# POLLUTANT ALERT SYSTEM


2020-2021

ALBERTO JIMÉNEZ DOMINGUEZ

MASTER DATA SCIENCE
KSCHOOL

## DEDICATED TO

My familiar "nucli dur",

for the support received to start

and the confidence to FINISH this PROJECT.


## ACKNOWLEDGMENTS

To Álex, for the advice and guidance.

To Hardland, for giving light to this tunnel.

# ABSTRACT

Solid particles suspended in the air are responsible for over 160.000 deaths in biggest five cities in the world during 2020[1]. A high concentration of particles has adverse health effects to humans allocated in massive areas of population. In Catalonia, existing alert systems are focused on warning when excess of $NO_2$ and $PM_{10}$ pollutants are detected[2][3], but behind that, small-size particle, known as $PM_{2.5}$, was demonstrated to become more dangerous than the other two.

The aim of this work is to develop an Alert System from $PM_{2.5}$ particles when its concentration is exceeding safety threshold. To do that, multiple machine learning models were explored, basically Classification models, based on database meteorological and air pollutant features measured in detection stations.

Databases were extracted, pre-processed, pre-sampled, over-sampled and feature selection applied to simplify complexity of the data. Different classification models were run to evaluate the most suitable one, and to launch later a FrontEnd. Results got, demonstrated that SVC tuned method provided best F1, Recall and Accuracy scores (metrics used).

As $PM_{2.5}$ detection is complex and not available in many cities, the pilot Alert $PM_{2.5}$ System is focused on a specific area of Tarragona city called Bonavista, where surrounded industries, schools and high density of population concentrated, create the excellent job-place to be analysed.

[1] https://www.greenpeace.org/southeastasia/press/44319/pm2-5-air-pollution-behind-an-estimated-160000-deaths-in-world-5-biggest-cities-in-2020/

[2] Pollutants description: http://mediambient.gencat.cat/ca/05_ambits_dactuacio/atmosfera/qualitat_de_laire/principals_contaminants/

[3] http://mediambient.gencat.cat/ca/05_ambits_dactuacio/atmosfera/qualitat_de_laire/qualitat-de-laire-a-la-conurbacio-de-barcelona/pla_millora_qua_aire_2011_2015/mesures-del-pamqa/episodis_ambientals/

# INDEX

# Forewords

Based on suggested structure of the TFM on this Master[4]:

A Memory will be written in order to support a repository uploaded into https://github.com/alji81/TFM.git to be followed and replicated by someone with knowledge in Data Science. We should demonstrate in both deliverables, the capacity to work as data scientist and able to solve a problem or research question with data.

TFM should include:

1. Introduction: what, how and why it is relevant
2. Raw data description
3. Machine learning techniques used
4. Summary of main results
5. Conclusions
6. User manual for the FrontEnd

Structure of this document:

1. Memory. It contains all explanation included between Abstract to Appendix, incorporating all sections noticed above.
2. Scripts. It defines and contains all code developed and instructions to follow and reply the TFM.

[4] https://kschool.myopenlms.net/course/view.php?id=14#section-33 (last access: 07-07-2021)

All data information is analysed in meteorological and air pollutant stations allocated closed one each other, and all data extracted, is uploaded into 'Govern' authority Open-sourced data[5].

Based on all above, I hope reader will find interesting the issue discussed, and potentially, my intention is to extend this Alert System $PM_{2.5}$ to other cities, through increasing investment of $PM_{2.5}$ detectors in other areas of Catalonia where impact could become bigger.

From EU (European Union) and WHO (World Health Organization) air pollution control is a must and $PM_{2.5}$ is considered more dangerous than $PM_{10}$[6][7].

[5] http://governobert.gencat.cat/ca/dades_obertes/)

[6] https://www.euro.who.int/__data/assets/pdf_file/0006/189051/Health-effects-of-particulate-matter-final-Eng.pdf

[7] https://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//NONSGML+IM-PRESS+20060922IPR10875+0+DOC+PDF+V0//ES&language=ES

# Chapter 1 - Introduction

## 1.1 Why

It was chosen an "Alert system Pollutant" because today, in Catalonia, it just exists similar one for $NO_2$ and $PM_{10}$ pollutants[2], but not available for $PM_{2.5}$. Moreover, $PM_{2.5}$ was demonstrated as more accurate way to define the pollution in a specific area[8] and more impacting into the vulnerable population[9]. For this reason, the area selected was Tarragona (Bonavista), because of surrounded industry, schools and high-density population co-living together.

## 1.2 What

The main objective of this TFM is to develop a $PM_{2.5}$ Pollutant Alert System application capable to predict in 24h forecast if to Activate or not some countermeasures to keep pollutant values under threshold established by WHO. To do so, this TFM will allow someone with a bit of knowledge in Data Science to understand how to :

1. Live download, pre-process and merge datasets from different formats structured into a unique dataset to be analysed.
2. Explore and select different features, and to compare different preparation and sampling methods.
3. Launch, compare and evaluate different Machine Learning classification techniques under comprehensive metrics.
4. Build a friendly FrontEnd interface to allow a non-Data Science knowledge user predicts $PM_{2.5}$ pollutant 24h forecast.

8 https://www.um.es/estructura/servicios/sprevencion/c-seguridad/documentos/EE_58_PM25.pdf

9 LINARES, Cristina y DIAZ, Julio. Efecto de las partículas de diámetro inferior a 2,5 micras (PM2,5) sobre los ingresos hospitalarios en niños menores de 10 años en Madrid. Gac Sanit [online]. 2009, vol.23, n.3 [citado 2021-07-07], pp.192-197. Disponible en: <http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S0213-91112009000300005&lng=es&nrm=iso>. ISSN 0213-9111.

## 1.3 How

Different techniques, tools and services were used to develop TFM presented:

1. Jupyter Notebooks for importing, cleansing, exploring, preparing data and launching models.
2. Python to develop FrontEnd.
3. Github to share and control version of TFM Notebooks.
4. Google Collab to run Jupyter Notebooks.
5. Microsoft word and Acrobat Reader to elaborate TFM memory.

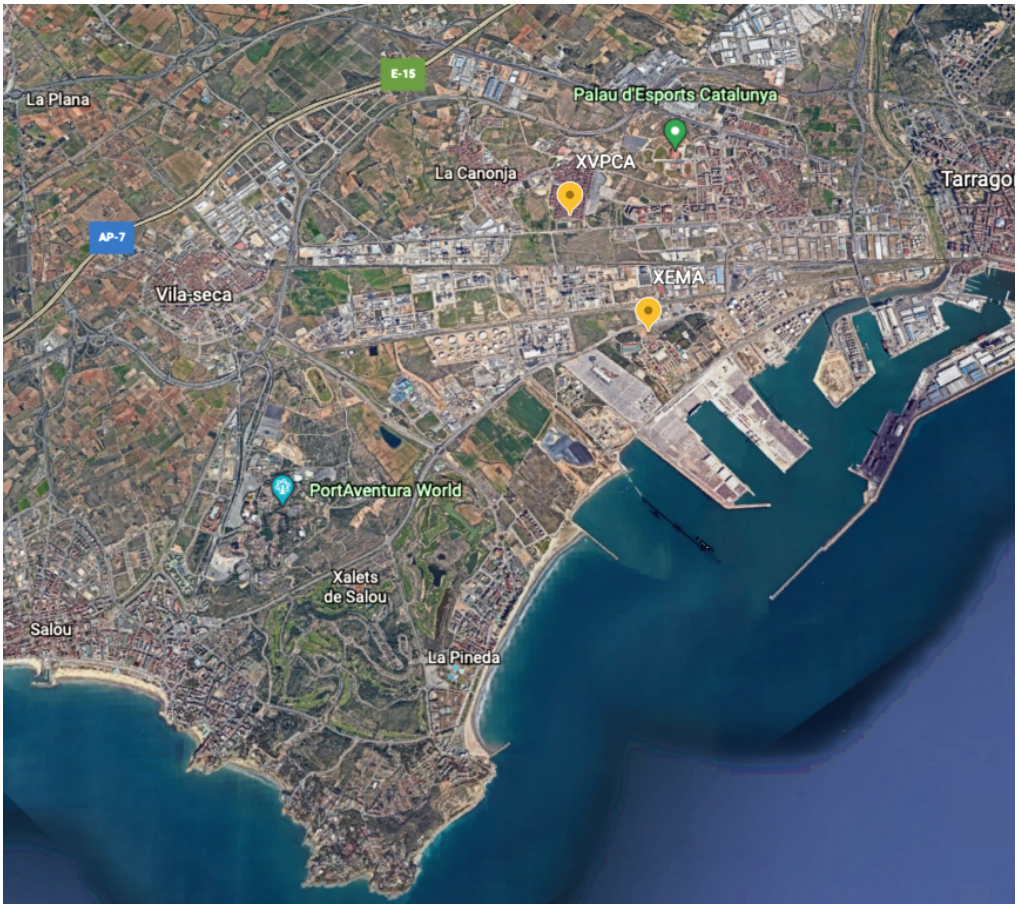
Jupyter Notebooks, python FrontEnd and documents related, Datasets generated by Notebooks and Memory were uploaded into Github repository https://github.com/alji81/TFM.git.

