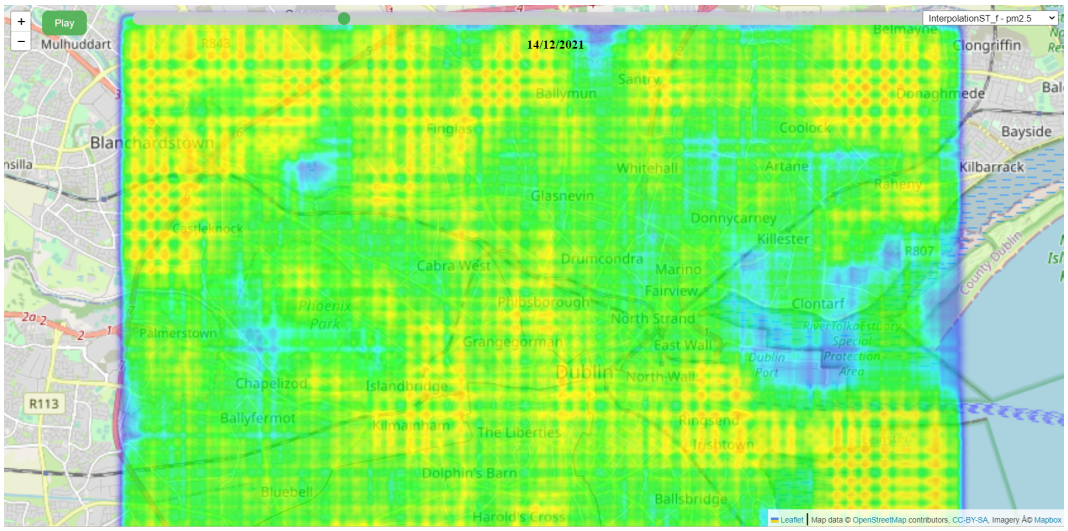



Figure 19: Visualization of Interpolation-ST with convolution

Following the application of the filter, the resulting image is much more usable and realistic. It allows for a better identification of polluted and unpolluted areas, thus providing a solid basis for the analysis and interpretation of the data. I then proceeded in the same way for the other models studied. For example, here is the visual output of the Random Forest model:

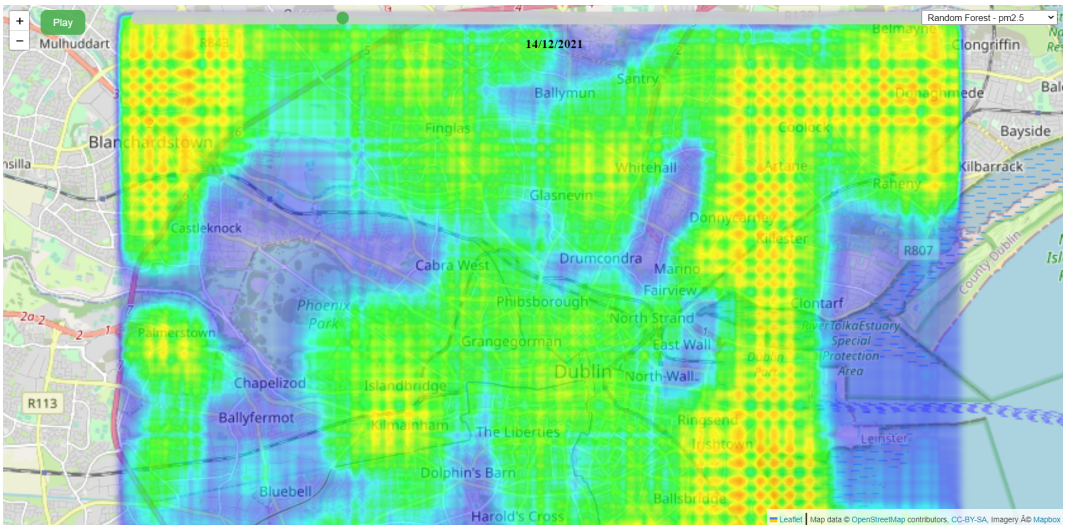



Figure 20: Visualization of random forest with convolution

## 6.4 Selection of the Best Method

The selection of the best prediction model, with and without meteorological data, will be discussed.

To provide a more detailed analysis of model performance for air quality prediction without and with meteorological data, we will examine the top three models for each scenario in terms of MSE (Mean Squared Error), MAE (Mean Absolute Error) and R2 (coefficient of determination).

### 6.4.1 Without weather data

1. **KNN_T**: The KNN_T model performed best among all models tested without weather data. The MSE is 27.21, the MAE is 1.33 and the R2 is 0.8721.

2. **KNN**: The second best model is KNN, with an MSE of 31.18, MAE of 1.49 and R2 of 0.8534. Although the performance of this model is slightly lower than KNN_T, it is still relatively good compared to the other models.

3. **KNN_S**: In third place, the KNN_S model has an MSE of 36.11, an MAE of 1.62, and an R2 of 0.8302. This model performs worse than the KNN_T and KNN models, but outperforms the other models without weather data.

