17ª edición-Máster de Data Science

Data Science

Kschool

Curso 2019-2020

Predicción de la contaminación atmosférica de Barcelona mediante datos meteorológicos e intensidad del tráfico.

Fecha de entrega
04/09/2020

Fecha de presentación
18/09/2020

Marc Jiménez Silva

Agradecimientos

Estas líneas del trabajo van dirigidas a todas aquellas personas que me han ayudado y me han apoyado durante estos meses con este proyecto, aunque haya sido difícil por la situación que hemos vivido, os lo agradezco a todos.

En primer lugar, mi agradecimiento va para todos los profesores y compañeros que he tenido en Kschool por enseñarme todos los conocimientos adquiridos y ayudado cuando no se entendían las cosas, en especial, a Alberto Rubio por su confianza y las recomendaciones.

A mis amigos, por darme consejos cuando el máster se estaba poniendo muy difícil y apoyarme en esos momentos difíciles.

Y sobre todo, a mis padres y a mi hermana por el apoyo incondicional durante todo el máster y depositar en mí toda su confianza.

¡Gracias a todos!

Resumen

En este trabajo se muestra un procedimiento para predecir la contaminación atmosférica de Barcelona. Para empezar, se compara diferentes modelos que son Arima, regresión lineal, regresión polinómica, árboles de decisión, random forest y Xg boost para predecir la contaminación de tres contaminantes, (NO, NO2 y NOX) en las tres estaciones que miden la contaminación urbana de la ciudad de Barcelona. Las variables meteorológicas son factores claves en la difusión de los contaminantes, producidos en mayor cantidad por las emisiones de fábricas y vehículos que hay en Barcelona. Por este motivo, se añaden estas variables en los modelos de predicción para encontrar una mejor solución.

Resum

En aquest treball es mostra un procediment per predir la contaminació atmosfèrica de Barcelona. Per començar, es compara diferents models que son Arima, regressió lineal, regressió polinòmica, arbres de decisió, random forest i Xgboost per predir la contaminació de tres contaminants (NO, NO2 i NOx) en les tres estacions que mesuren la contaminació urbana de la ciutat de Barcelona. Les variables meteorològiques són factors claus en la difusió dels contaminants, produïts en major quantitat per les emissions de fàbriques i vehicles que hi ha a Barcelona. Per aquest motiu, s'afegeixen aquestes variables en els models de predicció per trobar una millor solució.

Abstract

This work shows a procedure to predict the atmospheric pollution of Barcelona. To start, different Arima, linear regression, polynomial regression, decision trees, random forest and Xgboost models are compared to predict the contamination of three pollutants (NO, NO2 and NOX) in the three stations that measure urban pollution in the city from Barcelona. Meteorological variables are key factors in the diffusion of pollutants, produced in greater quantity by emissions from factories and vehicles in Barcelona. For this reason, these variables are added in the prediction models to find a better solution.

Índice general

Capítulo 1

Introducción

Existe una constatación de que las acciones humanas afectan al clima del planeta y la salud de los humanos. Todo esto se produce por la combustión ineficiente de combustibles fósiles, que esto a su vez, provoca el efecto invernadero.

Tal y como dice la Organización Mundial de la Salud (OMS), mediante la disminución de los niveles de contaminación del aire los países pueden reducir la carga de morbilidad derivada de accidentes cerebrovasculares, cánceres de pulmón y neumopatías crónicas y agudas, entre ellas el asma. Por lo que cuanto más bajos sean los niveles de contaminación del aire mejor será la salud cardiovascular y respiratoria de la población, tanto a largo como a corto plazo. Según estimaciones de 2016, la contaminación atmosférica en las ciudades y zonas rurales de todo el mundo provoca cada año 4,2 millones de defunciones prematuras [1].

La contaminación atmosférica se refiere a la presencia, en las distintas capas de aire que componen la atmósfera terrestre, de sustancias y formas de energía ajenas a su constitución natural y que pueden representar una fuente de riesgos, daños y molestias para la vida tal y como la conocemos [2]. La contaminación del aire afecta a diferentes grupos de personas de diferentes maneras. Se observan impactos más graves en la salud de las personas que ya están enfermas. Además, las poblaciones más vulnerables como los niños, los ancianos y los hogares con ingresos más bajos y acceso limitado a la atención médica son más susceptibles a los efectos adversos de la exposición a la contaminación del aire [3]. Por esa razón, debe de reducirse la exposición al aire sobretodo los grupos de riesgo.

Para controlar los límites de los contaminantes se utilizan las estaciones de medición, donde miden la concentración del aire de los principales contaminantes ambientales que perjudican la salud de las personas [4].

Las estaciones en Barcelona están situadas en lugares representativos de los diferentes tipos de calle. Así, los registros que obtienen pueden ser extrapolados a otras áreas con trama urbana similar [4] .

Las estaciones se diferencian en tres tipologías: estaciones de fondo urbano, estaciones de suburbano y estaciones de tráfico.

- Estaciones de fondo urbano: Palau Reial, Zona Universitària, Sants, parque de la Ciutadella (IES Verdaguer), Poblenou, Vall d'Hebron.
- Estaciones de suburbano: IES Goya.
- Estaciones de tráfico: L'Eixample, Universitat, Gràcia-Sant Gervasi [4] .

Por otra parte, el volumen y variedad de información que se encuentra informatizada en bases de datos y la inmensa cantidad de datos producidos por los sensores, ha crecido espectacularmente en las últimas décadas. Mucha de esta información almacena históricos sobre diferentes mediciones, lo cual resulta útil para explicar el pasado, entender el presente y predecir información futura. Además, los datos pueden provenir de diversas fuentes, por lo que es necesario un análisis de los mismos para la obtención de información útil [5]

Estas restricciones han causado la creación de herramientas que permiten sacar el conocimiento práctico desde la información de la base de datos. Estas técnicas se engloban en la minería de datos que hacen posible construir modelos predictivos para poder tener un mayor poder en la toma de decisiones.