To reply TFM, it is needed to follow strictly items (Notebooks) ordered below:

- Data acquisition:
  - Pollutants: 1A. Obtain_data_pollutants.ipynb
  - Meteorology: 1B. Obtailn_data_meteo.ipynb
- Data merge:
  - 2. Merge_data_pollutant_meteo.ipynb
- Data cleansing, preparation and Analysis:
  - 3. Modelling.ipynb
- FrontEnd:
  - FrontEnd.ipynb

## 1.4 Where

Tarragona-Bonavista (North-East Spain) was the location selected because of long-time $PM_{2.5}$ pollutant data available, and interested area where industries, schools and high-density population is co-living together.




- *Tarragona-Bonavista XVPCA (Xarxa de Vigilància i Previsió de la Contaminació Atmosfèrica):* Air pollutant detection station.
- *Tarragona-Complex Educatiu XEMA (Xarxa d'Estacions Meteorològiques Automàtiques):* Meteorological detection station.

In this case, XVPCA (1.1919986,41.11591) and XEMA (1.201 41.10393) Bonavista stations are separated by 1,3 km.

# Chapter 2 - Related work

Air pollution is globally one of the most critical issues due to adverse health effects in many cities. Particular matter is $PM_{2.5}$[10]. Therefore, TFM is focused in $PM_{2.5}$ only.

Many researches have been done for developing predictive models to estimate $PM_{2.5}$ concentrations in advanced, being the aim of most of these researches forecasting directly a value of $PM_{2.5}$[11]. In our case, it was considered to develop an Alert System of $PM_{2.5}$ focused on activating measures when a threshold is exceeded ($25\mu g/m^3$)[5] with the intention of simplifying and warning easily to population. As summary, to develop a similar existing $NO_2$ and $PM_{10}$ Alert System in Catalonia[12], but for $PM_{2.5}$.

In comparison with works mentioned before, our TFM is based only on locally collected features. It means that, meteorological detectors and air pollutant detectors are closely allocated in Bonavista area, near to Tarragona city in Spain.

10 Jan Kleine Deters, Rasa Zalakeviciute, Mario Gonzalez, Yves Rybarczyk, "Modeling PM2.5 Urban Pollution Using Machine Learning and Selected Meteorological Parameters", Journal of Electrical and Computer Engineering, vol. 2017, Article ID 5106045, 14 pages, 2017. https://doi.org/10.1155/2017/5106045

11 Mauro Castelli, Fabiana Martins Clemente, Aleš Popovič, Sara Silva, Leonardo Vanneschi, "A Machine Learning Approach to Predict Air Quality in California",Complexity, vol. 2020, Article ID 8049504, 23 pages, 2020.https://doi.org/10.1155/2020/8049504

12 http://mediambient.gencat.cat/ca/05_ambits_dactuacio/atmosfera/qualitat_de_laire/qualitat-de-laire-a-la-conurbacio-de-barcelona/pla_millora_qua_aire_2011_2015/mesures-del-pamqa/episodis_ambientals/

# Chapter 3 - Data

## 3.1 Datasets

Datasets were downloaded directly from the domain belonging to government of Catalonia http://www.analisi.transparenciacatalunya.cat through running Notebook accordingly.

### *3.1.1 Air pollutant*

In case of Air pollutant data, identifier data set was "tasf-thgu" Jason file. Once there, it was filtered by nom_estacio:"Tarragona(Bonavista)", as seen in 1A.Obtain_data_conta.ipynl Jupyter Notebook. Table 1 highlights Features identified in columns including time samples got, and in rows, it can be seen different pollutants and date.

| | codi_eoi | nom_estacio | data | magnitud | contaminant | unitats | tipus_estacio | area_urbana | codi_ine | municipi | ... | h19 | h20 | h21 | h22 | h23 | h24 | altitud | latitud | longitud | geocoded_column |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 43148003 | Tarragona (Bonavista) | 2014-07-28T00:00:00.000 | 11 | PM1 | µg/m3 | industrial | peri-urban | 43148 | Tarragona | ... | 10 | 9 | 7 | 8 | 10 | 11 | 39 | 41.11591 | 1.191999 | {'type': 'Point', 'coordinates': [1.191999, 41... |
| 1 | 43148003 | Tarragona (Bonavista) | 2014-05-04T00:00:00.000 | 7 | NO | µg/m3 | industrial | peri-urban | 43148 | Tarragona | ... | 3 | 2 | 1 | 1 | 1 | 3 | 39 | 41.11591 | 1.191999 | {'type': 'Point', 'coordinates': [1.191999, 41... |
| 2 | 43148003 | Tarragona (Bonavista) | 2014-04-15T00:00:00.000 | 11 | PM1 | µg/m3 | industrial | peri-urban | 43148 | Tarragona | ... | 10 | 14 | 14 | 17 | 20 | 31 | 39 | 41.11591 | 1.191999 | {'type': 'Point', 'coordinates': [1.191999, 41... |
| 3 | 43148003 | Tarragona (Bonavista) | 2014-10-09T00:00:00.000 | 11 | PM1 | µg/m3 | industrial | peri-urban | 43148 | Tarragona | ... | 19 | 15 | 18 | 16 | 22 | 26 | 39 | 41.11591 | 1.191999 | {'type': 'Point', 'coordinates': [1.191999, 41... |
| 4 | 43148003 | Tarragona (Bonavista) | 2014-05-15T00:00:00.000 | 12 | NOX | µg/m3 | industrial | peri-urban | 43148 | Tarragona | ... | 6 | 3 | 3 | 13 | 6 | 5 | 39 | 41.11591 | 1.191999 | {'type': 'Point', 'coordinates': [1.191999, 41... |

Table 1. Air pollutant dataset (67.847 rows x 40 columns)

One of the first actions done were to convert object features to Datetime for Data, and float values for measures. Period time chosen was from 01/01/2010 to 31/12/2020. Because of Normative threshold definition, and thus simplifying dataframe, it was created an average value column from different hours and maximum value got during one day.

After several data treatments, including pivot table function with dataframe created, we reduce original dataset to one with date by day in rows as sample, and columns with features from air pollutants average and maximum values (µg/m$^3$ unit), as seen in Table 2 below.

| | data | CO max | H2S max | NO max | NO2 max | NOx max | PM1 max | PM10 max | PM2.5 max | SO2 max | CO | H2S | NO | NO2 | NOx | PM1 | PM10 | PM2.5 | SO2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2010-01-01 | 0.2 | 1.4 | 1.0 | 7.0 | NaN | NaN | NaN | NaN | 1.0 | 0.200000 | 1.108333 | 1.000000 | 2.625000 | NaN | NaN | NaN | NaN | 1.000000 |
| 1 | 2010-01-02 | 0.2 | 1.8 | 13.0 | 44.0 | NaN | NaN | NaN | NaN | 1.0 | 0.200000 | 1.137500 | 2.250000 | 12.916667 | NaN | NaN | NaN | NaN | 1.000000 |
| 2 | 2010-01-03 | 0.3 | 1.6 | 17.0 | 48.0 | NaN | NaN | NaN | NaN | 7.0 | 0.204167 | 1.158333 | 3.625000 | 22.166667 | NaN | NaN | NaN | NaN | 1.875000 |
| 3 | 2010-01-04 | 0.2 | 2.0 | 26.0 | 42.0 | NaN | NaN | NaN | NaN | 4.0 | 0.200000 | 1.382609 | 8.217391 | 28.304348 | NaN | NaN | NaN | NaN | 1.652174 |
| 4 | 2010-01-05 | 0.4 | 1.4 | 23.0 | 44.0 | NaN | NaN | NaN | NaN | 2.0 | 0.220833 | 1.037500 | 5.750000 | 24.958333 | NaN | NaN | NaN | NaN | 1.083333 |

Table 2. Air pollutant average and maximum values (µg/m$^3$) grouped by day dataframe (3.992 rows x 19 columns)

Dataset created was exported to csv ('Contaminacio_ready.csv') to be merged later with Meteorological one.

