Data Analytics

ARIF
Acute Respiratory Infections Forecast

June, 2023

IronHacker : Romain Courtois

Table of content

Introduction

Acute respiratory infections (ARI)

Acute respiratory infection is a serious infection that prevents normal breathing function. It usually begins as a viral infection in the nose, trachea (windpipe), or lungs. If the infection is not treated, it can spread to the entire respiratory system. Acute respiratory infection prevents the body from getting oxygen and can result in death.

Acute respiratory infections are infectious, which means they can spread from one person to another. According to the World Health Organization (WHO), acute respiratory infections kill an estimated 2.6 million children annually every year worldwide. According to Santé publique France : 40,000 deaths in Europe per year and nearly 8 months of life expectancy lost due to exposure to fine particles

ARIs are caused by various respiratory viruses including SARS-CoV-2 (Covid-19), influenza viruses and other respiratory viruses such as RSV, rhinovirus, or metapneumovirus. The purpose of ARI surveillance is to monitor epidemics caused by these viruses.

Some factors seem to favor the occurrence of such pathologies:

- Male gender;
- Age (the risk of death is higher in infants aged 1 to 3 months);
- Prematurity;
- **Climate and season** (infections mainly develop in cold and rainy weather);
- **Pollution**;
- Overcrowding;
- Nutritional status;
- Immunological status;
- Low level of education;
- Low socioeconomic level of the country.

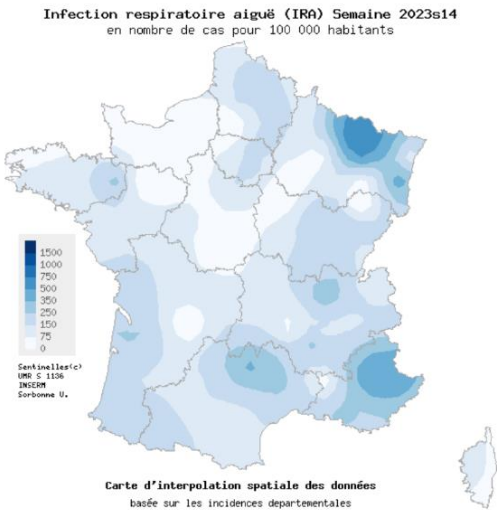
The Main goal of this project is to create a tool able to predict forecast of ARI incidence using weather forecast and pollution data.
Alternate goal: Perform an analysis of correlation between ARIs, weather and pollution.

1. Datasources

3 dataset will be used for this experimentation:

- ARI for the incidence of ARI per region,
- POLU for the pollutants measured
- SYNOP for the weather data.

1.1 ARI Dataset: French Weekly epidemiological surveillance



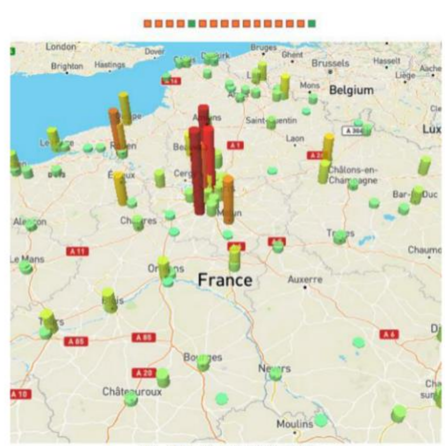
The Sentinelles network (INSERM/Sorbonne University, https://www.sentiweb.fr) collects a set of data allowing the epidemiological progress of certain diseases to be monitored with a weekly frequency.

opendatasoft

1.2 POLU_Dataset : "Real-time" data from measurements of concentrations of regulated air pollutants

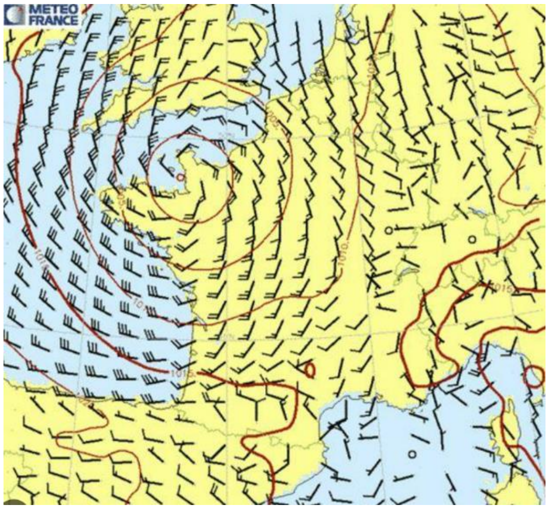
Hourly data from automatic analyzers : The concentrations of the following atmospheric pollutants are measured:

Ozone (O_3)
Nitrogen dioxide (NO_2)
Sulfur dioxide (SO_2)
Particles with a diameter of less than 10 µm (PM10)
Particles with a diameter of less than 2.5 µm (PM2.5)
Carbon monoxide (CO)



RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité
data.gouv.fr

1.3 SYNOP_Dataset : Historical meteorological observation France



Observation data from international surface observation reports (SYNOP) for the World Meteorological Organization (WMO).

Atmospheric parameters :
- measured (temperature, humidity, wind direction and force, atmospheric pressure, amount of precipitation)
- observed (weather sensitive, description of clouds, visibility) from the earth's surface.

2 Data base type selection

Data Model: SQL or NoSQL. Strict data integrity is not required. A flexible schema is not useful for the moment.

For this study, most data are imported in .csv format, SQL relational database will be used.
For a live predicting tool, based on scraped data on the web, a JSON storage will be implemented .

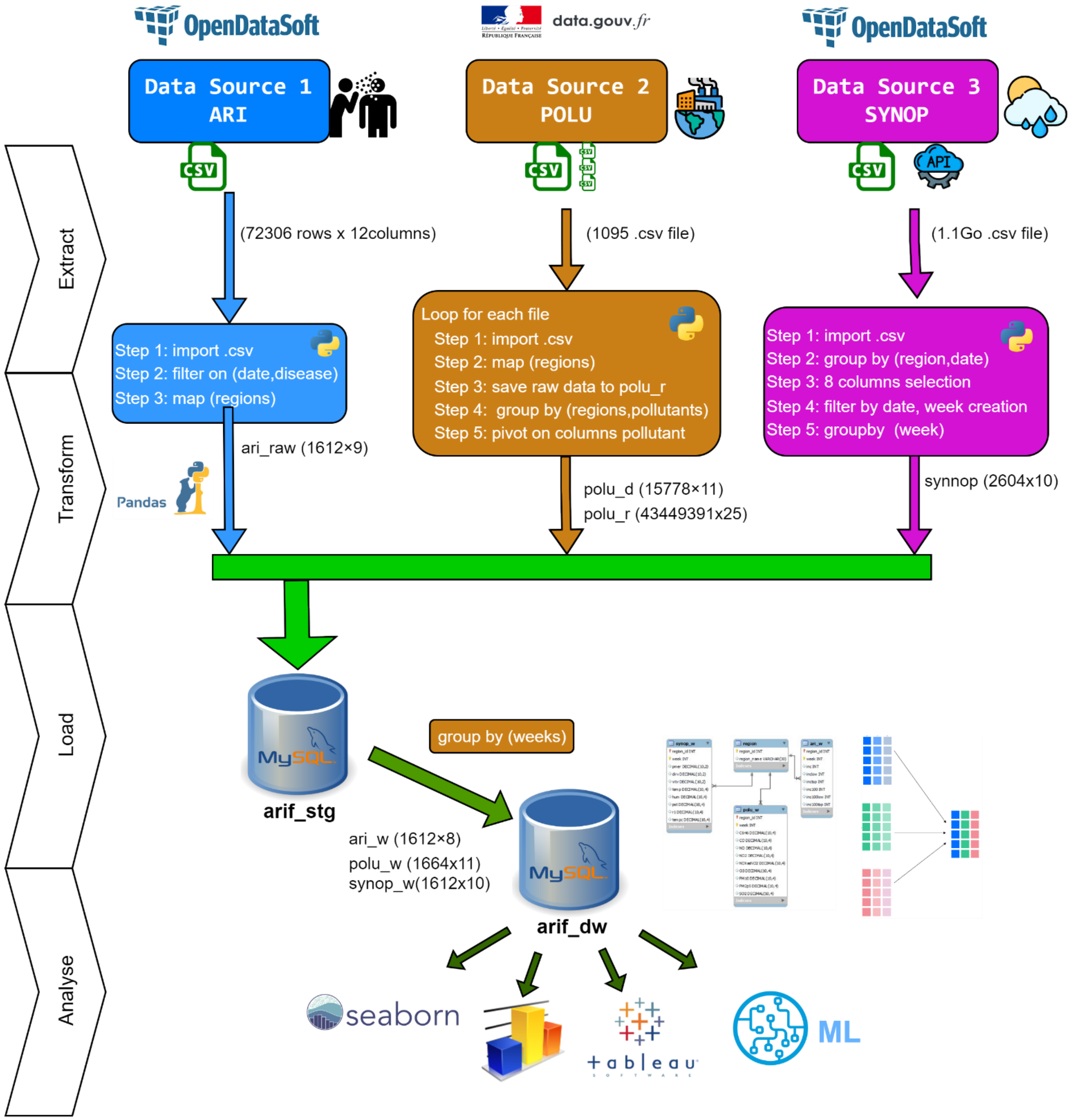
Performance: Data are loaded in a python environment for ML. Low performance accepted.

Data Relationships: Required for course certification

Cost , Community and Support: MySQL is free Database Management System, cross-platform, ideal for open source project.

A first staging **arif_stg** database with no relationship will receive raw and aggregated data from all sources.
A second **arif_dw** database with prepared data will be the warehouse for analytics.