Por tanto, la contaminación atmosférica es un problema grave. Es importante observar los niveles de contaminación atmosférica para conseguir su reducción a niveles admisibles de aquellos agentes cuya apariencia en la atmósfera pueden producir efectos desfavorables en la salud de todos los agentes afectados. A no ser que se lleve a cabo un control adecuado, la multiplicación de las fuentes contaminantes del mundo moderno puede llegar a producir daños irreparables para el medio ambiente y para toda la humanidad [7].

1.1 Objetivo

En este proyecto final debido a la situación ambiental que se haya, donde el problema de la contaminación cada vez se está haciendo más relevante, se ha plasmado lo que se ha aprendido en este máster, en este tema que me parece tan importante. Con esta intención, se estudia cómo ejecutar la predicción de la contaminación atmosférica. Para hacer estas predicciones, se analiza los resultados que se han obtenido en distintas técnicas para construir un modelo, se tiene en cuenta diferentes características que puedan influir en la contaminación, la intensidad del tráfico y las variables meteorológicas.

1.2 Estructura

La memoria se estructura de la siguiente manera:

- En el capítulo 2 se describen los datos que se usarán en el proyecto.
- En el capítulo 3 se explica cómo se ha seleccionado los contaminantes, las variables meteorológicas y la intensidad del tráfico.
- En el capítulo 4 se describe el proceso de predicción de nuestros modelos y su exploración.
- En el capítulo 5 se muestran las visualizaciones que hemos hecho en la exploración de los datos.
- En el capítulo 6 comprende las conclusiones del proyecto.

Capítulo 2

Datos

2.1 Introducción

En este capítulo se indaga en la contaminación atmosférica y en las características que pueden influir en ella, como el efecto que tiene el tránsito en la concentración de la contaminación atmosférica y el efecto que causa las variables meteorológicas de dispersión y de concentración de la contaminación atmosférica.

2.2 Contaminación atmosférica

La contaminación atmosférica es la presencia en la atmósfera de sustancias, materias o formas de energía que supongan una molestia grave, un peligro o un daño para el ser humano o el medio ambiente [8].

Hay diversas fuentes de contaminación atmosférica, tanto de origen humano como natural:

- el consumo de combustibles fósiles para la generación de electricidad, el transporte, la industria y los hogares;
- los procesos industriales y el uso de disolventes, por ejemplo en las industrias químicas y minerales;
- la agricultura;
- el tratamiento de residuos;
- algunos ejemplos de fuentes de emisión naturales son las erupciones volcánicas, el polvo arrastrado por el viento, el aerosol de sal marina y las emisiones de compuestos orgánicos volátiles de las plantas [9].

Hay dos grandes grupos de contaminantes atmosféricos son los siguientes: las partículas y los gases. Los gases contaminantes son componentes que su concentración en cantidades muy grandes en la atmósfera genera problemas medioambientales y de salud. Las partículas en suspensión (PM, del inglés Particulate Matter) son todas las partículas sólidas y líquidas que se encuentran suspendidas en el aire, la mayor parte de las cuales suponen un peligro. Esta mezcla compleja contiene, entre otras cosas, polvo, polen, hollín, humo y pequeñas gotas [10].

Dentro de estos dos grandes grupos hay otra clasificación entre primarios y secundarios. Los primarios son contaminantes que se emiten directamente desde una fuente de emisión puede ser natural o origen no natural y, los secundarios son los contaminantes que no se emiten de forma directa, sino que su principio es debido a la interacción que le sucede a los contaminantes primarios en la atmósfera.

Los principales contaminantes de la atmósfera son los siguientes:

- Las partículas en suspensión tienen una gran variedad de tamaños, emitidas al aire de forma directa cuando viene de la combustión. Los sulfatos son los principales componentes de las partículas en suspensión que contaminan el aire. Su formación proviene del origen primario y secundario, su estado es sólido y líquido y provienen de los vehículos y los procesos industriales.

- El dióxido de azufre (SO2) es un gas incoloro, no inflamable y no explosivo que tiene una vida de unos 3 días.Su principal fuente es la quema de combustibles fósiles ricos en azufre, si bien procesos naturales tales como las erupciones volcánicas también liberan SO2 a la atmósfera. Este gas contribuye a la acidificación a los ecosistemas mediante su deposición seca o húmeda (cuando entra en contacto con el vapor de agua presente en la atmósfera)[11].

- Los óxidos de nitrógeno se presentan en la atmósfera como:

    - Óxido nitroso (N2O), un gas volátil, incoloro, de olor dulce y de ligera toxicidad que se ha empleado de forma habitual como droga por sus efectos alucinatorios y el estado de euforia que genera (también se le conoce como el gas de la risa).
    - Dióxido de nitrógeno (NO2) es un gas contaminante que resulta de la combustión efectuada a alta temperatura (origen natural y antropogénico). Es tóxico e irritante y precursor de la formación de contaminantes secundarios tales como el ozono o partículas PM2.5.
    - Monóxido de nitrógeno (NO), que es incoloro, inodoro, no inflamable y tóxico que se oxida con rapidez convirtiéndose en NO2. Al igual que en el caso anterior, sus principales fuentes son tanto naturales (descomposición bacteriana, incendios, etc.) como derivadas de la actividad humana (vehículos motorizados y quema de combustibles fósiles) [11].

- El CO es un gas inodoro, incoloro, tóxico e inflamable. Tiene un período de vida de entre 30 y 90 días y es uno de los contaminantes atmosféricos más abundantes y con mayor distribución. Su origen puede ser tanto natural como antropogénico. Así y de entre sus principales fuentes, cabe citar la quema de combustibles fósiles (vehículos con motor de combustión, industria, etc.) y biomasa (madera). Reduce la capacidad de la sangre para transportar oxígeno a los tejidos corporales, comportando un grave riesgo para personas con

- El ozono es un contaminante secundario que, cuando se localiza en la troposfera, es decir, entre la superficie terrestre y los 10-12 primeros kilómetros de la atmósfera, genera un grave efecto sobre la salud humana y el entorno. El ozono troposférico se forma por la reacción fotoquímica de los precursores, sustancias emitidas de forma directa tales como los NOx, el CO o los COVs, que reaccionan con la luz solar en condiciones atmosféricas estables

(temperaturas altas y viento escaso). Su impacto sobre la salud es notable, ya que tiene un marcado carácter oxidativo que le capacita para destruir incluso órganos completos.