(Pollutant acronyms are explained in page 3, Abstract's reference 2)

### *3.1.2 Meteorological*

Meteorological dataset identifier was: "nzvn-apee" Jason file. Despite it was desired to filter by period of time, Socrata API instructions wasn't clear, and it was decided to download all info available filtered by 'codi_estacio': 'XE', as seen in 1B.Obtain_data_meteo.ipynb. Table 3 highlights Features identified in columns including time samples got, and in rows, it can be seen different meteorological features and date by measurements registered every thirty minutes.

| | id | codi_estacio | codi_variable | data_lectura | valor_lectura | codi_estat | codi_base | data_extrem |
|---|---|---|---|---|---|---|---|---|
| 0 | XE360502091530 | XE | 36 | 2009-02-05T15:30:00.000 | 197 | V | SH | NaN |
| 1 | XE340502091530 | XE | 34 | 2009-02-05T15:30:00.000 | 996 | V | SH | NaN |
| 2 | XE350502091530 | XE | 35 | 2009-02-05T15:30:00.000 | 0 | V | SH | NaN |
| 3 | XE440502091600 | XE | 44 | 2009-02-05T16:00:00.000 | 60 | V | SH | 2009-02-05T16:05:00.000 |
| 4 | XE420502091600 | XE | 42 | 2009-02-05T16:00:00.000 | 14.4 | V | SH | 2009-02-05T16:28:00.000 |

Table 3. Meteorological dataset (3.451.993 rows x 8 columns)

Again, one of the first actions done were to convert object features to Datetime for Data, and float values for measures. Dataframe was reduced, and a pivot table was created to configure the desired dataframe as a combination of samples grouped by day in rows and extended meteorological features in columns. Additionally, numeric features from the original dataset were labelled accordingly by downloading features dataset separately.

All above and extended info available in Notebook helped to reduce dataframe desired, as seen in Table 4, below.

| | data_lectura | Pressió atmosfèrica màxima | Pressió atmosfèrica mínima | Humitat relativa màxima | Velocitat del vent a 10 m (esc.) | Direcció de vent 10 m (m. 1) | Temperatura | Humitat relativa | Pressió atmosfèrica | Precipitació | Irradiància solar global | Temperatura màxima | Temperatura mínima | Humitat relativa mínima | Ratxa màxima del vent a 10 m | Direcció de la ratxa màxima del vent a 10 m | Precipitació màxima en 1 minut |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2010-01-01 | 1003.531915 | 1002.680851 | 45.191489 | 5.848936 | 267.234043 | 11.278723 | 44.170213 | 1003.127660 | 0.000000 | 71.574468 | 11.444681 | 11.102128 | 43.042553 | 12.021277 | 271.957447 | 0.000000 |
| 1 | 2010-01-02 | 1016.875000 | 1016.312500 | 57.020833 | 3.537500 | 220.875000 | 9.833333 | 55.625000 | 1016.645833 | 0.000000 | 61.458333 | 10.056250 | 9.604167 | 54.312500 | 6.714583 | 216.187500 | 0.000000 |
| 2 | 2010-01-03 | 1016.458333 | 1016.125000 | 79.250000 | 1.527083 | 101.083333 | 9.666667 | 78.000000 | 1016.312500 | 0.000000 | 93.895833 | 9.927083 | 9.418750 | 76.687500 | 2.870833 | 118.208333 | 0.000000 |
| 3 | 2010-01-04 | 1008.625000 | 1008.354167 | 82.583333 | 1.802083 | 81.458333 | 9.406250 | 81.625000 | 1008.500000 | 0.250000 | 36.604167 | 9.572917 | 9.233333 | 80.500000 | 3.570833 | 82.333333 | 0.027083 |
| 4 | 2010-01-05 | 1000.083333 | 999.833333 | 86.187500 | 1.868750 | 158.625000 | 9.716667 | 85.458333 | 999.979167 | 0.079167 | 14.375000 | 9.787500 | 9.654167 | 84.625000 | 3.497917 | 174.458333 | 0.022917 |

Table 4. Meteorological samples grouped by day and features labelled dataframe (4.015 rows x 17 columns)

Despite feature units are explained in web, a summary is attached in Table 5 below.

| CODI_VARIABLE | NOM_VARIABLE | UNITAT |
|---|---|---|
| 72 | Precipitació màxima en 1 minut | mm |
| 3 | Humitat relativa màxima | % |
| 30 | Velocitat del vent a 10 m (esc.) | m/s |
| 31 | Direcció de vent 10 m (m. 1) | ° |
| 32 | Temperatura | °C |
| 33 | Humitat relativa | % |
| 34 | Pressió atmosfèrica | hPa |
| 35 | Precipitació | mm |
| 36 | Irradiància solar global | W/m² |
| 38 | Gruix de neu a terra | mm |
| 40 | Temperatura màxima | °C |
| 42 | Temperatura mínima | °C |
| 44 | Humitat relativa mínima | % |
| 46 | Velocitat del vent a 2 m (esc.) | m/s |
| 47 | Direcció del vent a 2 m (m. 1) | ° |
| 48 | Velocitat del vent a 6 m (esc.) | m/s |
| 49 | Direcció del vent a 6 m (m. 1) | ° |
| 50 | Ratxa màxima del vent a 10 m | m/s |
| 51 | Direcció de la ratxa màxima del vent a 10 m | ° |
| 53 | Ratxa màxima del vent a 6 m | m/s |
| 54 | Direcció de la ratxa màxima del vent a 6 m | ° |
| 56 | Ratxa màxima del vent a 2 m | m/s |
| 57 | Direcció de la ratxa màxima del vent a 2 m | ° |
| 59 | Irradiància neta | W/m² |
| 1 | Pressió atmosfèrica màxima | hPa |
| 2 | Pressió atmosfèrica mínima | hPa |

Table 5. Meteorologic features units

Dataset created was exported to csv ('Meteo_ready.csv') to be merged later with Air pollutant one.

### *3.1.3 Merge Air pollutant and Meteorological dataframes*

Once it was got both dataframes, let's merge together. To do so, as seen in Notebook 2. Merge_data_conta_meto.ipynb, it was merged Air pollutant in left side and Meteorological dataframe in right side using merge function. The key element to merge together was the date. In both dataframes, dates were modified accordingly in order to get the same format and later on allowed us to merge without issues. As Air pollutant dataframe has fewer rows, it was decided to consider it as preferent in merge operation. Table 6, highlights the resultant dataframe merged.

| | data | CO max | H2S max | NO max | NO2 max | NOx max | PM1 max | PM10 max | PM2.5 max | SO2 max | CO | H2S | NO | NO2 | NOx | PM1 | PM10 | PM2.5 | SO2 | data_lectura | Pressió atmosfèrica màxima | Pressió atmosfèrica mínima | Humitat relativa màxima | Velocitat del vent a 10 m (esc.) | Direcció de vent 10 m (m. 1) | Temperatura | Humitat relativa |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2010-01-01 | 0.2 | 1.4 | 1.0 | 7.0 | NaN | NaN | NaN | NaN | 1.0 | 0.200000 | 1.108333 | 1.000000 | 2.625000 | NaN | NaN | NaN | NaN | 1.000000 | 2010-01-01 | 1003.531915 | 1002.680851 | 45.191489 | 5.848936 | 267.234043 | 11.278723 | 44.170213 |
| 1 | 2010-01-02 | 0.2 | 1.8 | 13.0 | 44.0 | NaN | NaN | NaN | NaN | 1.0 | 0.200000 | 1.137500 | 2.250000 | 12.916667 | NaN | NaN | NaN | NaN | 1.000000 | 2010-01-02 | 1016.875000 | 1016.312500 | 57.020833 | 3.537500 | 220.875000 | 9.833333 | 55.625000 |
| 2 | 2010-01-03 | 0.3 | 1.6 | 17.0 | 48.0 | NaN | NaN | NaN | NaN | 7.0 | 0.204167 | 1.158333 | 3.625000 | 22.166667 | NaN | NaN | NaN | NaN | 1.875000 | 2010-01-03 | 1016.458333 | 1016.125000 | 79.250000 | 1.527083 | 101.083333 | 9.666667 | 78.000000 |
| 3 | 2010-01-04 | 0.2 | 2.0 | 26.0 | 42.0 | NaN | NaN | NaN | NaN | 4.0 | 0.200000 | 1.382609 | 8.217391 | 28.304348 | NaN | NaN | NaN | NaN | 1.652174 | 2010-01-04 | 1008.625000 | 1008.354167 | 82.583333 | 1.802083 | 81.458333 | 9.406250 | 81.625000 |
| 4 | 2010-01-05 | 0.4 | 1.4 | 23.0 | 44.0 | NaN | NaN | NaN | NaN | 2.0 | 0.220833 | 1.037500 | 5.750000 | 24.958333 | NaN | NaN | NaN | NaN | 1.083333 | 2010-01-05 | 1000.083333 | 999.833333 | 86.187500 | 1.868750 | 158.625000 | 9.716667 | 85.458333 |

Table 6. Air pollutant and meteorological dataframes merged. (3.992 rows x 36 columns)

Dataset created was exported to csv ('data_Conta_Meteo_merged.csv') to be considered as main dataset to work with and analyse under Data Science tools.