### 6.4.2 With weather data

1. **RandomForest**: The best model with weather data is RandomForest, with an MSE of 19.99, MAE of 1.13, and R2 of 0.9060. Incorporating weather data significantly improved the performance of this model, placing it at the top of the models tested.

2. **Gradient Boosting Machines (GBM)**: The second best model was the GBM, with an MSE of 22.14, MAE of 1.26, and R2 of 0.8959. Although the performance of GBM is slightly lower than RandomForest, it is still excellent and shows the importance of weather data in improving predictions.

3. **Artificial Neural Networks (ANN)**: In third place, the ANN model has an MSE of 22.91, an MAE of 1.21, and an R2 of 0.8923. This model also benefited from the integration of weather data, and its performance is just below that of the GBM.

In summary, for forecasting without weather data, the KNN_T, KNN, and KNN_S models perform best, while with weather data, the RandomForest, GBM, and ANN models are the best choices. However, it is essential to keep in mind that the performance of the models may vary depending on the specifics of the problem and the data. Therefore, it is recommended to test several models and evaluate their performance to choose the best one for the situation.

# 7 Ethics

In this section, we will discuss the ethical issues related to the use and analysis of air quality data.

The project aims to process air quality data collected by sensors placed on delivery trucks in Dublin. Ethical considerations associated with this project are essential to ensure compliance with ethical standards and principles.

1. **Privacy and Data Protection** The data collected by the sensors includes GPS location, which can potentially identify specific individuals or locations. It is crucial to ensure that the data is anonymized and that personal information is protected in accordance with applicable regulations, such as the GDPR.

2. **Transparency** The methods and results of air quality analysis must be clearly presented and explained to ensure transparency and understanding of the process. Researchers must be accountable for their decisions and consider the potential impact of their work on society and the environment.

3. **Equity and non-discrimination** The project must ensure that the data and results of the air quality analysis do not unfairly bias or discriminate against certain groups or neighborhoods. Researchers must ensure that the sampling and data collection methodology is fair and representative.

4. **Stakeholder consent and participation** It is important to obtain informed consent from relevant stakeholders, such as delivery truck owners and local authorities, to ensure ethical and respectful collaboration. The project should also consider the views and concerns of residents and encourage their participation.

By addressing these ethical issues, the project can responsibly contribute to the improvement of Dublin's air quality and the well-being of its residents, while respecting ethical principles and data protection regulations.

# 8 Conclusion and Future Work

In conclusion, this project has provided a better visualization and understanding of air quality data in the Dublin area. We identified a significant correlation between meteorological data and fine particulate matter (pm2.5) concentrations in the air. Furthermore, the best performing models for predicting missing air quality (pm2.5) values were found to be:

- The KNN model without the integration of meteorological data
- The random forest, GBM and ANN models with meteorological data included

Regarding the prospects for improving and extending this research, several directions can be considered:

- Analyze a larger dataset to better understand the impact of seasonal variations on air quality and refine predictions based on these variations.
- Incorporate weather forecasts into the models to improve their accuracy and ability to anticipate air quality fluctuations based on weather conditions.
- Extend the current approach to predict not only missing values, but also future air quality values, which would allow the development of decision support tools for environmental management and urban planning stakeholders.

# 9 Follow-up of the project in EE457

Throughout the realization of this project, a regular and structured follow-up was set up to ensure the progression and the good progress of the work. This follow-up took the form of weekly meetings with my supervisor, Dr. Ali Intizar. These meetings allowed us to discuss the progress of the project, to evaluate the results obtained and to determine the orientations to be taken for the continuation of the work.

At each meeting, we discussed several key aspects of the project:

- **Assessing progress**: We reviewed the progress made over the past week, identifying milestones achieved and results obtained. This gave us an overview of how the project was progressing and where we needed to focus our efforts.
- **Discussion of issues**: In the event that difficulties or obstacles arose, these weekly meetings provided an opportunity to discuss them with Dr. Intizar. With his expertise and wise counsel, we were able to find appropriate and effective solutions to overcome these challenges and stay on track with our goals.
- **Planning Next Steps**: At the end of each meeting, we defined the goals we wanted to achieve for the following week and developed an action plan to get there. This planning allowed us to optimize the organization and the distribution of tasks, thus ensuring a constant and coherent progression of the project.
- **Adjustments and orientations**: Finally, these weekly meetings were also an opportunity to discuss the general direction of the project. Depending on the results obtained and the problems encountered, we were able, if necessary, to readjust our priorities and work methods to ensure that our objectives were met.

Thanks to this regular monitoring and the close collaboration with Dr. Ali Intizar, we were able to successfully complete this project, overcoming obstacles and adapting our strategies as the work evolved. This approach was crucial to ensure the quality and relevance of our research and to produce results that were both reliable and usable.