3 Extract Transform Load Process



4 Data Extraction

4.1 ARI Dataset Extraction

Downloaded : healthref-france-sentinelles-weekly.csv 5.4 Mo 72306 rows 12 columns

Semaine	Code malac	Incidence esti	Borne inférieur d	Borne supérieu	Taux d'incidenc	Borne inférieu	Borne supéri	Zone géogr	Nom de la zone géographique	Date	Nom maladie
201139	6	1008	0	2094	28	0	58	52	PAYS-DE-LA-LOIRE	26/09/2011	Diarrhée aigüe
201136	6	8354	3935	12773	71	33	109	11	ILE-DE-FRANCE	05/09/2011	Diarrhée aigüe
201135	6	3184	546	5822	27	5	49	11	ILE-DE-FRANCE	29/08/2011	Diarrhée aigüe
201135	6	8648	3372	13924	143	56	230	32	HAUTS-DE-FRANCE	29/08/2011	Diarrhée aigüe
201132	6	4719	1494	7944	95	30	160	93	PROVENCE-ALPES-COTE-D-AZUR	08/08/2011	Diarrhée aigüe
201127	6	5041	0	13876	150	0	413	28	NORMANDIE	04/07/2011	Diarrhée aigüe
201126	6	1857	181	3533	52	5	99	52	PAYS-DE-LA-LOIRE	27/06/2011	Diarrhée aigüe
201125	6	4155	0	8345	160	0	321	24	CENTRE-VAL-DE-LOIRE	20/06/2011	Diarrhée aigüe
201123	6	501	159	843	165	53	277	94	CORSE	06/06/2011	Diarrhée aigüe
201115	6	933	0	1989	26	0	55	52	PAYS-DE-LA-LOIRE	11/04/2011	Diarrhée aigüe
201114	6	9196	5813	12579	121	76	166	84	AUVERGNE-RHONE-ALPES	04/04/2011	Diarrhée aigüe

Exploration : For Acute Respiratory Infection : from 23/03/2020 to 08/05/2023 1612 rows
1 line of data per region per week : This set my geographical granularity to French regions, and my time granularity to weeks. During integration, other dataset aim to be aggregated by region and weeks to reduce the size manipulated.

ARI Dataset Extract:

Python notebook : ARI_import_1.ipynb

Step 1: Import .csv file

```
ARI = pd.read_csv(r'healthref-france-sentinelles-weekly.csv',sep=';')
```

Step 2: filter on (date,disease)

```
ARI_25 = ARI.loc[(ARI['Code maladie'] == 25)]
ARI_25 = ARI_25.loc[(ARI_25['Date'].str[:4].isin(['2021','2022','2023']))]
```

Step 3: map (INSEE regions)

```
RegionsDic = {'AUVERGNE-RHONE-ALPES' : 'Auvergne-Rhône-Alpes',
              'GRAND EST' : 'Grand Est',
              'NOUVELLE-AQUITAINE' : 'Nouvelle-Aquitaine' […] }
ARI_25['region'] = ARI_25['Nom de la zone géographique'].map(RegionsDic)
```

Step 5: Save data to a MySQL table ari_raw, arif_stg

```
ARI_25.to_sql(' ari_raw', SQLengine, if_exists='append', index=False)
```

4.2 POLU Dataset Extraction

Exploration :
POLU dataset are proposed in set of .csv and .xml files in 3 folder, one per year 2021,2022,2023.
Each file named like FR_E2_2021-01-01.csv contain 1 days of data from all measuring station in France.
Available data are from 01/01/2021 to now.

Index of /lcsqa/concentrations-de-polluants-atmospheriques-reglementes/temps-reel/2021/

```
../
FR_E2_2021-01-01.csv                    30-Jun-2022 07:00           11253891
FR_E2_2021-01-01.xml                    30-Jun-2022 07:00           10140472
FR_E2_2021-01-02.csv                    30-Jun-2022 07:00           11207768
FR_E2_2021-01-02.xml                    30-Jun-2022 07:00           10094516
FR_E2_2021-01-03.csv                    30-Jun-2022 07:00           11192486
FR_E2_2021-01-03.xml                    30-Jun-2022 07:00           10083267
FR_E2_2021-01-04.csv                    30-Jun-2022 07:00           11279058
FR_E2_2021-01-04.xml                    30-Jun-2022 07:00           10161771
FR_E2_2021-01-05.csv                    30-Jun-2022 07:00           11289492
FR_E2_2021-01-05.xml                    30-Jun-2022 07:00           10172694
FR_E2_2021-01-06.csv                    30-Jun-2022 07:00           11235229
FR_E2_2021-01-06.xml                    30-Jun-2022 07:00           10119500
FR_E2_2021-01-07.csv                    30-Jun-2022 07:00           11270529
FR_E2_2021-01-07.xml                    30-Jun-2022 07:00           10149527
FR_E2_2021-01-08.csv                    30-Jun-2022 07:00           11254749
FR_E2_2021-01-08.xml                    30-Jun-2022 07:00           10139479
FR_E2_2021-01-09.csv                    30-Jun-2022 07:00           11233490
FR_E2_2021-01-09.xml                    30-Jun-2022 07:00           10121821
```

Example file FR_E2_2021-01-01.csv 10,9 Mo 50257 rows, 24 columns

Date de début	Date de fin	Organisme	code zas	Zas	code site	nom site	type d'implan	Polluant	type d'influence	discrimina	Réglemen
01/01/2021 00:00	01/01/2021 01:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 01:00	01/01/2021 02:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 02:00	01/01/2021 03:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 03:00	01/01/2021 04:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 04:00	01/01/2021 05:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 05:00	01/01/2021 06:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 06:00	01/01/2021 07:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 07:00	01/01/2021 08:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 08:00	01/01/2021 09:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 09:00	01/01/2021 10:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 10:00	01/01/2021 11:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 11:00	01/01/2021 12:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 12:00	01/01/2021 13:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 13:00	01/01/2021 14:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 14:00	01/01/2021 15:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 15:00	01/01/2021 16:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui
01/01/2021 16:00	01/01/2021 17:00	ATMO GRAND EST	FR44ZAG02	ZAG METZ	FR01005	Hayange	Périurbaine	PM10	Industrielle	A	Oui

Date de début	type d'évaluation	procédure de mesure	type de valeur	valeur	valeur bru	unité de m	taux de sa	couvertur	couvertur	code quali	validité
01/01/2021 00:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	18.9	18.875	µg-m3				A	1
01/01/2021 01:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	10.8	10.8	µg-m3				A	1
01/01/2021 02:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	10.3	10.3	µg-m3				A	1
01/01/2021 03:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	6.4	6.4	µg-m3				A	1
01/01/2021 04:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	8.1	8.05	µg-m3				A	1
01/01/2021 05:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	21.2	21.2	µg-m3				A	1
01/01/2021 06:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	22.7	22.675	µg-m3				A	1
01/01/2021 07:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	34.5	34.5	µg-m3				A	1
01/01/2021 08:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	33.1	33.075	µg-m3				A	1
01/01/2021 09:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	21.6	21.6	µg-m3				A	1
01/01/2021 10:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	21.7	21.725	µg-m3				A	1
01/01/2021 11:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	22.6	22.6	µg-m3				A	1
01/01/2021 12:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	21.4	21.35	µg-m3				A	1
01/01/2021 13:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	11.4	11.4	µg-m3				A	1
01/01/2021 14:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	14.8	14.8	µg-m3				A	1
01/01/2021 15:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	41.9	41.9	µg-m3				A	1
01/01/2021 16:00	mesures indicatives	Auto PM_Conf_app MP101M-RST	moyenne horaire validée	42	41.975	µg-m3				A	1

I used a python script to download all file per year and aggregate it per pollutant per region.
I will use Pandas Dataframes for :

- Reading the data without downloading it. (Pandas certainly download the file in his cache but it will handle deletion automatically.
- Performing aggregation per region and days (1 file per days) before insert into chosen database.

POLU Dataset Extract:

POLU_import_1.ipynb

Step 1: Import file
```
df =  readFile(pd.read_csv(fileUrl,sep=';')
```

Step 2: map (INSEE regions)
```
df_read['region'] = df_read['Organisme'].map(RegionsDic)
```

Step 3: Save raw data to table polu_r for other uses

Step 4: Group by regions and pollutants using mean
df_grouped = df.groupby(['region','Polluant']).agg({'unité de mesure': 'first', 'valeur': 'mean', 'valeur brute': 'mean'})

Step 5: Pivot the data to move values from columns to row
df_pivoted = df_grouped.pivot(index='region', columns='Polluant', values=['valeur brute'])

Step 6: Save data to a MySQL table polu_d
df_pivoted.to_sql('polu_d', SQLengine, if_exists='append', index=False)

4.3 SYNOP Dataset Extraction

SYNOP Dataset can be collected by two way:

- as a .csv file containing all data : donnees-synop-essentielles-omm.csv 1.1 Go
  2 247 960 rows 57 columns, with data from 05/01/2010 to 26/04/2023
- Using OpenDatasoft API : Implemented in a future session, it allow to get only the desired columns, filter on date or even weeks and use groupby to perform the aggregation.

SYNOP Dataset Extract

SYNOP_import_1.ipynb

Step 1: Import file
pd.read_csv(r'donnees-synop-essentielles-omm.csv',sep=';')

Step 2: Group by region and date and Step: 3 columns selection

```
df.groupby(['region (name)','Date']).agg({'Pression au niveau mer': 'mean',
                                'Direction du vent moyen 10 mn': mean',
                                'Vitesse du vent moyen 10 mn' : 'mean',
                                'Température' : 'mean',
                                'Humidité' : 'mean',
                                'Pression station' : 'mean',
                                'Précipitations dans la dernière heure' : 'mean',
                                'Température (°C)' : 'mean'})
```

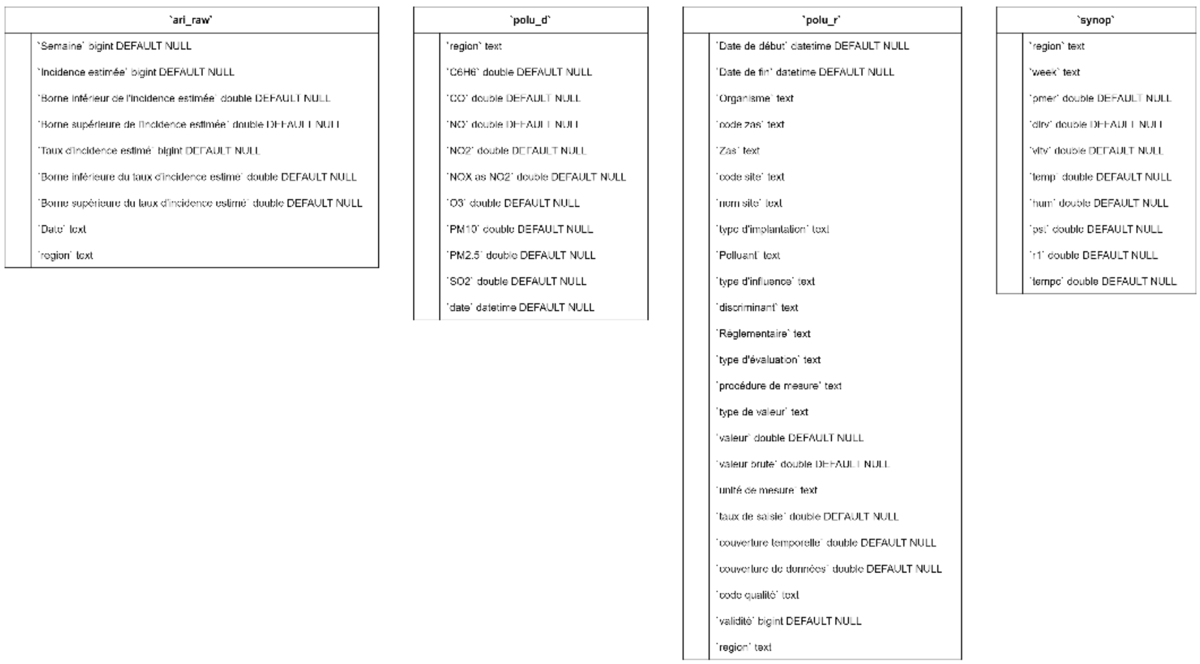
Step 4: Select period >2021 and create week column
df_grouped = df_grouped[df_grouped['DateD'] > "2021/01/01"]
df_grouped['Week']= df_grouped['DateD'].apply(lambda x: datetime.strftime(x, '%W'))

Step 5: Group by region, week
df_grouped.groupby(['region','week']).agg({'pmer': 'mean','dirv': 'mean', […]'tempc': 'mean'})

Step 6: Save to sql table synop
df_grouped_w.to_sql('synop', SQLengine, if_exists='append', index=False)

4.4 ari_stg staging area

This database is used as the SQL Data Lake, even if data are already transformed using group by and filters.