2.3 Variables meteorológicas

Se ha demostrado que hay una relación muy estrecha entre las condiciones meteorológicas y los niveles de contaminación, debido a que la meteorología es la ciencia de la atmósfera, sobretodo en su distribución y dispersión.

Los procesos atmosféricos tales como el movimiento del aire y el intercambio de calor determinan el destino de los contaminantes a medida que pasan por las etapas de transporte, dispersión, transformación y remoción. La meteorología de la contaminación del aire es el estudio de cómo estos procesos atmosféricos afectan el destino de los contaminantes del aire [12].

El conocimiento de la meteorología de la contaminación del aire sirve para manejar y controlar la descarga de contaminantes en el aire exterior. El control de la descarga de estos contaminantes ayuda a asegurar que las concentraciones de este tipo de sustancias en el ambiente cumplan con los estándares de calidad del aire en exteriores. Además, este conocimiento es esencial para entender el destino y transporte de las sustancias contaminantes del aire [12].

Para tener un control sobre la calidad del aire es indispensable comprender las variables meteorológicas y su influencia sobre los contaminantes. La meteorología se utiliza para prevenir el efecto ambiental de una nueva fuente de emisión de contaminación del aire y determinar el impacto de las transformaciones de las fuentes existentes en la calidad del aire.

Cuando se desarrollan condiciones meteorológicas que no conducen a la dispersión de las sustancias contaminantes, los organismos gubernamentales encargados de controlar la contaminación del aire deben actuar rápidamente para asegurar que los contaminantes no se concentren en niveles inaceptables en el aire que respiramos. Cuando estos niveles son excesivamente altos, se produce un caso de contaminación del aire y se deben reducir las emisiones en la atmósfera [12].

2.4 Sensores del tráfico

Los sistemas de sensores de tráfico para medir la intensidad del tráfico y registrar el volumen de vehículos que hay en las vías.

Dentro de los sensores de tráfico se puede realizar una nueva división en base a la colocación de los sensores y la necesidad, o no de dispositivos embarcados en los vehículos. Por tanto se diferenciará entre dos tipos:

• Sensores de tráfico autónomos: no requieren un dispositivo embarcado en los vehículos y el elemento sensor está situado en la infraestructura.

• Sensores de tráfico dependientes: o el elemento sensor está en la infraestructura y requiere la presencia de un dispositivo embarcado en el vehículo o el/los elementos sensores están en el vehículo [13].

Dentro de los sensores autónomos hay dos tipos: las tecnologías intrusivas, instaladas en el pavimento, y las no intrusivas, están encima o al lado de la carretera para que no interrumpa el flujo del tráfico.

Dentro de los sensores intrusivos se encuentran los siguientes:

- Las espiras magnéticas son los sensores más extendidos en las carreteras españolas ya que se trata de una tecnología barata y muy desarrollada, con un funcionamiento simple que no se ve afectado por las condiciones ambientales.

- Los tubos neumáticos son tubos de goma que se emplazan a lo largo del pavimento sobre la calzada y que son capaces de detectar el paso del vehículo por el cambio de presión que éste ejerce sobre el aire contenido en el tubo al pasar sobre el mismo.

- Los sensores piezo-eléctricos su detección se realiza en base a la presión que genera el vehículo al pasar sobre ellos. Detectan el paso del vehículo en base a la carga eléctrica que se genera en el material piezoeléctrico cuando es pisado por una rueda y éste se deforma.

- Los sensores de fibra óptica, al igual que los dos sensores explicados con anterioridad, basan la detección en la presión realizada por el vehículo al pasar sobre el sensor [13].

Los detectores no intrusivos empleados en la actualidad son, principalmente de dos tipos:

- Sensores activos: aquellos que emiten una señal y captan la respuesta reflejada sobre el vehículo. De este tipo son los radares de microondas, radares láser y los sensores ultrasónicos.

- Sensores pasivos: este tipo de sensores capta variaciones producidas, en ciertos parámetros, por el paso de un vehículo. Sensores pasivos son las cámaras de vídeo, los sensores infrarrojos y los sensores acústicos [13].

Dentro de los sensores activos hay distinto tipos como hemos visto antes:

- Los radares microondas su detección del vehículo del radar se consigue a través de la emisión de una señal de microondas. Estas son reflejadas por objetos y el radar calcula el tiempo transcurrido entre la emisión de la señal y la recepción de la señal reflejada.

- La detección del vehículo a través del radar láser, también conocido como LIDAR, se realiza a través de la emisión de un pulso lumínico láser (en el infrarrojo cercano) para medir la distancia a la que se encuentra un determinado objeto. Para calcular dicha distancia, se mide el tiempo que tarda el pulso lumínico en viajar desde el LIDAR hasta el vehículo y volver.

- Los sensores ultrasónicos transmiten ondas sonoras a una frecuencia entre 25 y 50 kHz, siempre por encima del rango auditivo humano. Se puede observar un esquema de

colocación típico en la siguiente figura con un ejemplo de montaje sobre la calzada y en un lateral [13].

Dentro de los sensores pasivos nos encontramos los siguientes:

- El funcionamiento de los sensores acústicos se basa en la detección de la energía acústica (sonidos audibles) producida por el tráfico vehicular (ruido del vehículo y contacto de las ruedas del vehículo con la calzada). Cuando un vehículo atraviesa la zona de detección se produce un aumento de la energía acústica y esta energía es convertida en señal eléctrica.