# Chapter 4 - Modelling

To go ahead with the dataset merged analysis, it was created a new feature in column labelled as Alert, as seen in 3. Modelling.ipynb Notebook. This Alert column is the result of having considered, based on WHO, $PM_{2.5}$ max value column exceeding threshold 25 $\mu g/m^3$. Moreover, as we want to predict or forecasting 24h later, it was just deleted one row and move up one row all Alert feature column, in order to get in today, tomorrow's $PM_{2.5}$ max value. With that, our target column will be Alert feature.

This is the reason behind it was selected Machine Learning Supervised Classification techniques.

## 4.1 Exploratory Data Analysis

### *4.1.1 Null values*

First exploration done was focused on identifying null values. Based on results found (Figure 1), there are many null values in CO and CO max, and $PM_1$ and $PM_1$ max features.

| | |
|---|---|
| CO max | 3649 |
| CO | 3649 |
| PM1 max | 1354 |
| PM1 | 1354 |
| PM2.5 | 926 |
| PM2.5 max | 926 |
| PM10 | 919 |
| PM10 max | 919 |
| NOx max | 669 |
| NOx | 669 |
| NO max | 40 |
| NO2 max | 40 |
| NO | 40 |
| NO2 | 40 |
| H2S max | 31 |
| H2S | 31 |
| SO2 | 17 |
| SO2 max | 17 |
| Temperatura | 5 |
| Temperatura mínima | 5 |
| Temperatura màxima | 5 |
| Humitat relativa | 4 |
| Humitat relativa mínima | 4 |
| Irradiància solar global | 4 |

Figure 1. Null values

The reason behind those missing values is seen in Figure 2 below:

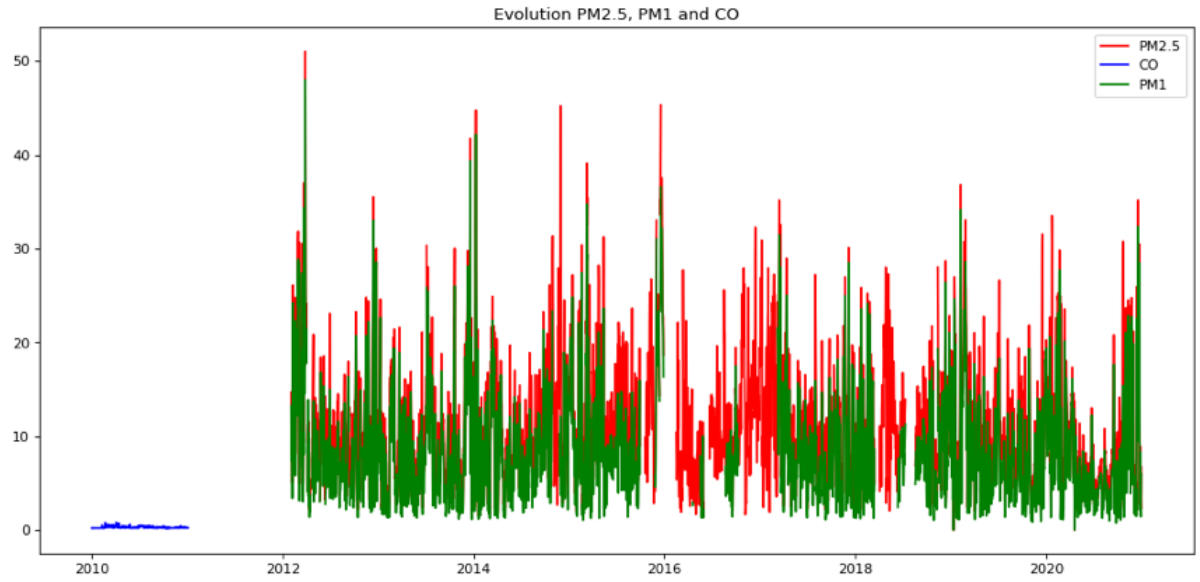



Figure 2. Evolution of CO, $PM_1$ and $PM_{2.5}$ features from 2010 to 2020

Graph highlights that CO registers finished by beginning 2011, $PM_1$ and $PM_1$ max. missing values corresponds to a period between 2016 and middle 2017, and first $PM_{2.5}$ registers dated since February 2012. With all of that, $PM_{2.5}$ is the main importance feature, so CO features will be removed, and as we unknown the root cause of $PM_1$ misssing values, considering 1,5 years (14%) as considerable missing data within 8 years database available, it will also be removed.

A new dataframe without CO and $PM_1$ polluntant features is seen below (Table 7).

| | data | H2S max | NO max | NO2 max | NOx max | PM10 max | PM2.5 max | SO2 max | H2S | NO | NO2 | NOx | PM10 | PM2.5 | SO2 | data_lectura | Pressió atmosfèrica màxima | Pre atmosfè mír |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **743** | 2012-02-03 | 1.9 | 12.0 | 56.0 | 75.0 | 29.0 | 20.0 | 1.0 | 1.371429 | 3.400000 | 17.900000 | 22.900000 | 26.555556 | 14.666667 | 1.000000 | 2012-02-03 | 1017.964583 | 1017.59 |
| **744** | 2012-02-04 | 1.4 | 7.0 | 37.0 | 46.0 | 59.0 | 14.0 | 2.0 | 1.291667 | 2.583333 | 12.666667 | 15.708333 | 19.666667 | 9.125000 | 1.041667 | 2012-02-04 | 1023.589583 | 1023.09 |
| **745** | 2012-02-05 | 2.4 | 21.0 | 64.0 | 95.0 | 30.0 | 24.0 | 4.0 | 1.595833 | 4.375000 | 22.416667 | 28.500000 | 15.291667 | 11.541667 | 1.166667 | 2012-02-05 | 1022.868750 | 1022.46 |
| **746** | 2012-02-06 | 2.1 | 14.0 | 60.0 | 81.0 | 24.0 | 23.0 | 1.0 | 1.604545 | 3.900000 | 20.800000 | 26.450000 | 13.333333 | 7.458333 | 1.000000 | 2012-02-06 | 1020.218750 | 1019.85 |
| **747** | 2012-02-07 | 1.8 | 5.0 | 42.0 | 48.0 | 41.0 | 9.0 | 1.0 | 1.445833 | 2.521739 | 13.521739 | 16.869565 | 17.541667 | 5.041667 | 1.000000 | 2012-02-07 | 1016.256250 | 1015.71 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3987** | 2020-12-27 | 3.1 | 13.0 | 44.0 | 63.0 | 28.0 | 24.0 | 5.0 | 1.408333 | 2.833333 | 13.625000 | 17.416667 | 10.458333 | 8.916667 | 1.500000 | 2020-12-27 | 1009.404167 | 1008.76 |
| **3988** | 2020-12-28 | 1.6 | 6.0 | 27.0 | 37.0 | 26.0 | 6.0 | 9.0 | 1.208333 | 1.541667 | 7.833333 | 9.666667 | 7.541667 | 3.083333 | 5.416667 | 2020-12-28 | 995.566667 | 994.95 |
| **3989** | 2020-12-29 | 1.5 | 5.0 | 15.0 | 16.0 | 14.0 | 9.0 | 4.0 | 1.150000 | 1.250000 | 6.791667 | 8.166667 | 4.708333 | 3.208333 | 2.333333 | 2020-12-29 | 1001.879167 | 1001.51 |
| **3990** | 2020-12-30 | 1.3 | 3.0 | 20.0 | 25.0 | 42.0 | 5.0 | 1.0 | 1.120833 | 1.208333 | 5.458333 | 6.916667 | 7.500000 | 2.250000 | 1.000000 | 2020-12-30 | 1010.639583 | 1010.08 |
| **3991** | 2020-12-31 | 1.5 | 7.0 | 40.0 | 51.0 | 27.0 | 22.0 | 1.0 | 1.175000 | 1.500000 | 10.791667 | 12.583333 | 8.916667 | 6.708333 | 1.000000 | 2020-12-31 | 1011.370833 | 1011.06 |

3063 rows × 33 columns

Table 7. Dataframe with $PM_{2.5}$>0

The last step is to treat NAN values, to do so we have two strategies, to drop samples or to fill with back or forward data available. Before applying any method, it was evaluated the data lost impact by dropping or filling. In Figure 3, it is highlighted that dropping samples it was lost around 20% of the Alert feature data, so as our dataset is unbalanced, it was prioritized to fill data instead of dropping it.

```
# NA data strategies to be managed. Drop or fill.
dfA = df0.dropna()
dfB = df0.fillna(method = 'ffill')
```

```
print(dfA['Alert'].value_counts())
print(dfB['Alert'].value_counts())
```

```
0    2122
1     856
Name: Alert, dtype: int64
0    2191
1     872
Name: Alert, dtype: int64
```

Figure 3. Comparison between dropping or filling NAN values. If dropping is selected, it will be lost around 20% of the Alert feature info. Essential info in our unbalanced dataset.