5 Data Transformation

Creation of the arif_dw data warehouse.

Region : creating a categorical table for INSEE region.

```sql
CREATE TABLE region (
    region_id INT AUTO_INCREMENT PRIMARY KEY,
    region_name VARCHAR(30) );
INSERT INTO region (region_name)
SELECT DISTINCT region
FROM arif_stg.ari_raw;
```

ari_w table creation:

```
CREATE TABLE ari_w (
    region_id INT NOT NULL,
    week INT NOT NULL,
    inc INT,
    inclow INT,
    inctop INT,
    inc100 INT,
    inc100low INT,
    inc100top INT,
    PRIMARY KEY (region_id, week),
    FOREIGN KEY (region_id) REFERENCES region(region_id) );
INSERT INTO ari_w
SELECT region.region_id
    , CAST( a1.Semaine AS  SIGNED)
    , a1.`Incidence estimée` AS inc
    , a1.`Borne inférieur de l'incidence estimée` AS inclow
    , a1.`Borne supérieure de l'incidence estimée` AS inctop
    , a1.`Taux d'incidence estimé` AS inc100
    , a1.`Borne inférieure du taux d'incidence estimé` AS inc100low
    , a1.`Borne supérieure du taux d'incidence estimé` AS inc100top
FROM arif_stg.ari_raw a1
    INNER JOIN region ON a1.region = region.region_name;
```

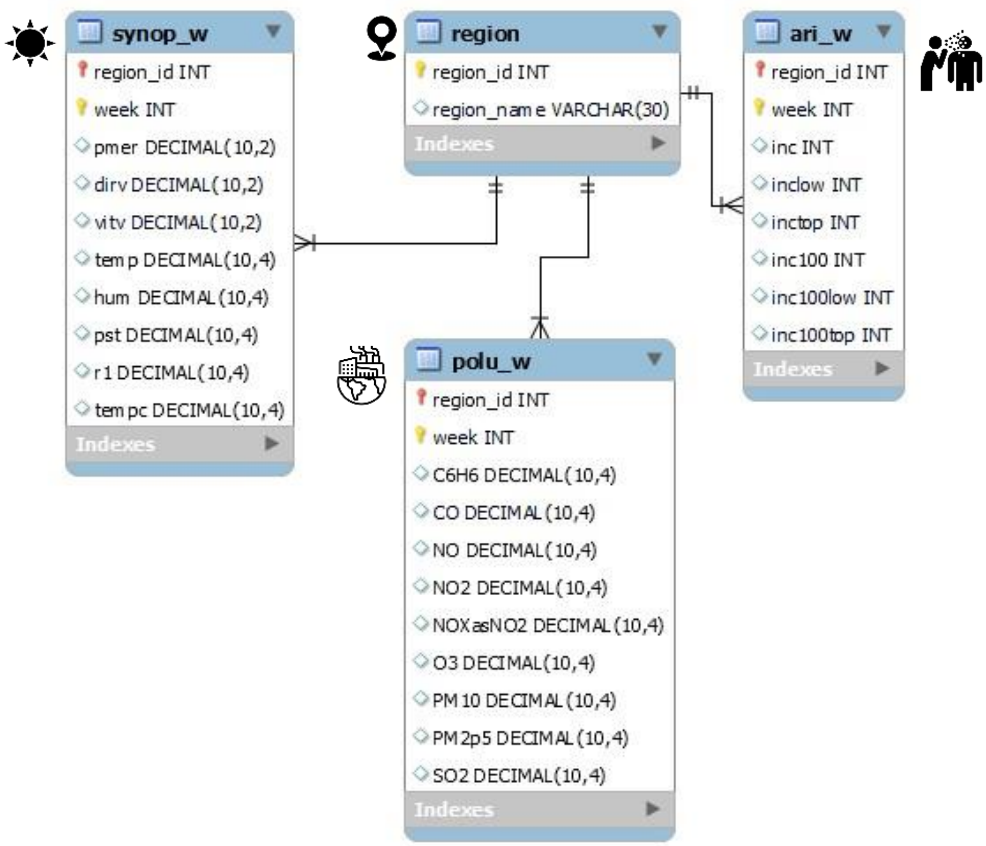
polu_w table creation with group by (regions, weeks)

```
CREATE TABLE polu_w (
    region_id INT NOT NULL,
    week INT NOT NULL,
    C6H6 DECIMAL(10, 4),
    CO DECIMAL(10, 4),
    `NO` DECIMAL(10, 4),
    NO2 DECIMAL(10, 4),
    NOXasNO2 DECIMAL(10, 4),
    O3 DECIMAL(10, 4),
    PM10  DECIMAL(10, 4),
    PM2p5 DECIMAL(10, 4),
    SO2 DECIMAL(10, 4),
    PRIMARY KEY (region_id, week),
    FOREIGN KEY (region_id) REFERENCES region(region_id) );
INSERT INTO polu_w
SELECT region.region_id
    , CAST( CONCAT(YEAR(p1.date), LPAD(WEEK(p1.date) +1, 2, '0')) AS  SIGNED) as week
    , AVG(p1.C6H6), AVG(p1.CO), AVG(p1.`NO`), AVG(p1.NO2), AVG(p1.`NOX as NO2`)
    , AVG(p1.O3), AVG(p1.PM10), AVG(p1.`PM2.5`), AVG(p1.SO2)
FROM arif_stg.polu_d p1
    INNER JOIN region ON p1.region = region.region_name
Group by region.region_id , week;
```

Synop_w table creation

```
CREATE TABLE synop_w (
    region_id INT NOT NULL,
    week INT NOT NULL,
    pmer DECIMAL(10, 2),
    dirv DECIMAL(10, 2),
    vitv DECIMAL(10, 2),
    temp DECIMAL(10, 4),
    hum DECIMAL(10, 4),
    pst DECIMAL(10, 4),
    r1 DECIMAL(10, 4),
    tempc DECIMAL(10, 4),
    PRIMARY KEY (region_id, week),
    FOREIGN KEY (region_id) REFERENCES region(region_id) );
INSERT INTO synop_w
SELECT region.region_id
    , CAST( s1.week AS  SIGNED) as week
    , s1.pmer, s1.dirv, s1.vitv, s1.temp, s1.hum, s1.pst, s1.r1, s1.tempc
FROM arif_stg.synop s1
    INNER JOIN region ON s1.region = region.region_name;
```

6 arif_dw Entity Relationship Diagram



6.1 Aggregate ARI, POLU and SYNOP

View **v_ari_polu_synop** creation

```
CREATE VIEW v_ari_polu_synop AS
SELECT region.region_name, region.region_id,   ari_w.week
    , DATE_FORMAT(DATE_ADD(DATE_FORMAT(CONCAT(SUBSTRING(ari_w.week, 1, 4), '-01-01'), '%Y-%m-%d'), INTERVAL (SUBSTRING(ari_w.week, 5) - 1) WEEK), '%Y-%m-%d') AS date
    ,   ari_w.inc,   ari_w.inclow,   ari_w.inctop,   ari_w.inc100,   ari_w.inc100low,   ari_w.inc100top
    ,   polu_w.C6H6,   polu_w.CO,   polu_w.NO,   polu_w.NO2,   polu_w.NOXasNO2,   polu_w.O3,   polu_w.PM10,   polu_w.PM2p5,   polu_w.SO2
    ,   synop_w.pmer,   synop_w.dirv,   synop_w.vitv,   synop_w.temp,   synop_w.hum,   synop_w.pst,   synop_w.r1,   synop_w.tempc
FROM region
    INNER JOIN ari_w ON region.region_id = ari_w.region_id
    INNER JOIN polu_w ON region.region_id = polu_w.region_id AND ari_w.week = polu_w.week
    INNER JOIN synop_w ON region.region_id = synop_w.region_id AND ari_w.week = synop_w.week
ORDER BY ari_w.week, region.region_id
```