- Los sistemas video se introdujeron en la gestión de autopistas para labores de vigilancia remota por parte de un operario físico. Sin embargo, en la actualidad se emplean técnicas de visión artificial para extraer, de manera automática, distintos tipos de información [13].

Dentro de los sensores dependientes hay dos tipos.

- La identificación por radiofrecuencia (RFID) es un sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados etiquetas, transpondedores o tags RFID. El propósito fundamental de esta tecnología es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio.
- Un Sistema Global de Navegación por Satélite es una constelación de satélites que transmite rangos de señales utilizados para el posicionamiento y localización en cualquier parte del globo terrestre, ya sea en tierra, mar o aire (en este documento su utilidad principal se refiere al posicionamiento en tierra, concretamente en autopistas)

Capítulo 3

Análisis y preparación de los datos

3.1 Introducción

Hay un total de 10 estaciones que miden la contaminación que se distinguen en tres tipos que son los siguientes:

- Las estaciones de fondo urbano son seis Palau Reial, Zona Universitària, Ciutadella, Poblenou y Vall d'Hebron.
- En las estaciones del suburbano sólo hay una, el IES Goya.
- Las estaciones de tráfico son tres: Eixample, Universitat y Gràcia-Sant Gervasi.

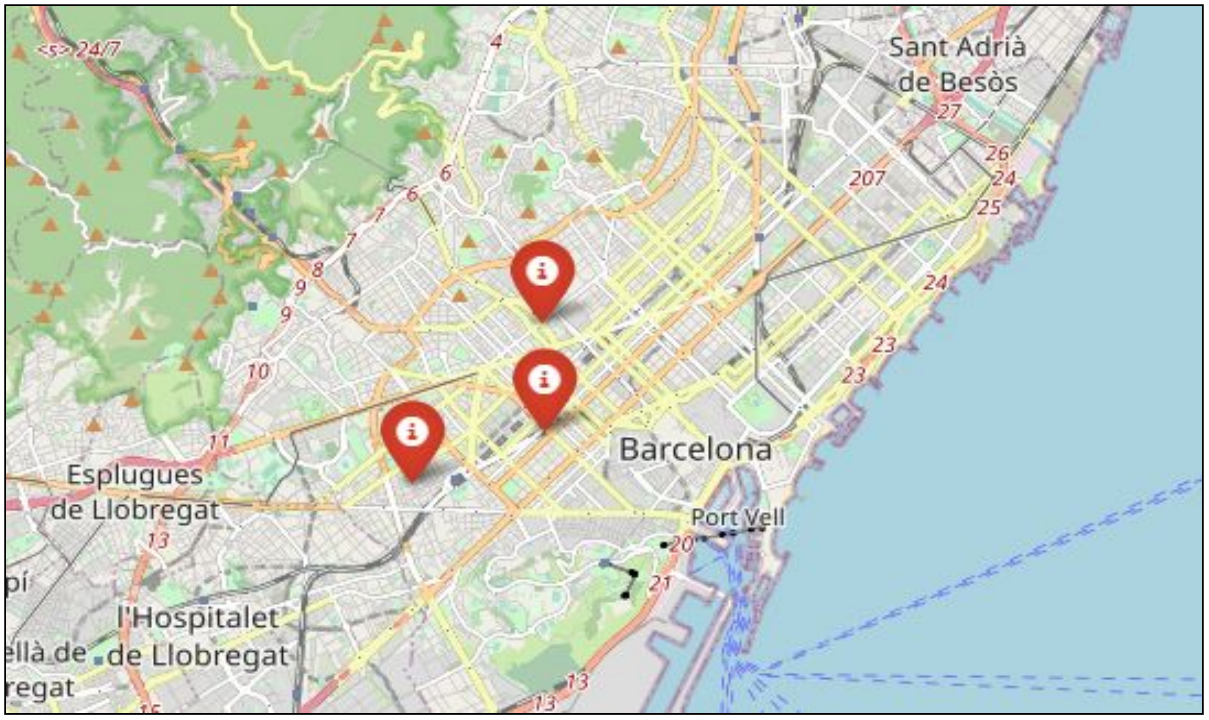
Estas estaciones facilitan los datos de contaminación para cada hora del día con su correspondiente validación.

3.2 Selección de los contaminantes, variables meteorológicas y sensores de tráfico

3.2.1 Selección de los contaminantes

Las diez estaciones de medición de contaminación de Barcelona calculan distintos niveles de contaminantes y tienen su correspondiente validación para cada hora, por lo que son datos con un nivel alto de calidad. Los datos de contaminación se comprenden en un período de tres años desde enero del 2017 hasta diciembre del 2019. Para hacer este proyecto se ha seleccionado sólo tres estaciones de las diez estaciones que hay en Barcelona, porque no había datos de todos los contaminantes en todas las estaciones. Esta estación es Sants, que sólo tiene mediciones de los óxidos de nitrógeno de esta manera se ha acotado el proyecto. Después de hacer este paso, se ha seleccionado las dos estaciones que tienen esta familia de contaminantes que son el Eixample y Sant Gervasi-Gràcia.

Con esta selección hemos descargado un csv con todos los datos de los contaminantes y luego los hemos explorado.