It was used forward method to fill NANs in dataset because backward filling NAN's in last raw cannot be filled (become NAN when Alert feature 24h was created), unless we increase data window (not the case).

Once Dataset is completed, it was analysed the distribution of Quartiles that divide data samples into four equal parts, and help us to quickly assess the spread and central tendency of our dataset, as seen in Table 8, Quartiles are good distributed and regular so no estrange values were highlighted.

| | H2S max | NO max | NO2 max | NOx max | PM10 max | PM2.5 max | SO2 max | H2S | NO | N( |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 3063.000000 | 3063.000000 | 3063.000000 | 3063.000000 | 3063.000000 | 3063.00000 | 3063.000000 | 3063.000000 | 3063.000000 | 3063.0000 |
| mean | 2.815540 | 20.856676 | 47.199478 | 75.781587 | 35.038198 | 20.32713 | 5.597453 | 1.601034 | 4.917049 | 20.1526 |
| std | 1.781291 | 24.134056 | 22.620814 | 51.230926 | 19.259940 | 12.02200 | 6.754843 | 0.641040 | 5.132810 | 9.8817 |
| min | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 4.000000 | 2.00000 | 0.000000 | 0.795238 | 0.000000 | 0.0000 |
| 25% | 1.700000 | 5.000000 | 32.000000 | 41.000000 | 23.000000 | 12.00000 | 1.000000 | 1.158333 | 1.916667 | 12.7445 |
| 50% | 2.400000 | 13.000000 | 45.000000 | 63.000000 | 32.000000 | 18.00000 | 4.000000 | 1.425000 | 3.041667 | 18.6666 |
| 75% | 3.400000 | 27.000000 | 60.000000 | 98.000000 | 43.000000 | 26.00000 | 7.000000 | 1.816667 | 5.708333 | 26.3333 |
| max | 28.300000 | 200.000000 | 396.000000 | 421.000000 | 263.000000 | 141.00000 | 144.000000 | 6.637500 | 48.708333 | 61.8888 |

Table 8. Quartile distribution

### *4.1.2 Data Distribution*

Let's evaluate feature distribution importing the libraries accordingly as seen in Figure 4. A list of Libraries and packages imported are summarized in Appendix.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Figure 4. Libraries to evaluate feature distribution

Once libraries were imported, it was selected Density plots, Box plots and Correlation matrix with the aim of evaluating distribution, possible outliers and correlation between features accordingly. In Figure 5, it is represented in Density plots where the vast majority of features follow a symmetric normal distribution, others like some pollutants have a right or left bias and others, like Temperature, Wind and Solar irradiation follow a 2 peaks distribution bimodal. In any case, the overall view presents a regular and well distributed data with no relevant information to consider.

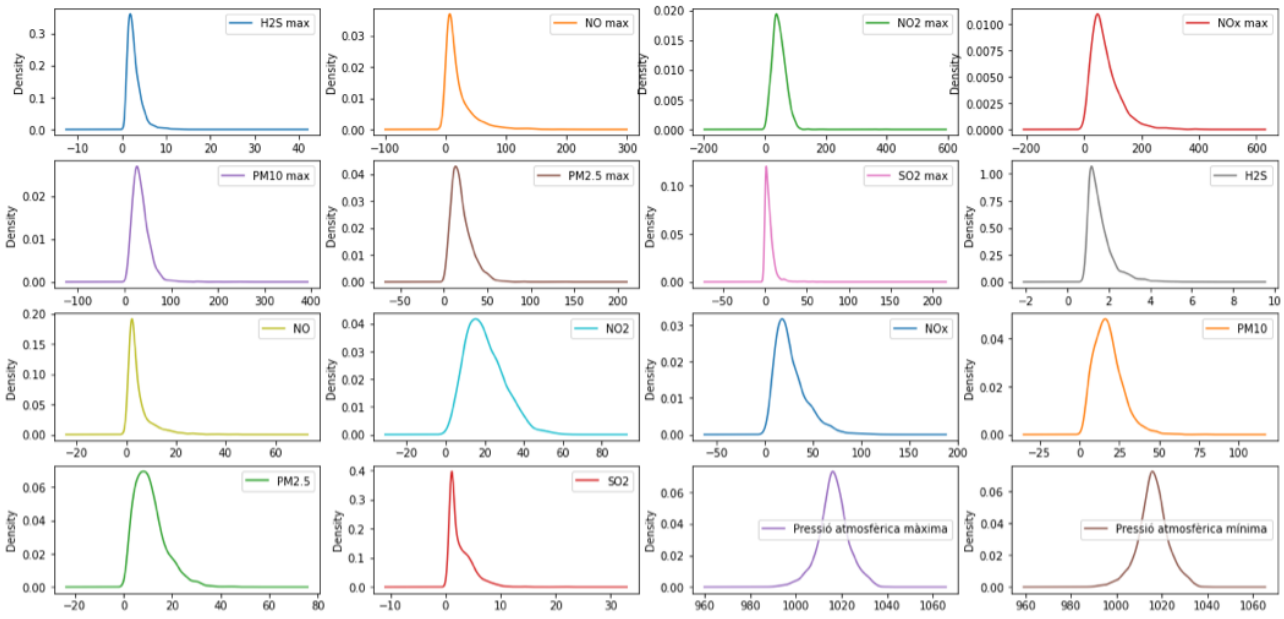



Figure 5. A sample of Density plot distribution

As a second graph, Box plot was selected to evaluate possible outliers. Based on boxplots got, Figure 6, it seems there are many outliers, so later on we will evaluate the impact on our Dataset results.

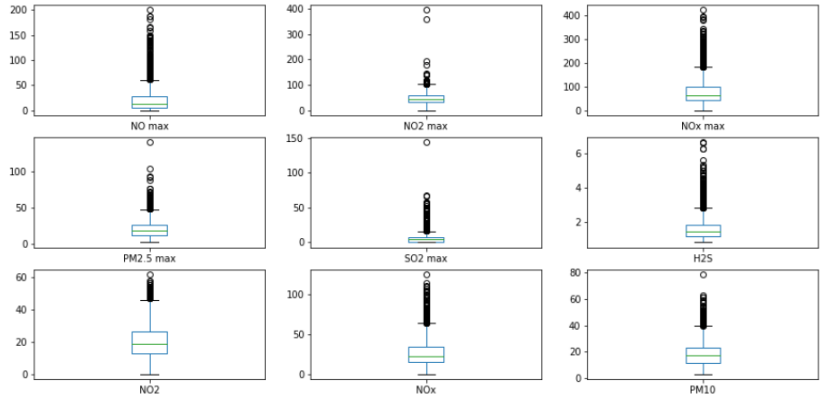



Figure 6. Representative sample of box plots.

Third data analysis visualization was correlation matrix, it identifies features strongly correlated which can create problems when we launched our models. So, it is extremely important to analyse case by case and remove them if needed.

In our matrix, there are features extremely correlated, like weather features with maximum, average and minimum values, so probably it was just kept one of three. Same issue was found in pollutant features, where maximum and average values are correlated, too.

In Figure 7, it can be seen, features correlated with our target Alert. In first view, Alert is correlated with $PM_{2.5}$, other pollutants and few meteorological features. Anyhow, all features will be considered, but once launched models, feature reduction techniques will be applied and based on results, we will decide to keep it or remove it.


Figure 7. Correlation matrix between features and target Alert

Last visualization corresponds to Distribution graph between $PM_{2.5}$ max (Target) against other features. As seen on Figure 8, it is interesting to extract that some pollutant, as $NO_2$, $PM_{2.5}$, $PM_{10}$, and some Meteorological features, as Wind speed, Pressure atmosphere and temperature have a sensitive relation (red slope) with the $PM_{2.5}$ max feature.


Figure 8. A sample of distribution graph between $PM_{2.5}$ and other features

As an overall result got on this section, let's summarize that first strategy will be to consider all features as relevant on modelling phase, but later on they will be compared with feature selection methods results. Moreover, some pollutants and weather features are correlated (positive or negative) with target.

## 4.2 Outliers

As noticed in Boxplots graphs, there are possible outliers that need to be analysed. To do so, let's import pyod and sklearn libraries. Method used is KNN Detector[13].