A Date column is added for easy manipulation in tableau

View **v_ari_polu_synop_w1w2** creation : columns _w1 and _w2 with data from rows -1 and -2. Allowing to take account of diseases incubation periods.

```
CREATE VIEW v_ari_polu_synop_w1w2 AS
SELECT region.region_name, region.region_id,   a1.week
    , DATE_FORMAT(DATE_ADD(DATE_FORMAT(CONCAT(SUBSTRING(a1.week, 1, 4), '-01-01'), '%Y-%m-%d'), INTERVAL (SUBSTRING(a1.week, 5) - 1) WEEK), '%Y-%m-%d') AS date
    , a1.inc, a2.inc as inc_w1, a3.inc as inc_w2, a1.inclow, a2.inclow as inclow_w1, a3.inclow as inclow_w2, a1.inctop, a2.inctop as inctop_w1, a3.inctop as inctop_w2
    , a1.inc100, a2.inc100 as inc100_w1, a3.inc100 as inc100_w2, a1.inc100low, a2.inc100low as inc100low_w1, a3.inc100low as inc100low_w2
    , a1.inc100top, a2.inc100top as inc100top_w1, a3.inc100top as inc100top_w2
    , p1.C6H6, p2.C6H6 as C6H6_w1, p3.C6H6 as C6H6_w2
    , p1.CO, p2.CO as CO_w1, p3.CO as CO_w2
    , p1.`NO`, p2.`NO` as NO_w1, p3.`NO` as NO_w2
    , p1.NO2, p2.NO2 as NO2_w1, p3.NO2 as NO2_w2
    , p1.NOXasNO2, p2.NOXasNO2 as NOXasNO2_w1,p3.NOXasNO2 as NOXasNO2_w2
    , p1.O3, p2.O3 as O3_w1, p3.O3 as O3_w2
    , p1.PM10, p2.PM10 as PM10_w1, p3.PM10 as PM10_w2
    , p1.PM2p5, p2.PM2p5 as PM2p5_w1, p3.PM2p5 as PM2p5_w2
    , p1.SO2,p2.SO2 as SO2_w1,p3.SO2 as SO
    , s1.pmer, s2.pmer as pmer_w1, s3.pmer as pmer_w2
    , s1.dirv, s2.dirv as dirv_w1, s3.dirv as dirv_w2
    , s1.vitv, s2.vitv as vitv_w1, s3.vitv as vitv_w2
    , s1.temp, s2.temp as temp_w1, s3.temp as temp_w2
    , s1.hum, s2.hum as hum_w1, s3.hum as hum_w2
    , s1.pst, s2.pst as pst_w1, s3.pst as pst_w2
    , s1.r1, s2.r1 as r1_w1, s3.r1 as r1_w2
    , s1.tempc, s2.tempc as tempc_w1, s3.tempc as tempc_w2
FROM region
    INNER JOIN ari_w a1 ON region.region_id = a1.region_id
        LEFT JOIN ari_w a2 ON a1.region_id = a2.region_id AND a1.week-a2.week= 1
        LEFT JOIN ari_w a3 ON a1.region_id = a3.region_id AND a1.week-a3.week= 2
    INNER JOIN polu_w p1 ON region.region_id = p1.region_id AND a1.week = p1.week
        LEFT JOIN polu_w p2 ON p1.region_id = p2.region_id AND p1.week-p2.week= 1
        LEFT JOIN polu_w p3 ON p1.region_id = p3.region_id AND p1.week-p3.week= 2
    INNER JOIN synop_w s1 ON region.region_id = s1.region_id AND a1.week = s1.week
        LEFT JOIN synop_w s2 ON s1.region_id = s2.region_id AND s1.week-s2.week= 1
        LEFT JOIN synop_w s3 ON s1.region_id = s3.region_id AND s1.week-s3.week= 2
    WHERE a1.week > 202102  ORDER BY a1.week, region.region_id
```

View POLU pivoted creation : For tableau dashboard, creating a pivoted view of POLU data

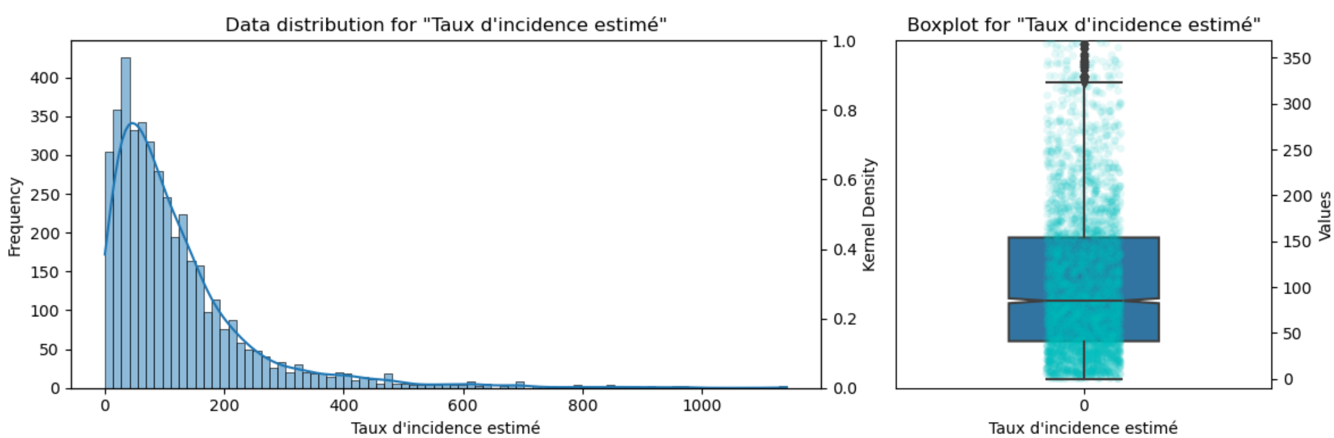
```
CREATE VIEW v_polu_list AS
SELECT region, year_week, 'C6H6' AS pollutant, C6H6 AS value FROM polu_w
UNION ALL
SELECT region, year_week, 'CO' AS pollutant, CO AS value FROM polu_w
UNION ALL
SELECT region, year_week, 'NO' AS pollutant, NO AS value FROM polu_w
UNION ALL
SELECT region, year_week, 'NO2' AS pollutant, NO2 AS value FROM polu_w
UNION ALL
SELECT region, year_week, 'NOXasNO2' AS pollutant, NOXasNO2 AS value FROM polu_w
UNION ALL
```

7 Data cleaning and Exploratory data analysis

7.1 Individual dataset analysis

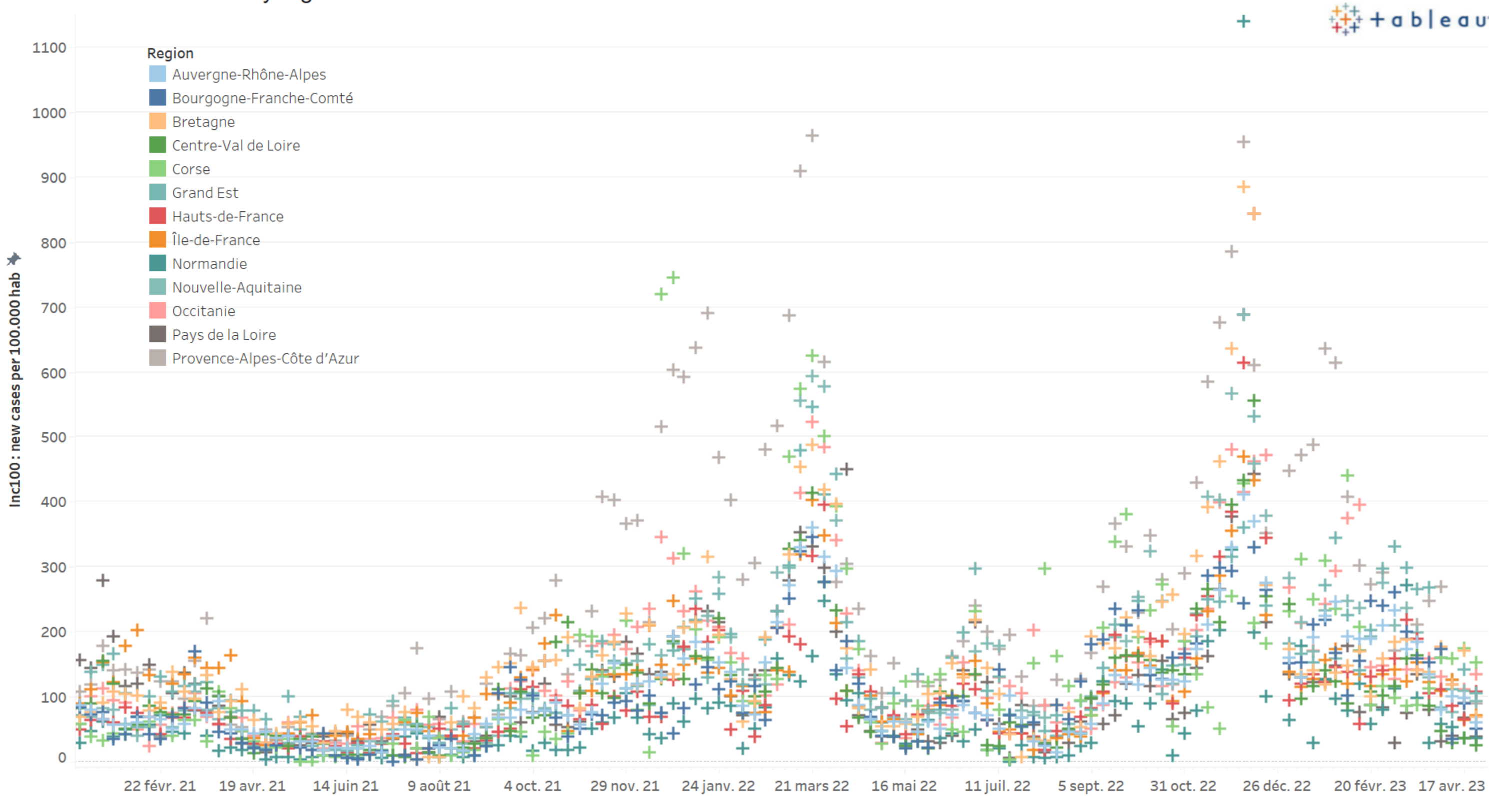
Creation of describeDataset, function to show statistics and distribution
describeDataset(df_ARI,showHead=0,showGraphs=True, dotcolor='cyan')

ARI Dataset



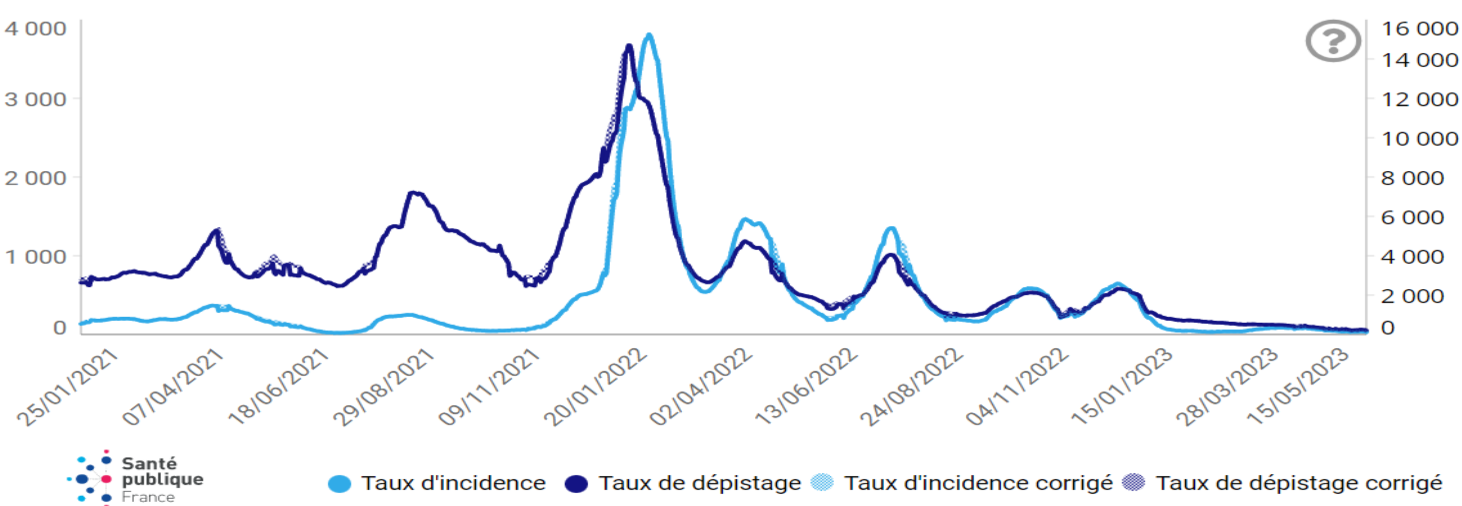
➔ The Incidence rate estimated, takes into account the population of the region.