3.2.2 Selección de las variables meteorológicas

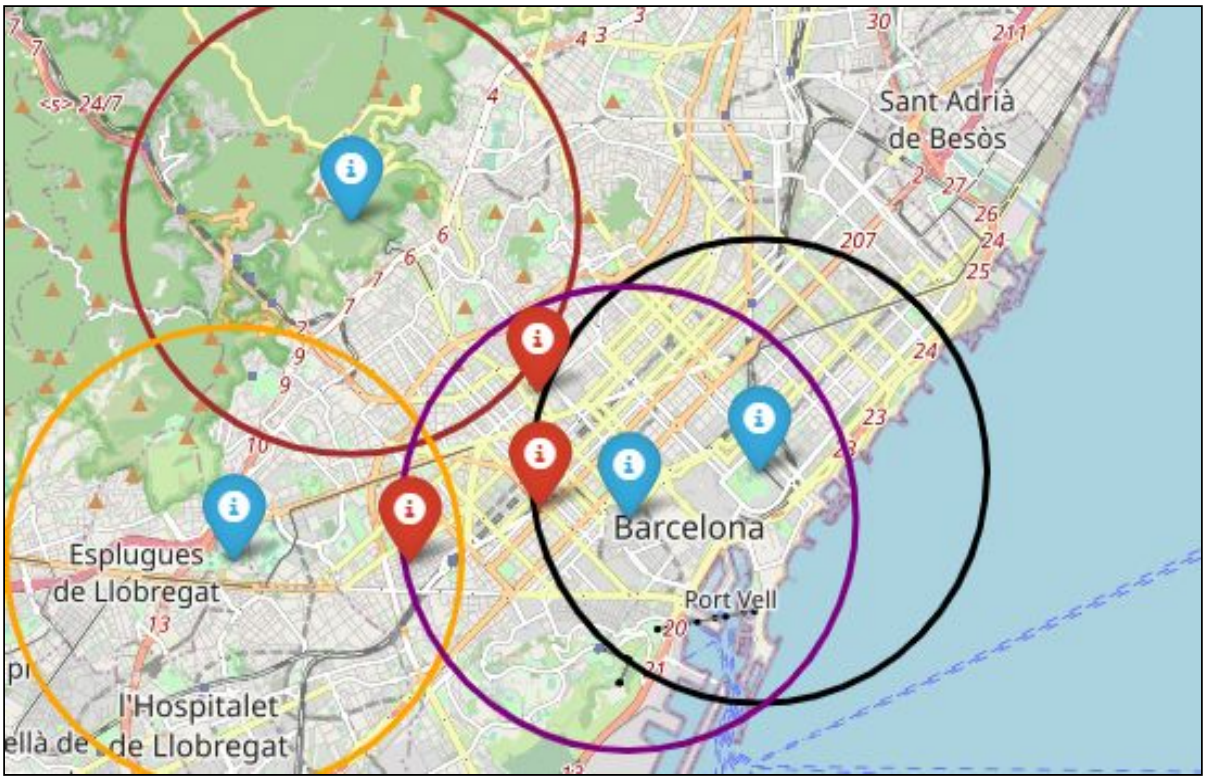
Las condiciones meteorológicas influyen en la contaminación y en su dispersión, por eso se han seleccionado cinco variables meteorológicas, que a partir de la búsqueda en diversos estudios, se ha observado que son las más influyentes.

Estas son la temperatura, la precipitación, la presión, la humedad relativa y el viento. Se destacan dos variables: el viento por la dispersión de los contaminantes y la temperatura que provoca efectos como el efecto invernadero que aumentan el nivel de contaminación y la inversión térmica.

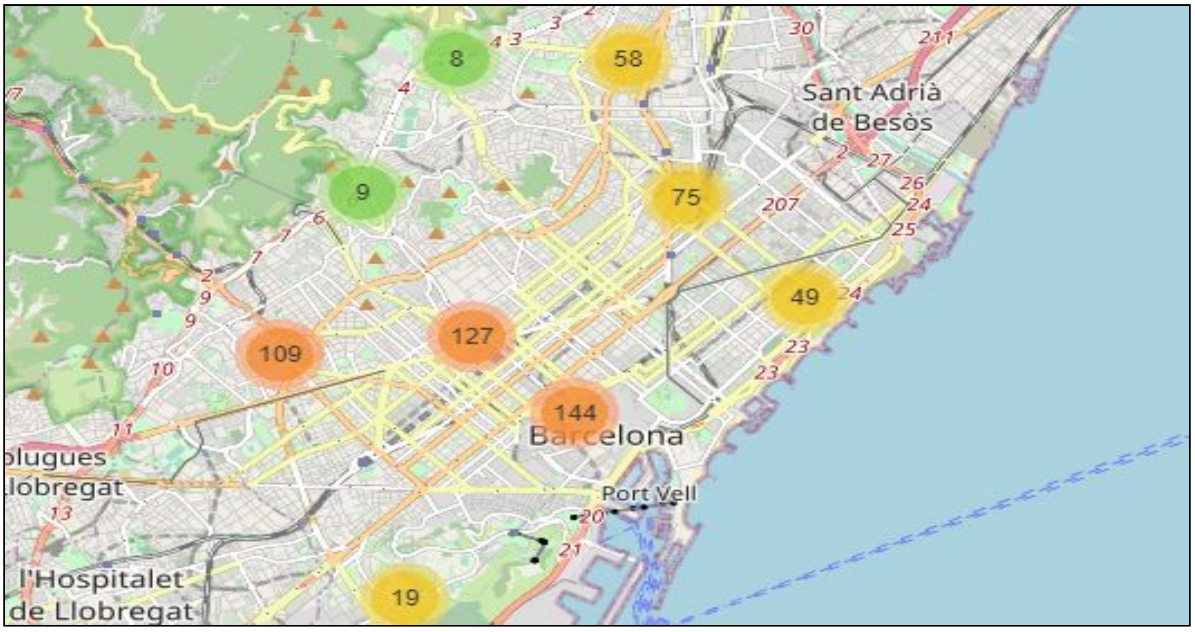
A continuación, tuvimos que elegir entre las estaciones meteorológicas que hay en Barcelona. Hay cuatro estaciones, teniendo en cuenta las estaciones de medición de la contaminación, se han acotado las estaciones.

Se ha creado un mapa con la librería folium, donde se han creado unos círculos con un radio de 3000 para hacer esta selección.