For an observation, its distance to its nearest neighbour could be viewed as the outlying score (a way to measure the density). Three kNN detectors are supported: largest which uses the distance to the kNN neighbour as the outlier score, mean and median. Probably, the most important parameter to define is the "contamination" (between 0-0.5) argument, needed to estimate number of outliers. In this case, it was set at 0.18.

Based on results got, there are 477 samples suspected to be outliers, under 18% of data contaminated parameter set. As seen in Figure 9, if outliers were removed, we would lose around 30% of our Alert target data, something very sensitive due to our unbalanced dataset. Despite that, in our Base Model we will test both cases.

```
#Let's compare how many Alert = 1 values we lose if outliers were removed
print(df0['Alert'].value_counts())
print(df0[y_pred!=1]['Alert'].value_counts())
```

```
0    2191
1     872
Name: Alert, dtype: int64
0    1956
1     630
Name: Alert, dtype: int64
```

Figure 9. Comparison between keeping or removing outliers against Alert data target feature

## 4.3 Classification Metrics

Metrics used for our results were split into 2 functions, full evaluate function with F1, Recall, Accuracy, Precision and Roc Auc scores, and Confusion matrix. All of them used to evaluate different models, tuning and oversampling methods. A partial function with F1 and Accuracy scores will be used for the rest of analyses. As Alert System desired to develop, it will be interested to evaluate False Positives and minimize as much as possible False Negatives, so F1 and Recall scores, and Confusion matrix metrics will help to do so.

13 https://pyod.readthedocs.io/en/latest/pyod.models.html#id68

## 4.4 Base Model

Base model will help to evaluate outliers' impact by keeping or removing. To do so, it was used sklearn libraries, split dataset into train and test with 80%-20% accordingly, and simple Logistic Regression method to compare results. As note to be considered, is that for small dataset, as we have, and unique target feature, it is better to use liblinear solver method in Logistic Regression.

As seen in Table 9, removing outliers got worst results than keeping it, in Train and Test results, so let's keep outliers.

| | with Outliers | | w/o Outliers | |
|---|---|---|---|---|
| | Train | Test | Train | Test |
| F1 score | 0,62 | 0,65 | 0,55 | 0,58 |
| Accuracy score | 0,81 | 0,83 | 0,83 | 0,83 |

Table 9. Comparison results with and without (w/o) outliers

## 4.5 Pre-Processing Data

Once a base model is defined, let's compare different pre-processing data techniques (Standardize, Rescale and Normalize) and evaluate which one provides a better result in comparison with no data pre-processed. To do that, additionally to import different pre-processing libraries, it will be imported a pipeline library to use pipelines with the aim of to speed up the operations. As seen on Figure 10, pipeline was applied.

```
steps = [('scaler', StandardScaler()),('lgr',LogisticRegression(solver='liblinear'))]
pipe_lr = Pipeline(steps)

pipe_lr.fit(X_train, y_train)
train_predictions = pipe_lr.predict(X_train)
test_predictions = pipe_lr.predict(X_test)

print("TRAIN:")
print(f1_score(y_train, train_predictions),accuracy_score(y_train, train_predictions))

print("TEST:")
print(f1_score(y_test, test_predictions),accuracy_score(y_test, test_predictions))
```

```
TRAIN:
0.6570048309178744 0.8261224489795919
TEST:
0.6148648648648649 0.8140293637846656
```

Figure 10. Pipeline applied to Standardize pre-processing method

On Table 10 below, it was compared pre-processing methods results in Train and Test datasets. Just to remark, that it is important to evaluate both datasets (train and test) in order to understand if overfitting is becoming. Similar results between train and test suggest that we do not have overfitting.

| | No preprocessing | | Standardize | | Rescale | | Normalize | |
|---|---|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test | Train | Test |
| F1 score | 0,62 | 0,65 | 0,64 | 0,67 | 0,62 | 0,65 | 0,14 | 0,17 |
| Accuracy score | 0,81 | 0,83 | 0,82 | 0,84 | 0,81 | 0,83 | 0,73 | 0,74 |

Table 10. Pre-processing methods results

Based on results got, Standardize pre-processing method got slightly better results than the rest. Not much better, but enough to go ahead with it.

## 4.6 Feature Importance

On this section, let's compare results got with PCA and Feature Importance methods in order to simplify our dataset and help us to develop FrontEnd in introduction data tasks. Once standardized our dataset, let's import libraries for PCA and Feature Importance (Figure 11).

```
from sklearn.decomposition import PCA
from sklearn.ensemble import ExtraTreesClassifier
```

Figure 11. Libraries for PCA and Feature Importance

As a result of PCA, with 8 components it is possible to explain more than 85% of variance (Figure 12).

```
shape of X_pca (2450, 8)
[0.31168955 0.17016025 0.11120792 0.07689592 0.06508444 0.05253912
 0.04746813 0.04283244]
suma: 0.8778777558992626
```

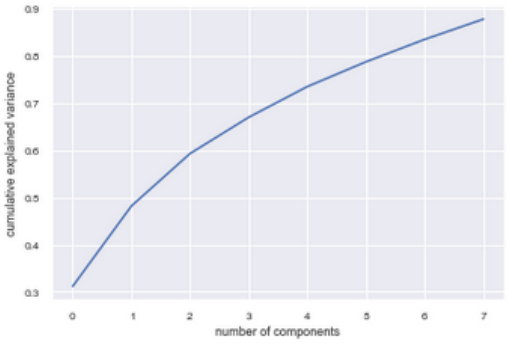



Figure 12. PCA 8 components explain 88% of variance

On the other hand, Feature Importance method, in combination with Data distribution visualization done (4.1.2 section), allowed us to conclude that $PM_{2.5}$, $PM_{10}$,$NO_2$, Atmosphere pressure, Temperature and Wind speed are the most important features to be chosen in order to simplify dataset (Figure 13).

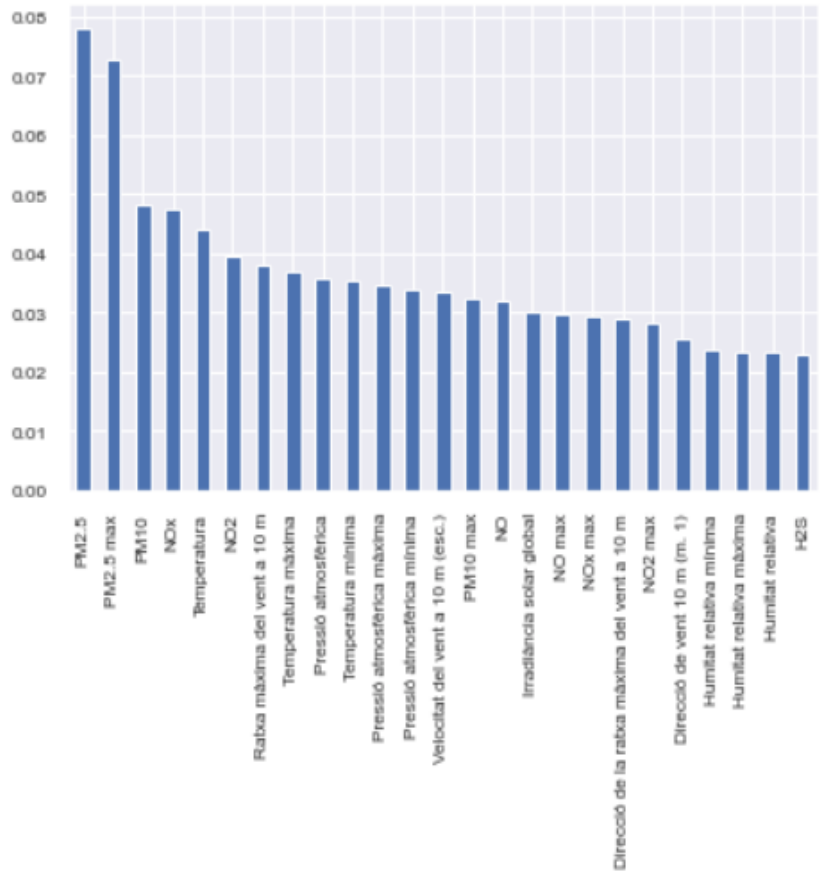



Figure 13. $PM_{2.5}$, $PM_{10}$, $NO_2$, atmosphere pressure, temperature and wind speed are the most importance features

Table 11 results confirms our expectations.

| | No method | | PCA | | Feature Importance | |
|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test |
| F1 score | 0,64 | 0,67 | 0,63 | 0,64 | 0,61 | 0,67 |
| Accuracy score | 0,82 | 0,84 | 0,81 | 0,83 | 0,82 | 0,82 |