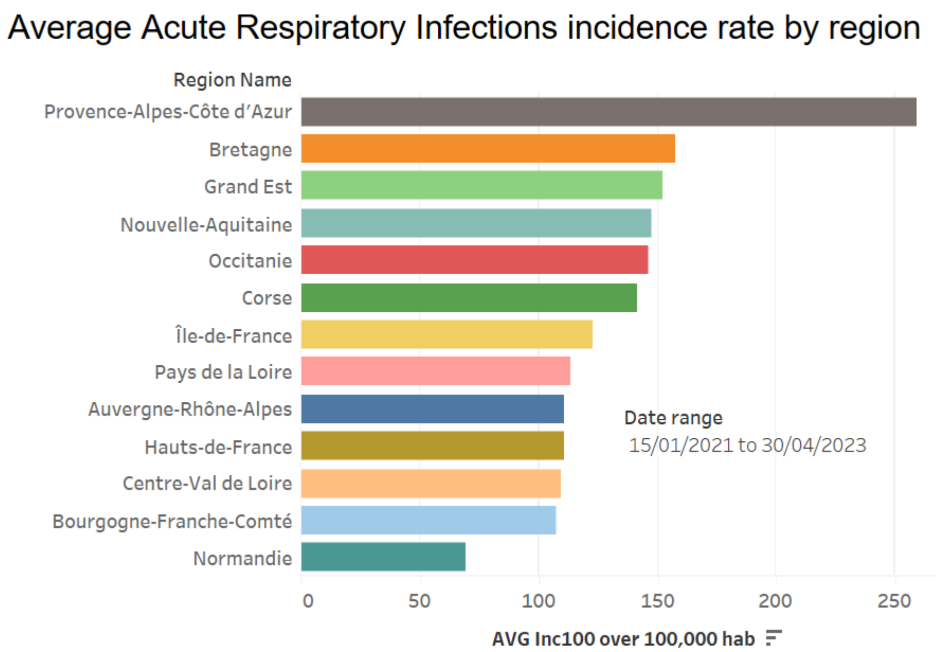
Incidence rate over time by region



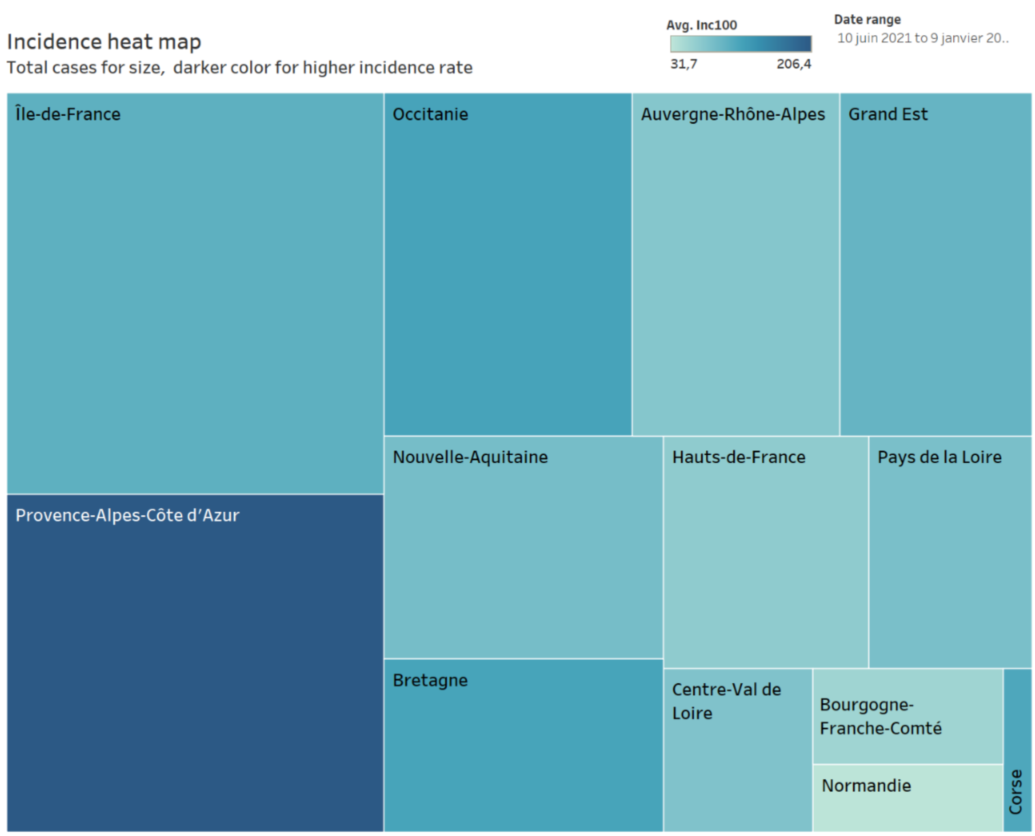
➔ We can identify two main waves in the past two years. By comparing the data to [COVID](#) from Santé Publique France below the detection rate has huge impact on incidence rate.

COVID incidence rate and detection rate

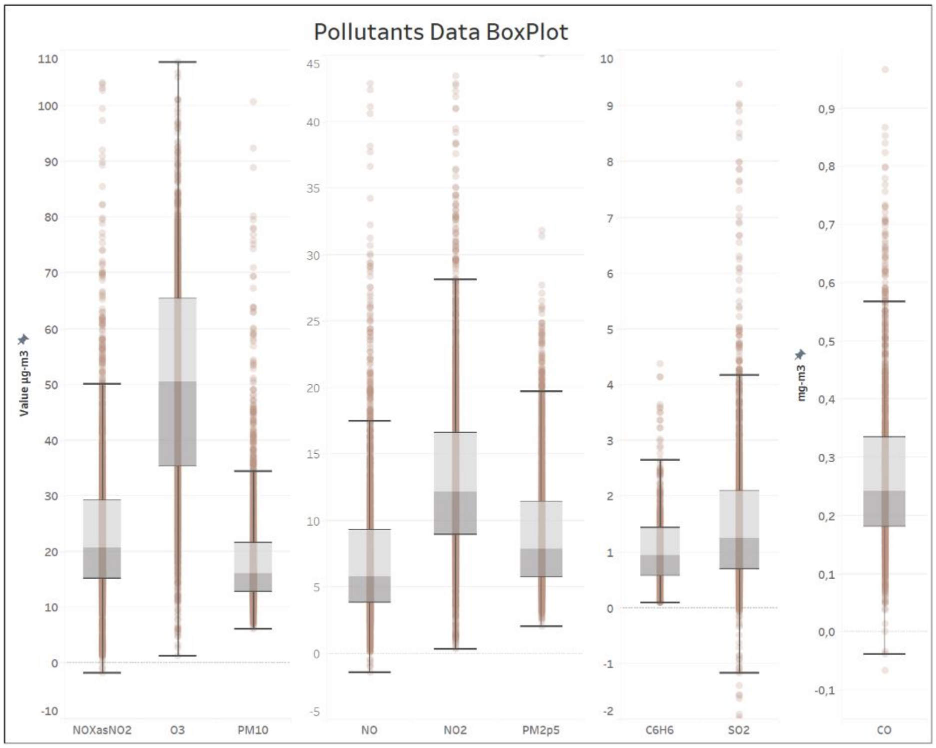




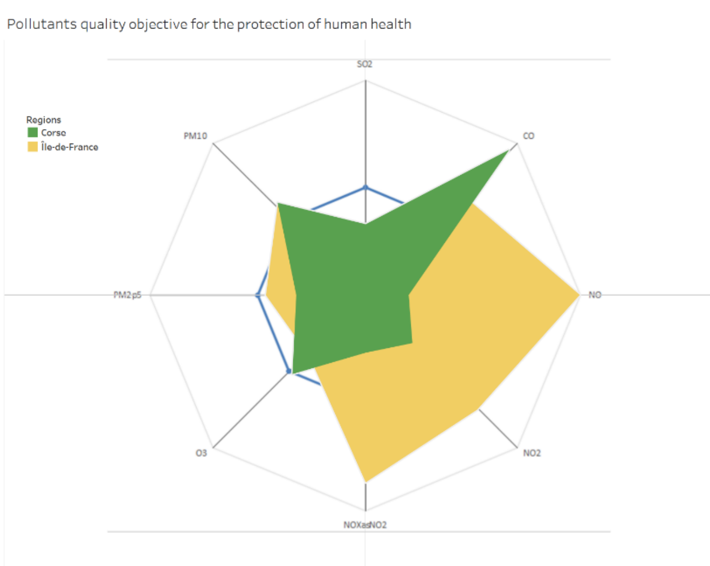
➔ If regions have not the same incidence rate, the number of cases is not necessary the highest