# References


[1] Nuryazmin Ahmat Zainuri, Abdul Aziz Jemain, and Nora Muda. A comparison of various imputation methods for missing values in air quality data. 44(3):449–456.

[2] Ahmad R. Alsaber, Jiazhu Pan, and Adeeba Al-Hurban. Handling complex missing data using random forest approach for an air quality monitoring dataset: A case study of kuwait environmental data (2012 to 2018). 18(3):1333.

[3] Heikki Junninen, Harri Niska, Kari Tuppurainen, Juhani Ruuskanen, and Mikko Kolehmainen. Methods for imputation of missing values in air quality data sets. 38(18):2895–2907.

[4] Soubhik Mahanta, T. Ramakrishnudu, Rajat Raj Jha, and Niraj Tailor. Urban air quality prediction using regression analysis. In *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pages 1118–1123. ISSN: 2159-3450.

[5] Kar Yong Ng and Norhashidah Awang. Multiple linear regression and regression with time series error models in forecasting PM10 concentrations in peninsular malaysia. 190(2):63.

[6] Norazian Mohamed Noor, Mohd Mustafa Al Bakri Abdullah, Ahmad Shukri Yahaya, and Nor Azam Ramli. Comparison of linear interpolation method and mean method to replace the missing values in environmental data set. 803:278–281. Publisher: Trans Tech Publications Ltd.

[7] Muhammad Izzuddin Rumaling, Fuei pien Chee, Jedol Dayou, Jackson Chang, Steven Kong, and J. Sentian. Missing value imputation for PM10 concentration in sabah using nearest neighbour method (NNM) and expectation-maximization (EM) algorithm. 14:62–72.

[8] Jasleen Kaur Sethi and Mamta Mittal. An efficient correlation based adaptive LASSO regression method for air quality index prediction. 14(4):1777–1786.

[9] Rasa Zalakeviciute, Yves Rybarczyk, Katiuska Alexandrino, Santiago Bonilla-Bedoya, Danilo Mejia, Marco Bastidas, and Valeria Diaz. Gradient boosting machine to assess the public protest impact on urban air quality. 11(24):12083. Number: 24 Publisher: Multidisciplinary Digital Publishing Institute.

[10] Rui Zhao, Xinxin Gu, Bing Xue, Jianqiang Zhang, and Wanxia Ren. Short period PM2.5 prediction based on multivariate linear regression model. 13(7):e0201011.

# AIR QUALITY IN DUBLIN USING IOT AND MACHINE LEARNING

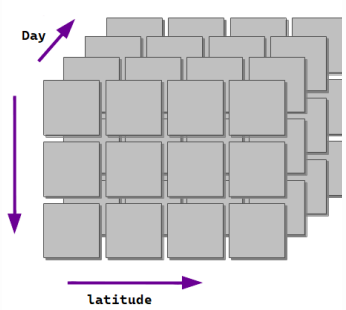
## PROBLEM STATEMENT

- Data Collection using DPD vehicles equipped with sensors
- Air Quality Indicators: pm2.5 and pm10, GPS coordinates, date and time

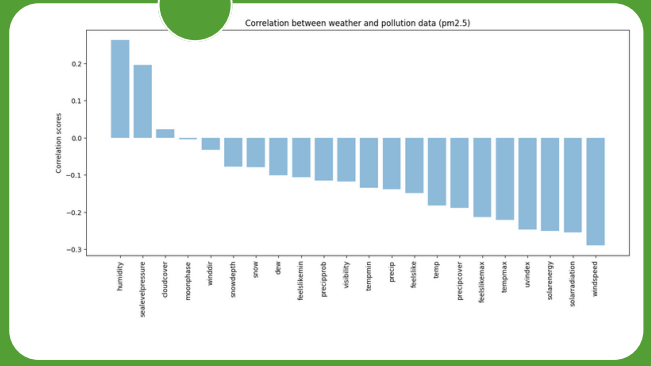
**Handling Missing Values:** (~80% missing values)

- Spatial Missing Values: Unequal distribution of data
- Temporal Missing Values: Days with little or no data

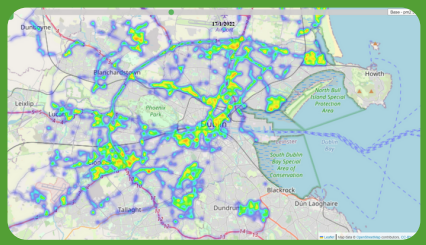
**Objective:**
Create a complete and reliable dataset for in-depth analysis of air quality in Dublin



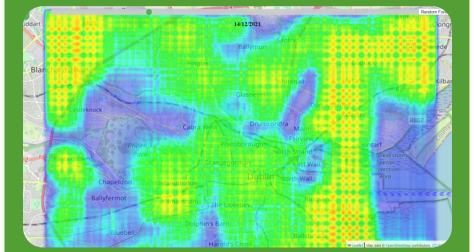

## 3D MATRIX


## CORRELATION WITH WEATHER DATA



## PREDICTION RESULTS



## PREDICTION RESULTS