El resultado es que las estaciones meteorológicas de Zona Universitaria y el Raval se relacionan con la estación de contaminantes de Sants. Las estaciones meteorológicas del Raval y la situada en el Zoo van relacionada con la estación de contaminación del Eixample. Y finalmente, la estación meteorológica del Raval va relacionada con la estación de contaminantes de Sant Gervasi-Gràcia. De esta forma cuando se han hecho los modelos se han realizado de esta manera. A partir de este acotamiento se ha descargado un csv para poder trabajar luego con él.



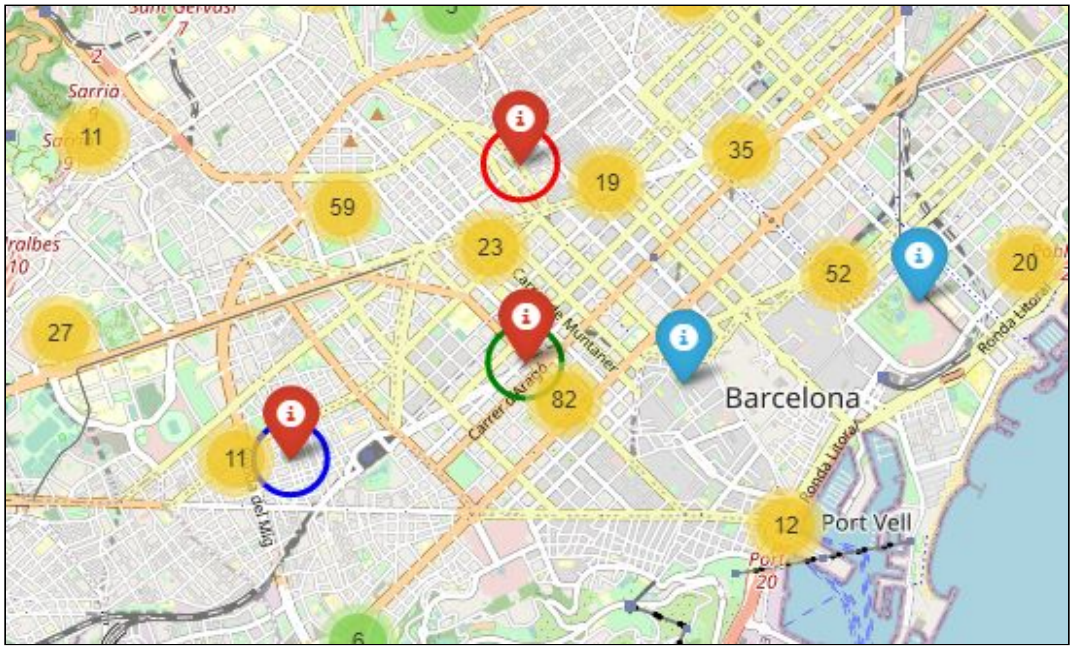
3.2.3. Selección de los sensores de tráfico



Otro de los factores claves que se tiene en cuenta son los sensores de tráfico, para cuantificar el tráfico en relación a la contaminación y cuánta implicación tiene en los niveles de contaminación.

Barcelona tiene más de 600 sensores de tráfico y se debe acotar la selección. Para hacerlo se ha procedido a crear unos círculos con un radio de 270, donde el centro de los círculos son las estaciones de contaminantes que se han seleccionado.

Con los círculos creados se han seleccionado los sensores para las distintas estaciones. Para la estación del Eixample se ha relacionado con los sensores 4071, 4095 y 20013. A continuación, la estación de Sants se relaciona con 2020 y 2021. Y por último, la estación de Sant Gervasi-Gràcia se relaciona con los sensores 8001 y 8039.

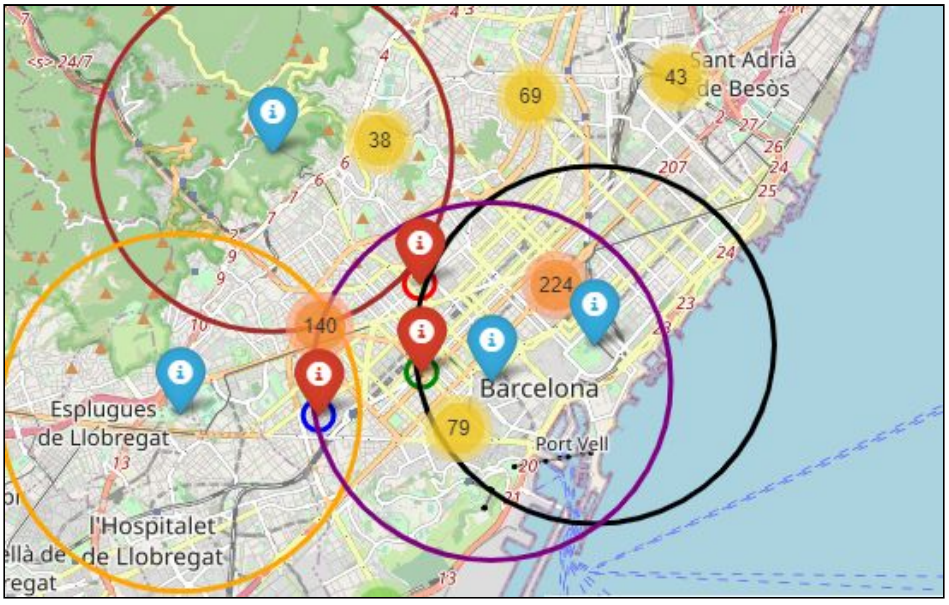



```
from folium.plugins import MarkerCluster

some_map2 = folium.Map(location=[subset_of_data1['Latitud'].mean(), subset_of_da

mc= MarkerCluster()

for row in subset_of_data1.itertuples():

    mc.add_child(folium.Marker(location= [row.Latitud, row.Longitud],popup=row.I
```