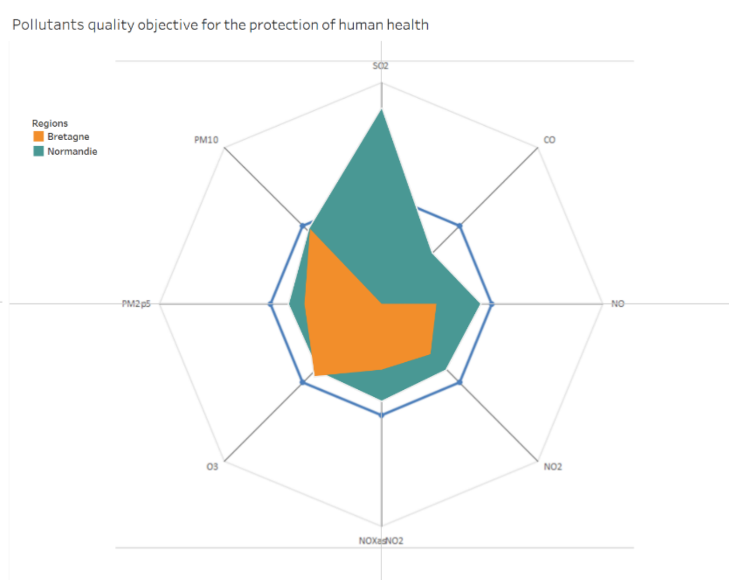


POLU Dataset



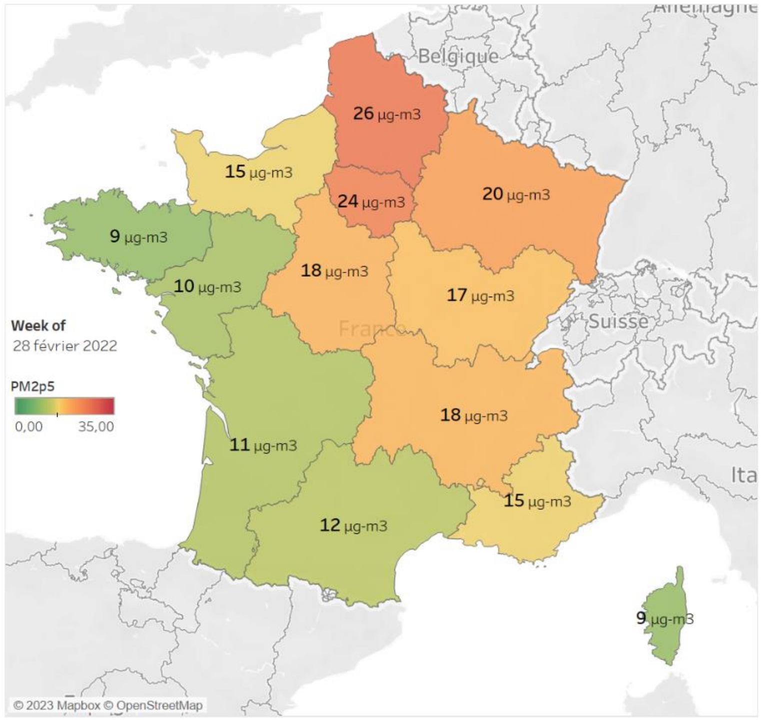
➔ If most pollutants share the same unit of measure, the distribution range maybe different





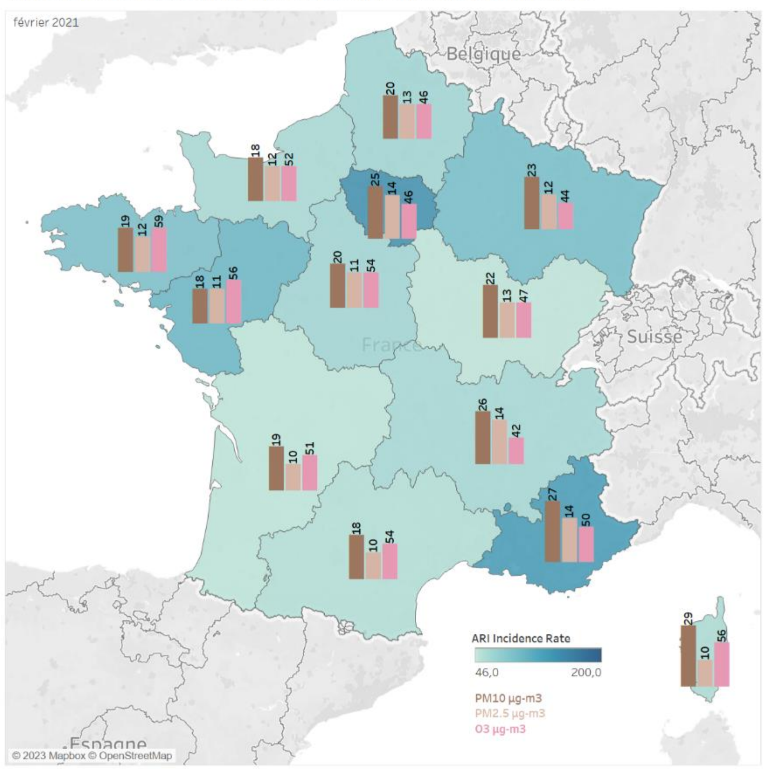
➔ Regions have differnets pollutants exposition due to geography or industry density