Table 11. Feature importance methods results

As highlighted in Table 11 above, Feature importance method got same results as PCA, and both, similar results to no method. As a FrontEnd user, in manual mode, it will

be needed to impute data manually to Predict, so it is preferable to simplify dataframe from 36 to 6 features selected by selecting Feature importance method. Based on that, let's create a new dataframe with 6 features and target, as seen in Figure 14 below.

| | PM2.5 | PM10 | NO2 | Pressió atmosfèrica | Temperatura | Velocitat del vent a 10 m (esc.) | Alert |
|---|---|---|---|---|---|---|---|
| 0 | 14.666667 | 26.555556 | 17.900000 | 1017.787500 | 2.714583 | 5.052083 | 0 |
| 1 | 9.125000 | 19.666667 | 12.666667 | 1023.345833 | 4.029167 | 4.885417 | 0 |
| 2 | 11.541667 | 15.291667 | 22.416667 | 1022.672917 | 3.562500 | 2.631250 | 0 |
| 3 | 7.458333 | 13.333333 | 20.800000 | 1020.043750 | 7.995833 | 3.593750 | 0 |
| 4 | 5.041667 | 17.541667 | 13.521739 | 1016.004167 | 8.281250 | 5.885417 | 0 |

Figure 14. Dataset with selected features ready for launching models

## 4.7 Model Selection

To run our dataframe it was selected Classification models: Logistic Regression (LR), K-Neighbours Classifier (KNC), Decision Tree Classifier (DTC), Support Vectors Classifier (SVC), Random Forest Classifier (RFC) and XG-Boost Classifier (XGB).

Additionally, the dataset was standardized, and it was applied Repeated K-fold cross validation method, in order to compare and select machine learning models.

On summarized Table 12 below, it is noticed that SVC provided similar results than LR but better results than the rest, despite that, it was selected SVC as method to launch final model, because of its slightly better Recall score and less False Negative results got.

| | LR | | KNC | | DTC | | SVC | | RFC | | XGB | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test | Train | Test | Train | Test | Train | Test |
| F1 score | 0,65 | 0,72 | 0,61 | 0,64 | 0,53 | 0,52 | 0,66 | 0,72 | 0,6 | 0,6 | 0,59 | 0,61 |
| Recall score | - | 0,78 | - | 0,6 | - | 0,51 | - | 0,79 | - | 0,52 | - | 0,55 |
| Accuracy score | | 0,81 | | 0,8 | | 0,71 | | 0,81 | | 0,79 | | 0,78 |
| Precision score | | 0,66 | | 0,69 | | 0,54 | | 0,65 | | 0,72 | | 0,69 |
| Roc auc score | | 0,8 | | 0,74 | | 0,66 | | 0,8 | | 0,71 | | 0,72 |
| Confusion matrix | | | | | | | | | | | | |
| TP FP | 348 | 77 | 375 | 50 | 343 | 82 | 346 | 79 | 386 | 39 | 378 | 47 |
| FN TN | 40 | 148 | 75 | 113 | 93 | 95 | 39 | 149 | 90 | 98 | 85 | 103 |

Table 12. Model selection results

## 4.8 Tuning model

Once pre-processing standardized method, reduced features and SVC model are selected, let's try to tune hyperparameters to understand if our metric results can be improved or not. To do so, it will be introduced GridSearchCV library as seen in Figure 15.

```
from sklearn.model_selection import GridSearchCV
```

Figure 15. GridSearch library

As seen on Figure 16, it will check parameters grid C, gamma and kernel. As unbalanced dataset we have, we also included class weight balanced parameter.

```
# defining parameter range
param_grid = {'C':[0.1,1,10,100],'gamma': [1,0.1,0.01,'scale'], 'kernel':['rbf','linear']}

svc = SVC(class_weight='balanced')


grid = GridSearchCV(
        estimator  = svc,
        param_grid = param_grid,
        scoring    = 'f1')
```

Figure 16, SVC hyperparameters to gridsearch

Table 13 results below confirms that parameters set as default by model in comparison 4.7 section are the tuned ones, so we got same results.

| | SVC | | SVC (tuned) | |
|---|---|---|---|---|
| | Train | Test | Train | Test |
| F1 score | 0,66 | 0,72 | | 0,72 |
| Recall score | | 0,79 | | 0,79 |
| Accuracy score | - | 0,81 | - | 0,81 |
| Precision score | | 0,65 | | 0,65 |
| Roc auc score | | 0,8 | | 0,8 |
| Confusion matrix | | | | |
| TP FP | 346 | 79 | 346 | 79 |
| FN TN | 39 | 149 | 39 | 149 |

```
SVC(C=1, class_weight='balanced')
(0.7163461538461539,
 0.7925531914893617,
 0.8075040783034257,
 0.6535087719298246,
 0.8033354192740925,
 array([[346,  79],
        [ 39, 149]]))
```

Table 13, SVC comparison with and w/o hyperparameters tuned

## 4.9 Oversampling

Once decided SVC tuned as Machine Learning method, let's try to evaluate the convenience of oversampling, in order to solve unbalanced issue, we have in our dataset. Techniques to be tested will be SMOTE, Oversampling and Undersampling.

- Oversampling: replicates the minority class samples.
- Undersampling: delete samples from the majority class.
- SMOTE: oversampling technique where the synthetic samples are generated for the minority class.

In all cases, we created a pipeline to introduce pre-processing and the model, as seen in Figure 17.

```
from imblearn.over_sampling import RandomOverSampler
oversample = RandomOverSampler(sampling_strategy='minority')
X_train_ove, y_train_ove = oversample.fit_resample(X_train, y_train)
```

```
pipe_svc.fit(X_train_ove, y_train_ove)
train_predictions_svc = pipe_svc.predict(X_train_ove)
test_predictions_svc = pipe_svc.predict(X_test)

full_evaluate(y_test, test_predictions_svc)
```

```
(0.7177033492822966,
 0.7978723404255319,
 0.8075040783034257,
 0.6521739130434783,
 0.8048185231539424,
 array([[345,  80],
        [ 38, 150]]))
```

Figure 17. Oversampling method tested in pipeline

Table 14. below, shows results between different methods used.

| | No method | | SMOTE | | Oversampling | | Undersampling | |
|---|---|---|---|---|---|---|---|---|
| | Test | | Test | | Test | | Test | |
| F1 score | 0,72 | | 0,71 | | 0,72 | | 0,72 | |
| Recall score | 0,79 | | 0,78 | | 0,8 | | 0,78 | |
| Accuracy score | 0,81 | | 0,8 | | 0,81 | | 0,81 | |
| Precision score | 0,65 | | 0,65 | | 0,65 | | 0,66 | |
| Roc auc score | 0,8 | | 0,8 | | 0,8 | | 0,8 | |
| Confusion matrix | | | | | | | | |
| TP FP | 346 | 79 | 346 | 79 | 345 | 80 | 349 | 76 |
| FN TN | 39 | 149 | 42 | 146 | 38 | 150 | 41 | 147 |

Table 14. Metrics results summary after oversampling dataset

As seen above, all oversampling methods used do not provide big differences between them. Despite that, Oversampling method provided slightly better results than others, in one hand, by F1 and Recall score, and by the other hand, by less False Negatives values got, convenient for our safety Alert System's purpose.

Once all Jupyter Notebook is run, we import Pickle library to save the model in order to be launched within FrontEnd (Figure 18).

```
import pickle
```

```
#Let's save the model into FrontEnd folder
pickle_out = open("FrontEnd_docs_related/aplication.pkl", "wb")
pickle.dump(pipe_svc, pickle_out)
pickle_out.close()
```

Figure 18. Import pickle library and model saved

# Chapter 5 - Summary and Results

As summary and overall results, we got that for the period of time between 2010 and 2020 in Tarragona (Bonavista) area, and selecting $PM_{2.5}$ as feature to be predicted, it was got results below (Figure 19) having simplified dataset from 36 to 7 features (Particles: $PM_{2.5}$ and $PM_{10}$, $NO_2$ pollutant, meteorological features as atmosphere pressure, temperature and wind speed, and target), applying Feature importance, Standardizing and Oversampling methods in dataset, for selecting SVC tuned and balanced machine learning method in a pipeline.