Con las líneas anteriores, se han creado los mapas. Con la primera línea, se importa folium.plugins para poder unir los puntos de los sensores de tráfico. Anteriormente, se ha importado la librería de folium para poder realizar los mapas. La segunda línea es una muestra para poder reducir el número de los sensores y unirlos. Y en la tercera, se crea un markercluster y se añade la información que se quiera.

```
marcador1 = folium.Marker(location=(41.398722, 2.153399),popup= 'Gràcia', icon=f
marcador2 = folium.Marker(location= (41.378783, 2.133099), popup= 'Sants', icon=
marcador3= folium.Marker(location=(41.385314, 2.153800),popup= 'Eixample', icon=
marcador4= folium.Marker(location=(41.41843, 2.12388),popup= 'Fabra', icon=foliu
marcador5= folium.Marker(location=(41.38943, 2.18847),popup= 'Zoo', icon=folium.
marcador6= folium.Marker(location=(41.37919, 2.1054),popup= 'Zona Universitaria'
marcador7= folium.Marker(location=(41.3839, 2.16775),popup= 'El Raval', icon=fol
circulo = folium.Circle([41.398722, 2.153399], radius=270, popup='Gràcia', color
circulo_2= folium.Circle([41.378783, 2.133099], radius=270, popup='Sants', color
circulo_3= folium.Circle([41.385314, 2.153800], radius=270, popup='Eixample', co
circulo_4= folium.Circle([41.41843, 2.12388], radius=3000, popup='Eixample', col
circulo_5= folium.Circle([41.38943, 2.18847], radius=3000, popup='Eixample', col
circulo_6= folium.Circle([41.37919, 2.1054], radius=3000, popup='Eixample', colo
circulo_7= folium.Circle([41.3839, 2.16775], radius=3000, popup='Eixample', colo
marcador1.add_to(some_map2)
marcador2.add_to(some_map2)
marcador3.add_to(some_map2)
marcador4.add_to(some_map2)
marcador5.add_to(some_map2)
marcador6.add_to(some_map2)
marcador7.add_to(some_map2)
```

Con las líneas anteriores se crean siete marcadores, uno por cada estación meteorológica y uno por cada estación de contaminantes. También se puede ver que hay, los siete círculos que se han creado para la elección de las estaciones meteorológicas y para los sensores de tráfico.



Por tanto, las relaciones con las estaciones de contaminación, sus contaminantes, las estaciones meteorológicas, su correspondientes variables meteorológicas y los sensores relacionados con cada estación de contaminación quedan de la siguiente manera:

- Sants
  - NO
    - Raval
      - Temperatura
      - Viento
      - Humedad Relativa
      - Precipitación
      - Presión
    - Zona Universitaria
      - Temperatura
      - Viento
      - Humedad Relativa
      - Precipitación
      - Presión
    - Sensores de tráfico
      - 2020
      - 2021
  - NO2
    - Raval
      - Temperatura
      - Viento
      - Humedad Relativa
      - Precipitación
      - Presión
    - Zona Universitaria
      - Temperatura
      - Viento
      - Humedad Relativa
      - Precipitación
      - Presión
    - Sensores de tráfico
      - 2020
      - 2021
  - NOX
    - Raval
      - Temperatura
      - Viento
      - Humedad Relativa
      - Precipitación
      - Presión
    - Zona Universitaria
      - Temperatura
      - Viento
      - Humedad Relativa
      - Precipitación
      - Presión
    - Sensores de tráfico
      - 2020
      - 2021
- Eixample
  - NO
    - Raval
      - Temperatura
      - Viento
      - Humedad Relativa
      - Precipitación
      - Presión
    - Zoo

            - Temperatura
            - Humedad Relativa
            - Presión
        - Sensores de tráfico
            - 4071
            - 4095
            - 20013
    - NO2
        - Raval
            - Temperatura
            - Viento
            - Humedad Relativa
            - Precipitación
            - Presión
        - Zoo
            - Temperatura
            - Humedad Relativa
            - Presión
        - Sensores de tráfico
            - 4071
            - 4095
            - 20013
    - NOX
        - Raval
            - Temperatura
            - Viento
            - Humedad Relativa
            - Precipitación
            - Presión
        - Zoo
            - Temperatura
            - Humedad Relativa
            - Presión
        - Sensores de tráfico
            - 4071
            - 4095
            - 20013
- Sant Gervasi-Gràcia
    - NO
        - Raval
            - Temperatura
            - Viento
            - Humedad Relativa
            - Precipitación
            - Presión
        - Sensores de tráfico
            - 8001
            - 8039
    - NO2
        - Raval
            - Temperatura
            - Viento
            - Humedad Relativa
            - Precipitación
            - Presión
        - Sensores de tráfico
            - 8001
            - 8039
    - NOX
        - Raval
            - Temperatura
            - Viento
            - Humedad Relativa
            - Precipitación

        - Presión
    - Sensores de tráfico
        - 8001
        - 8039

Estas son las relaciones que se han seguido para hacer todos los modelos.

3.3 Procedencia de los datos

3.3.1 Datos sobre los datos de la contaminación

Para hacer este proyecto se han utilizado datos de la contaminación de Barcelona, obtenidos en el portal Datos abiertos de Catalunya.

Los datos de la contaminación están en dos formatos, en horas y días. Los datos están clasificados por año, estación, área urbana y municipio. Este archivo se ha descargado en csv. Estos datos contienen:

- Año
- Dia
- Mes
- Data
- Longitud
- Latitud
- Nombre estación
- Horas, todas las horas del día en columnas.
- Contaminante
- Código municipio
- Código measurement

Para el proyecto se han utilizado los datos de fecha, hora, año, día, mes, nombre estación y los tres contaminantes de cada estación.

3.3.2 Datos sobre las variables meteorológicas

Las variables meteorológicas están muy relacionadas con la dispersión y la generación de la contaminación atmosférica. El conocimiento de los factores climáticos es clave para este proyecto.

Los datos recopilados de las condiciones meteorológicas de Barcelona provienen de los datos abiertos de Catalunya, en un período de tres años como se ha dicho anteriormente. Este archivo se ha descargado en csv, estos datos contienen:

- Código estación
- Código variable
- Data lectura
- Valor lectura

- Código estado
- Código base

Los parámetros meteorológicos escogidos relacionados con la contaminación son la temperatura, humedad relativa, precipitaciones, presión y el viento.