PM2.5 concentration for a week

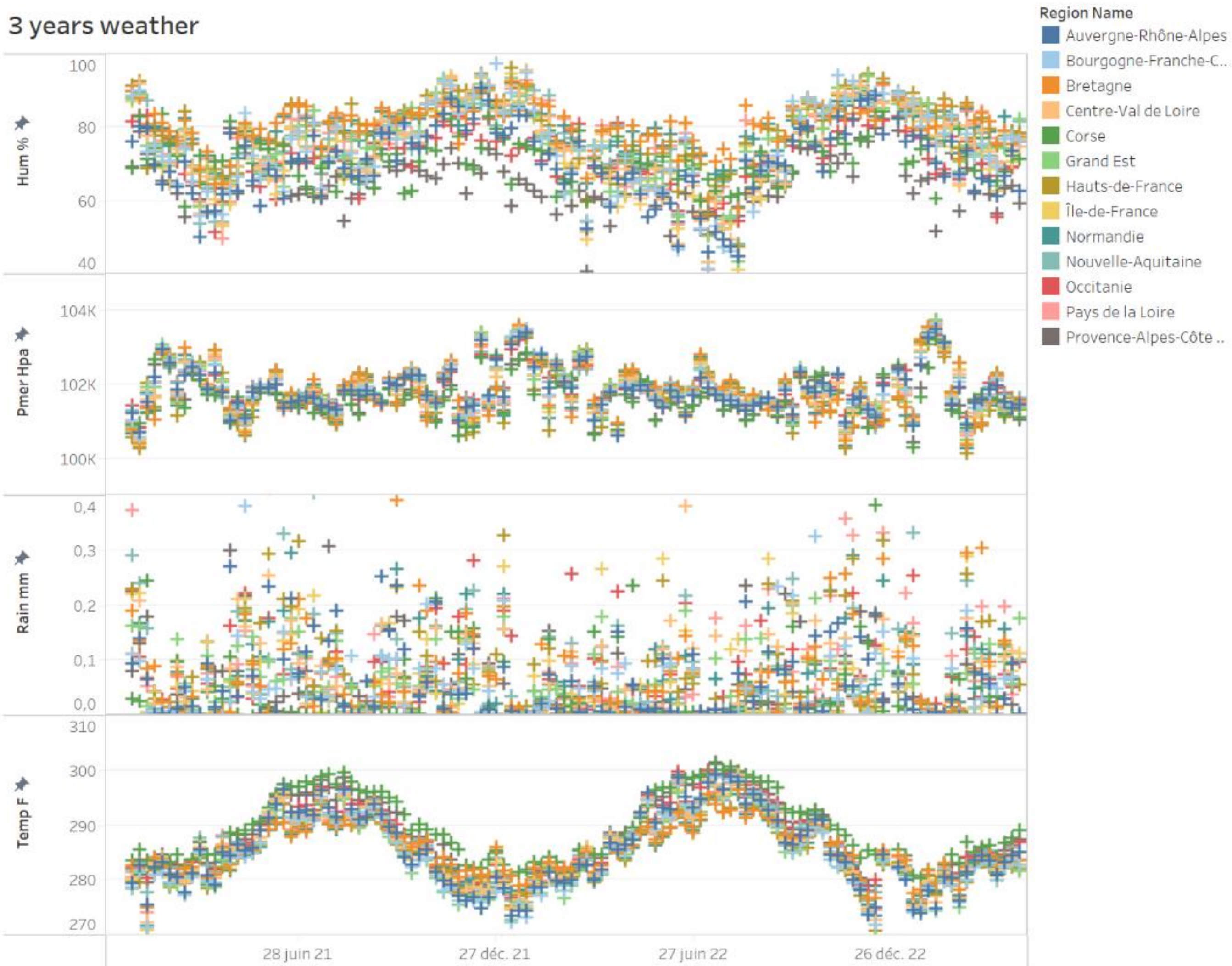


➔ The weather and wind condition shape the repartition

ARI Incidence rate, PM10, PM2.5 & O3 concentration for Feb 2021

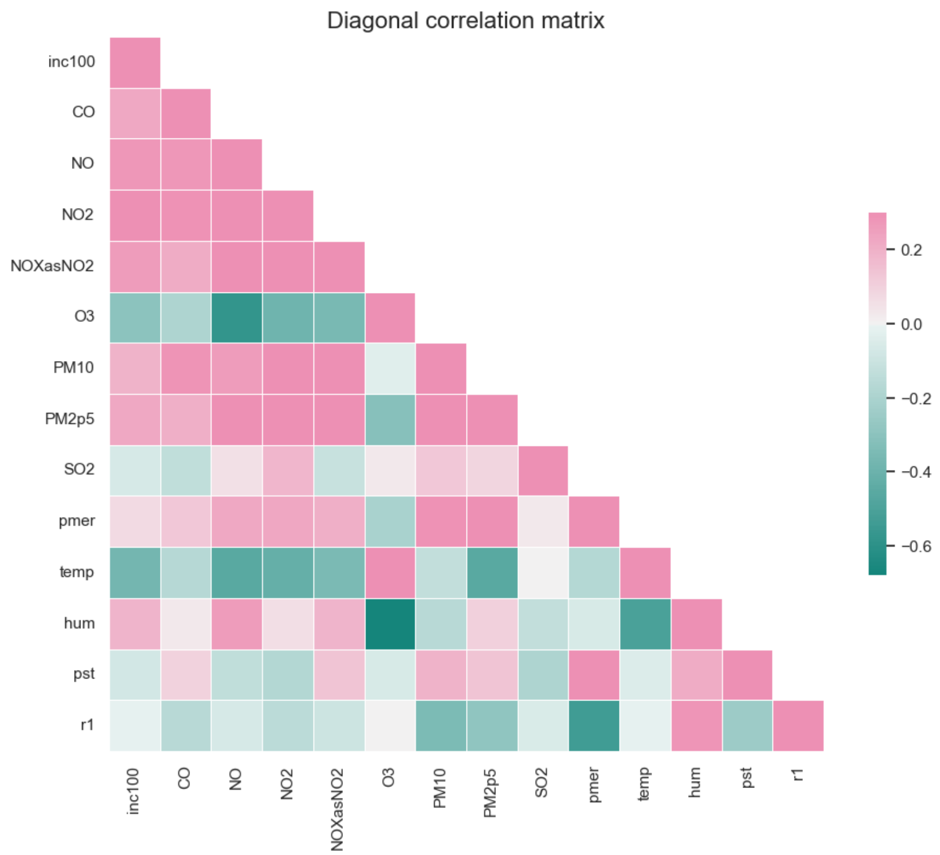


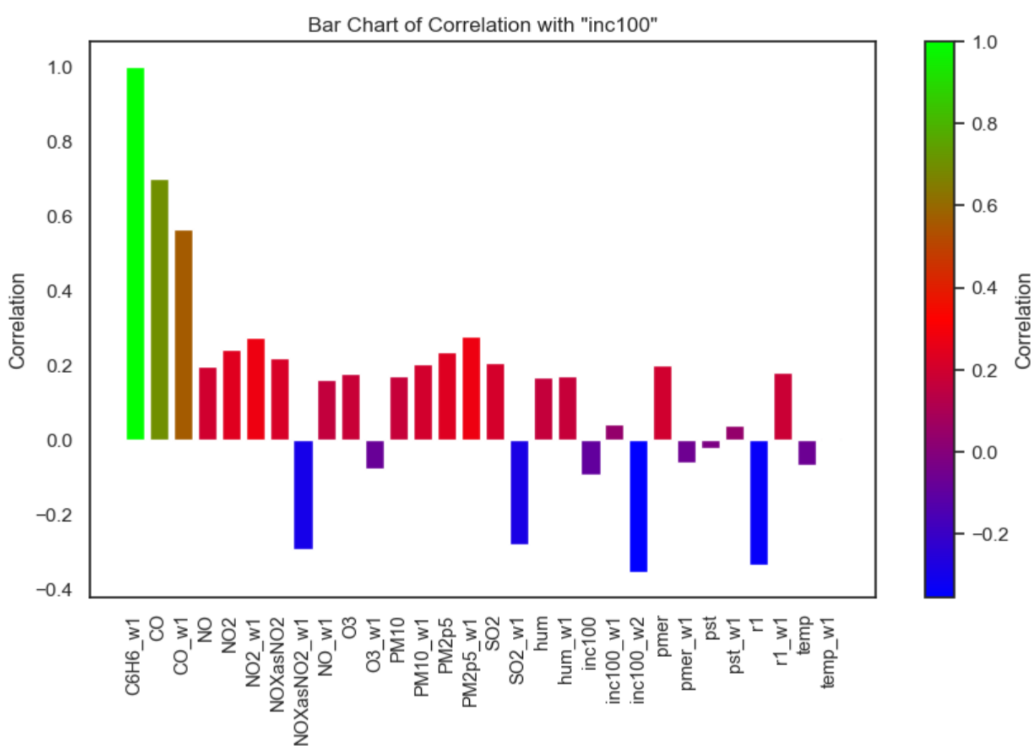
SYNOP Dataset



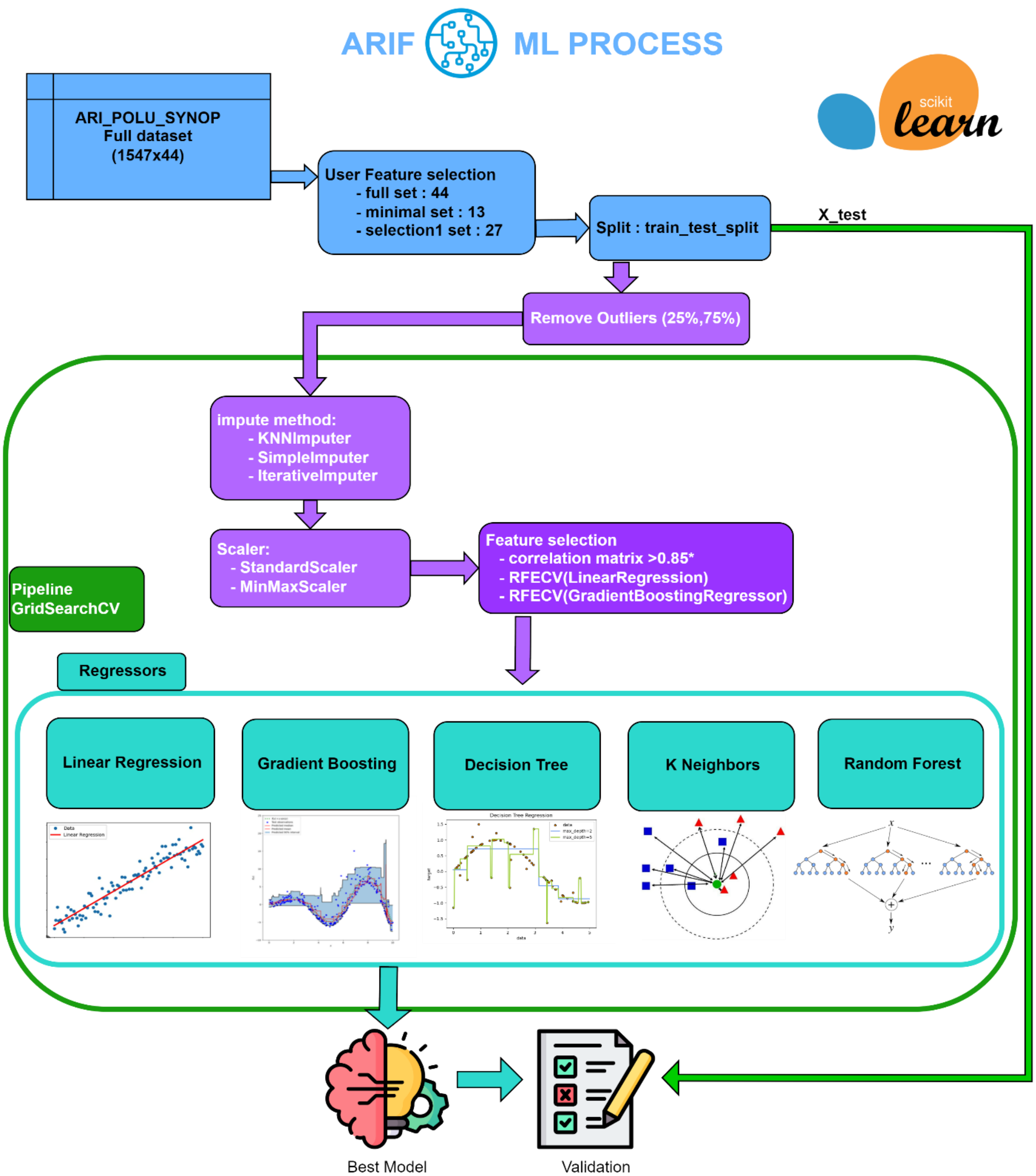
➔ Good repartition of data, seasons can be identified

7.2 Correlation analysis





8 Machine Learning



8.1 ML Process

5 models fitting continuous data are chosen from the python library scikit-learn :

```
LinearRegression
GradientBoostingRegressor
RandomForestRegressor
KNeighborsRegressor
DecisionTreeRegressor
```

3 imputing and a scaling method also from scikit-learn

`KNNImputer`, `SimpleImputer`, `IterativeImputer + StandardScaler`

`Pipeline + GridSearchCV` are used for automation and cross validation

Score selection:
MAPE score (Mean Absolute Percentage Error) a commonly metric used for forecasting, represent the average percentage difference between the predicted values and the actual values.

```
MAPE = (1 / n) * Σ(|(Y_actual - Y_pred) / Y_actual|) * 100
```

8.2 Models & methods evaluations

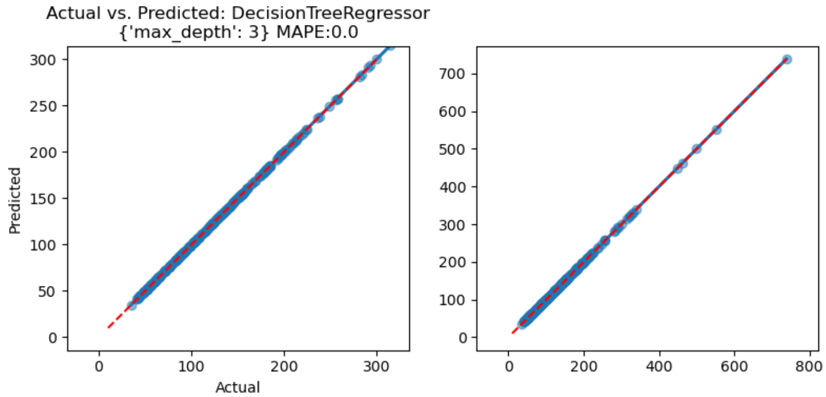
Scores, best hyperparameters and 10 biggest errors on prediction are stored for analysis.

scaler	impute	features	n_fea	gcv_model	tests	mape	bestparam
StandardScaler	imputeByMean	RFECV(GradientBoostingRegressor)	14	GradientBoostingRegressor	32	51,06%	{'loss': 'absolute_error', 'max_depth': 5, 'n_estimators': 400}
StandardScaler	SimpleImputer	RFECV(GradientBoostingRegressor)	18	GradientBoostingRegressor	32	52,57%	{'loss': 'absolute_error', 'max_depth': 7, 'n_estimators': 200}
StandardScaler	imputeByMean	none		GradientBoostingRegressor	32	55,84%	{'loss': 'absolute_error', 'max_depth': 4, 'n_estimators': 400}
StandardScaler	imputeByMean	none		DecisionTreeRegressor	5	56,65%	{'max_depth': 4}
StandardScaler	imputeByMean	RFECV(GradientBoostingRegressor)	14	RandomForestRegressor	16	58,20%	{'max_depth': 5, 'n_estimators': 400}
StandardScaler	SimpleImputer	RFECV(LinearRegression)	26	GradientBoostingRegressor	32	58,84%	{'loss': 'absolute_error', 'max_depth': 7, 'n_estimators': 400}
StandardScaler	SimpleImputer	none	17	GradientBoostingRegressor	32	58,91%	{'loss': 'absolute_error', 'max_depth': 7, 'n_estimators': 100}
StandardScaler	imputeByMean	none		RandomForestRegressor	16	59,35%	{'max_depth': 5, 'n_estimators': 300}
StandardScaler	SimpleImputer	RFECV(GradientBoostingRegressor)	18	RandomForestRegressor	16	59,62%	{'max_depth': 5, 'n_estimators': 300}
StandardScaler	IterativeImputer	RFECV(LinearRegression)	16	GradientBoostingRegressor	32	59,74%	{'loss': 'absolute_error', 'max_depth': 7, 'n_estimators': 400}
StandardScaler	imputeByMean	RFECV(LinearRegression)	26	RandomForestRegressor	16	59,99%	{'max_depth': 5, 'n_estimators': 200}
StandardScaler	SimpleImputer	none	17	RandomForestRegressor	16	60,05%	{'max_depth': 5, 'n_estimators': 100}
StandardScaler	KNNImputer	none	14	GradientBoostingRegressor	32	60,56%	{'loss': 'absolute_error', 'max_depth': 7, 'n_estimators': 300}
StandardScaler	SimpleImputer	RFECV(LinearRegression)	26	RandomForestRegressor	16	61,89%	{'max_depth': 7, 'n_estimators': 100}
StandardScaler	imputeByMean	RFECV(LinearRegression)	26	GradientBoostingRegressor	32	62,41%	{'loss': 'absolute_error', 'max_depth': 7, 'n_estimators': 200}
StandardScaler	IterativeImputer	RFECV(LinearRegression)	16	RandomForestRegressor	16	64,02%	{'max_depth': 9, 'n_estimators': 300}
StandardScaler	KNNImputer	RFECV(GradientBoostingRegressor)	17	GradientBoostingRegressor	32	64,38%	{'loss': 'ls', 'max_depth': 4, 'n_estimators': 100}
StandardScaler	IterativeImputer	RFECV(LinearRegression)	16	KNeighborsRegressor	12	65,24%	{'algorithm': 'auto', 'n_neighbors': 3}
StandardScaler	IterativeImputer	RFECV(GradientBoostingRegressor)	27	GradientBoostingRegressor	32	66,08%	{'loss': 'ls', 'max_depth': 7, 'n_estimators': 400}
StandardScaler	IterativeImputer	none	18	GradientBoostingRegressor	32	66,23%	{'loss': 'ls', 'max_depth': 7, 'n_estimators': 400}
StandardScaler	imputeByMean	RFECV(LinearRegression)	26	DecisionTreeRegressor	5	66,29%	{'max_depth': 3}