As overall conclusion, our best results were got by:

- Simplifying dataset to 7 features
- SVC method (parameters tuned)
- Class_weight balanced
- Standardized data
- Oversampling method

```
steps = [('scaler', StandardScaler()),
         ('SVC', SVC(class_weight='balanced'))]

pipe_svc = Pipeline(steps)

pipe_svc.fit(X_train_ove, y_train_ove)
train_predictions_svc = pipe_svc.predict( X_train_ove)
test_predictions_svc = pipe_svc.predict(X_test)


full_evaluate(y_test, test_predictions_svc)
```

```
(0.7177033492822966,
 0.797872340425531,
 0.8075040783034257,
 0.6521739130434783,
 0.8048185231539424,
 array([[345,  80],
        [ 38, 150]]))
```

Figure 19 Final results achieved

If we compare final results got (Figure 19), and ones got without any methodology applied (Table 9), it is highlighted a big improvement on results, reflected through all good activity and job done, in order to get the best model optimized (Table 15).

| | No treatment | Treatment |
|---|---|---|
| | Test | Test |
| F1 score | 0,58 | 0,72 |
| Recall score | - | 0,8 |
| Accuracy score | 0,83 | 0,81 |
| Precision score | - | 0,65 |
| Roc auc score | - | 0,8 |
| TP FP | - | 345 / 80 |
| FN TN | - | 38 / 150 |

Table 15 Comparison between Beginning (No treatment) and Final results achieved

# Chapter 6 - Conclusions

At the beginning of the project, the objective was to develop a Model capable to predict 24h forecast of pollutant $PM_{2.5}$ with a reasonable and acceptable metric results, despite we worked with an unbalanced dataset, so I can say that this objective was achieved, considering that existing more accurate Pollutant Alert Systems use to work with different resources mixed as evolution of Sahara's winds trends, satellite information and historical databases.

As a consequence objective, second one, and influenced by some decisions taken, like simplifying dataset features when similar results were got against all features, a FrontEnd development was a big challenge for me, because of my non-prior programming knowledge before this master.

In both objectives, I am really satisfied with the job done, results and the positive experience got.

For the next steps, I may suggest extending this $PM_{2.5}$ Alert System to cities which already have detectors, and promote to other city halls without, to invest on $PM_{2.5}$ detectors, because of the biggest impact particles $PM_{2.5}$ have, and it will have. In the near future, based on WHO recommendations, its control will be a must.

# Chapter 7 - FrontEnd Guide

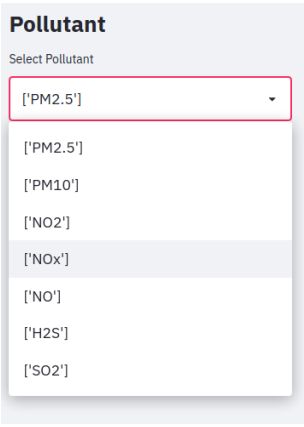
To run the FrontEnd, FrontEnd.ipynb jupyter notebook should be launched. All packages and libraries needed are noticed inside, so just run the code. The model is saved in FrontEnd_docs_related folder (application.pkl) as well as other docs needed that will be imported from the notebook directly. Once code is run, an app.py file will be downloaded and a new browser window will be opened. Check and follow the instructions noticed in application.

Pollution System Alert app is split in 2 sections, upper section where user can select Pollutant desired in left bar-side box, and Evolution of last years Pollutant graph and explanation will be showed. Below section is focused on Predict $PM_{2.5}$ Pollutant. To do so, just select impute method in left bar side box, Automatic or Manual. Automatic will take feature information by downloading from the web, and Manually mode, user should impute the data. In both cases, user should click on Predict box, and you will get the result, Alert System Activated or not. In case of activation, some countermeasures to citizens and city halls are suggested based on WHO recommendations.

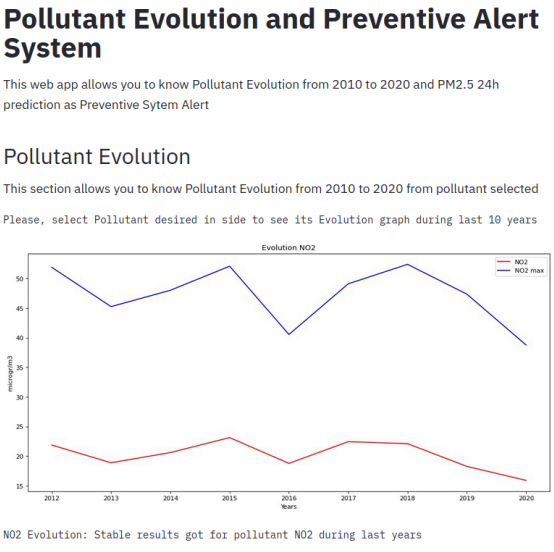
All information got in this app, it related to Tarragona (Bonavista) area.

Below I will show visual step instructions to follow:

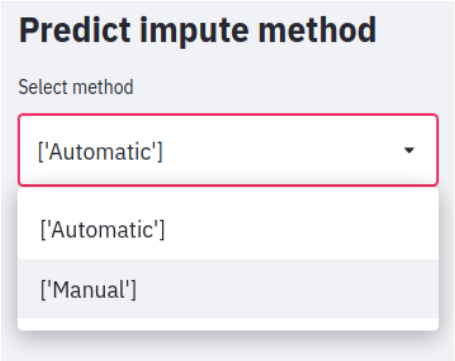
1. Select Pollutant



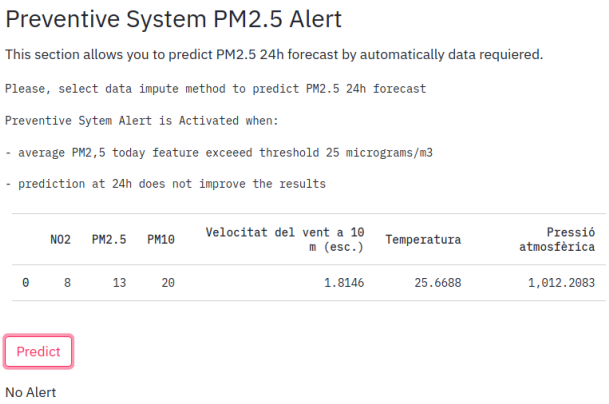

2. Check graph evolution and explanation



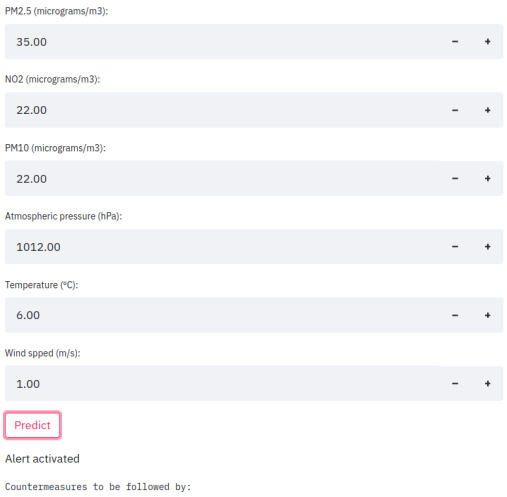
3. Select impute method




4. Automatic selection




5. Manual selection: impute parameters




6. Click on Predict to get result, in case of Alert activated, some countermeasures are highlighted

Alert activated

Countermeasures to be followed by:

- Citizen awareness:

-To reduce displacements with a private vehicle (use Public transport, trips on foot or by bicycle)

- Municipalities city hall:

-Local media diffusion campaigns to explain the warning situation

-Do not allow the burning of vegetation and enhance the management of plant residues, such as crushing or collection for its composting

-Suspended construction work

- Industry:

-Do not perform processes such as start up or set up not indispensable, if they can be delayed

# Appendix Packages and Libraries

Below it will be highlighted Packages installed and Libraries imported:

**Packages installed:**

- pip install sodapy
- pip install beautifulsoup4
- pip install lxml
- pip install requests

**Libraries imported:**

- import os
- import pandas as pd
- import numpy as np
- from sodapy import Socrata
- import datetime
- import operator
- import matplotlib.pyplot as plt
- from matplotlib.pyplot import figure
- import seaborn as sns
- from pyod.models.knn import KNN
- from sklearn import model_selection
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import f1_score, recall_score, accuracy_score, confusion_matrix, precision_score, roc_auc_score, classification_report
- from sklearn.preprocessing import MinMaxScaler
- from sklearn.preprocessing import Normalizer
- from sklearn.preprocessing import StandardScaler
- from sklearn.pipeline import Pipeline
- from sklearn.decomposition import PCA
- from sklearn.ensemble import ExtraTreesClassifier
- import warnings
- from warnings import simplefilter
- from sklearn.model_selection import RepeatedKFold
- from sklearn.linear_model import LogisticRegression
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.svm import SVC
- from sklearn.ensemble import RandomForestClassifier
- from xgboost import XGBClassifier
- from sklearn.model_selection import GridSearchCV
- from imblearn.combine import SMOTETomek
- from imblearn.over_sampling import RandomOverSampler
- from imblearn.under_sampling import RandomUnderSampler
- import pickle
- from datetime import datetime
- from datetime import timedelta
- import pickle
- import streamlit as st
- from PIL import Image
- import requests
- from bs4 import BeautifulSoup
- import re