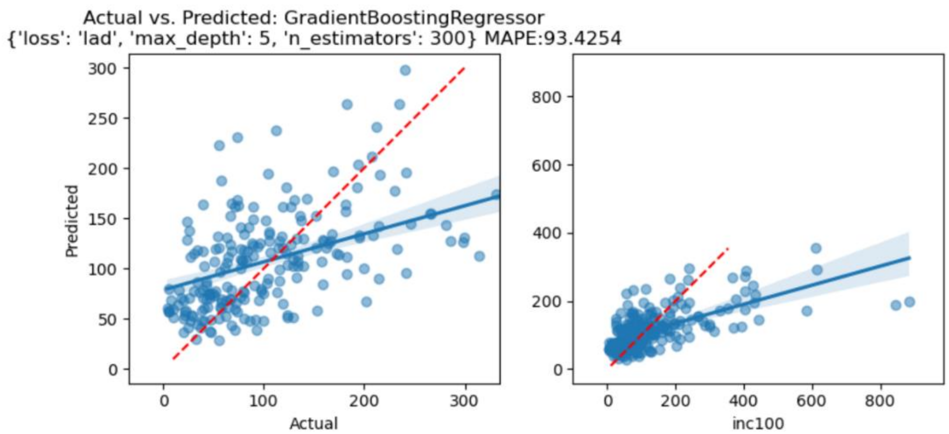
Best parameters computed are re-inserted in Recursive Feature Elimination (RFECV) for next run.

8.3 Result visualizations:

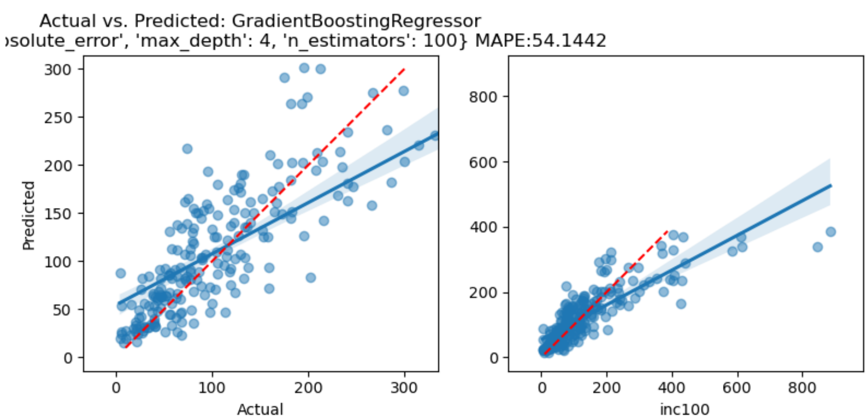
Target repartition with MAPE=0, perfect match for every point predicted



Bad repartition with MAPE=93%

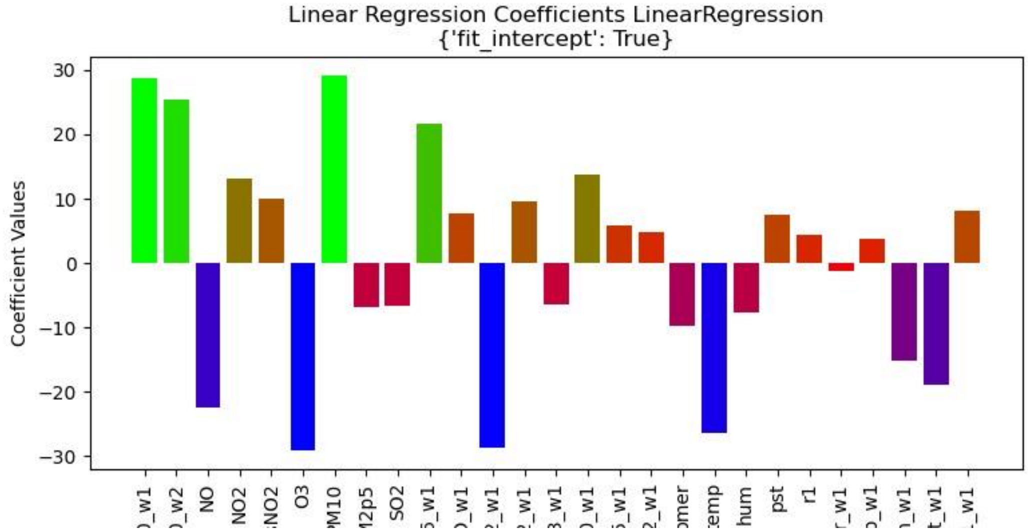


Better result, MAPE = 54%

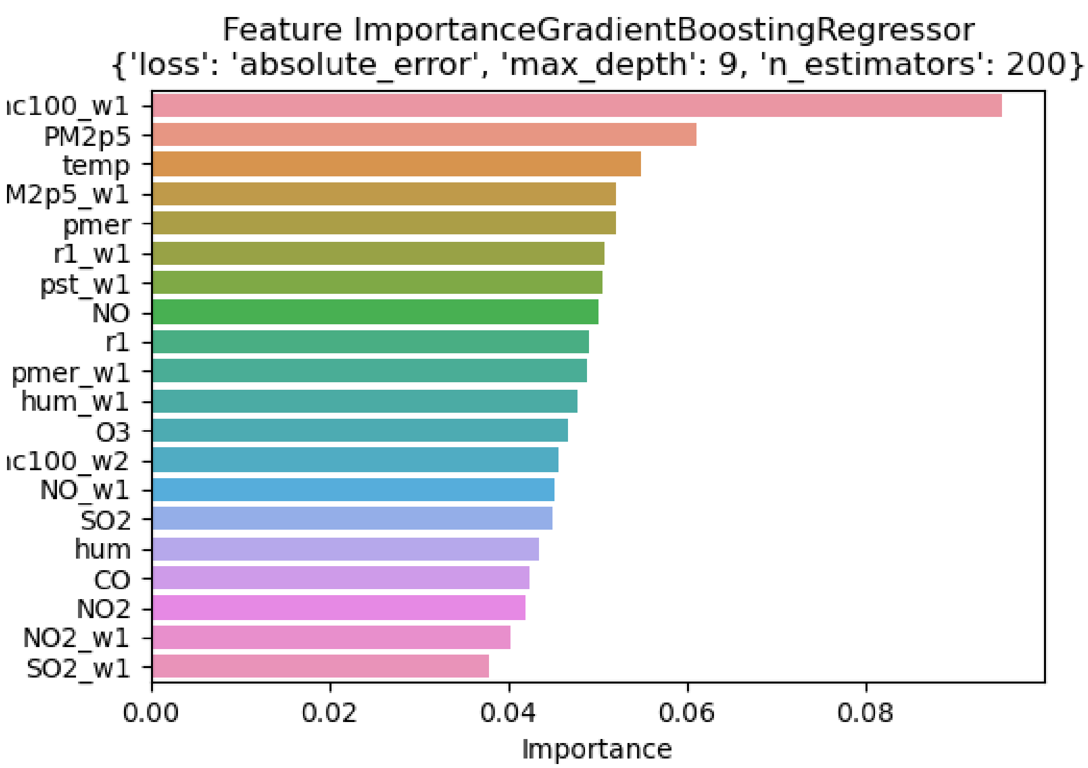


Features importance :

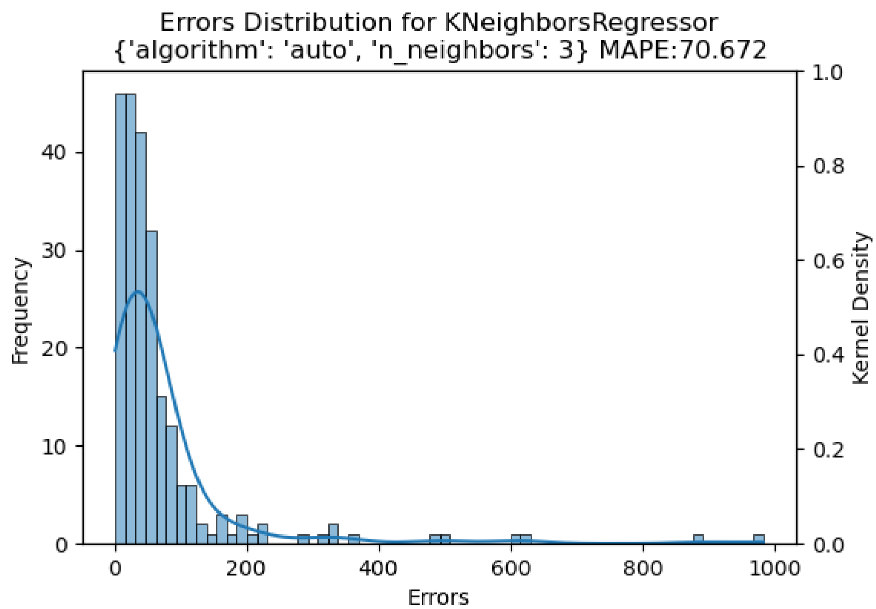
With Linear regression:



With Gradient Boosting Regressor, Random Forest Regressor and Decision Tree Regressor :



Error distribution



`in progress …`

9 Conclusion

For now the best MAPE score obtained is 52%. More simulation need to be done with more data investigation.
After splitting the data to isolate a test dataset, splitting a second time to perform training and evaluation, the algorithms have only 1039 row to train with.

While some data sources are limited before the starting study date 01/01/2021, they all are updated daily or weekly. This allow to add an update process by reading lasts POLU days file and using the API for other data sources.
After analyzing features importance, values from previous week (inc100_w1) are mandatory for a good 1 week prediction. The update process is then mandatory.
The ML algorithm will then use a autoML process to improve performance.