3.3.3 Datos sobre los sensores de tráficos

Para poder comprender la relación entre los niveles de tráfico de la ciudad y la contaminación, se han recopilado datos históricos desde enero de 2017 hasta diciembre de 2019. Para esto hemos recopilado información de los más de 600 sensores que hay en Barcelona para medir esta relación. Los sensores de tráfico que se utilizan en Barcelona son tres: las espiras, infrarrojos y visión artificial.

Los datos de los sensores de tráfico se han obtenido del portal de opendata BCN, se ha descargado el archivo como csv, estos datos contienen:

- Año
- Id Aforo
- Mes
- Código tipo dia
- Tipo dia
- Valor IMD

De estos datos, se ha utilizado todos los parámetros que se han comentado en el párrafo anterior.

3.4 Limpieza y preparación de los datos

3.4.1 Datos de la contaminación

Después de recopilar los datos se han limpiado y preparado. En primer lugar, se ha creado un dataframe para ver las columnas que tiene los datos y seleccionarlas.

```
data3.columns
```

```
Index(['CODI MESURAMENT', 'CODI EOI', 'PROVINCIA', 'CODI MUNICIPI',
       'CODI ESTACIÓ', 'NOM ESTACIÓ', 'MUNICIPI', 'LATITUD', 'LONGITUD',
       'ALTITUD', 'TIPUS ESTACIÓ', 'ÀREA URBANA', 'MAGNITUD', 'CONTAMINANT',
       'UNITATS', 'PUNT MOSTREIG', 'ANY', 'MES', 'DIA', 'DATA', 'H01', 'V01',
       'H02', 'V02', 'H03', 'V03', 'H04', 'V04', 'H05', 'V05', 'H06', 'V06',
       'H07', 'V07', 'H08', 'V08', 'H09', 'V09', 'H10', 'V10', 'H11', 'V11',
       'H12', 'V12', 'H13', 'V13', 'H14', 'V14', 'H15', 'V15', 'H16', 'V16',
       'H17', 'V17', 'H18', 'V18', 'H19', 'V19', 'H20', 'V20', 'H21', 'V21',
       'H22', 'V22', 'H23', 'V23', 'H24', 'V24', 'Georeferència'],
      dtype='object')
```

A continuación, se han quitado las columnas que no son necesarias para la exploración de los datos.

```
data3.drop(['TIPUS ESTACIÓ'], axis=1, inplace=True)
data3.drop(['ÀREA URBANA'], axis=1, inplace=True)
data3.drop(['MAGNITUD'], axis=1, inplace=True)
data3.drop(['UNITATS'], axis=1, inplace=True)
data3.drop(['PUNT MOSTREIG'], axis=1, inplace=True)
data3.drop(['ANY'], axis=1, inplace=True)
data3.drop(['MES'], axis=1, inplace=True)
data3.drop(['DIA'], axis=1, inplace=True)
data3.drop(['Georeferència'], axis=1, inplace=True)
```

```
data3.drop(['MUNICIPI'], axis=1, inplace=True)
data3.drop(['LATITUD'], axis=1, inplace=True)
data3.drop(['LONGITUD'], axis=1, inplace=True)
data3.drop(['ALTITUD'], axis=1, inplace=True)
data3.drop(['CODI EOI'], axis=1, inplace=True)
data3.drop(['PROVINCIA'], axis=1, inplace=True)
data3.drop(['CODI ESTACIÓ'], axis=1, inplace=True)
```

Por otra parte, se ha renombrado las columnas de las horas y de las validaciones, con un guión bajo para poder preparar los datos para tener un dataset limpio. Luego, se ha comprobado si hay valores faltantes, en los datos que quedan. Al ver que hay datos faltantes, se ha procedido hacer una media, para sustituir los valores faltantes por la media de cada columna.

A continuación, se han renombrado las columnas que quedan en el dataframe y se ha creado dos columnas, *variable* y *hora.* Después se ha creado una nueva tabla, con todas las columnas que se han seleccionado. Luego, se ha hecho una nueva columna *datetime,* importando datetime y sumando las columnas, DATA y hora. El resultado ha sido este:

```
data3 = data3.melt(id_vars= ['CODI MESURAMENT','CODI MUNICIPI', 'DATA', 'CONTAMINANT', 'NOM ESTACIÓ'])
```

```
data3[["variable", "hora"]] = data3.variable.str.split("_", expand=True)
```

```
data3 = pd.pivot_table(data3,
                       index= ['CODI MESURAMENT', 'DATA', 'CONTAMINANT','CODI MUNICIPI', 'hora', 'NOM ESTACIÓ'],
                       columns='variable', values='value', aggfunc= 'first').reset_index()
```

```
data3['datetime'] = pd.to_datetime(data3.DATA) + data3.hora.astype('timedelta64[h]')
```

```
data3.head()
```

variable	CODI MESURAMENT	DATA	CONTAMINANT	CODI MUNICIPI	hora	NOM ESTACIÓ	H	V	datetime
0	08019042_12_20171231	31/12/2017	NOX	19	01	Barcelona (Sants)	31.4026	N	2017-12-31 01:00:00
1	08019042_12_20171231	31/12/2017	NOX	19	02	Barcelona (Sants)	28.8919	N	2017-12-31 02:00:00
2	08019042_12_20171231	31/12/2017	NOX	19	03	Barcelona (Sants)	26.7548	N	2017-12-31 03:00:00
3	08019042_12_20171231	31/12/2017	NOX	19	04	Barcelona (Sants)	25.1911	N	2017-12-31 04:00:00
4	08019042_12_20171231	31/12/2017	NOX	19	05	Barcelona (Sants)	23.555	N	2017-12-31 05:00:00

Después se han quitado las columnas de DATA y hora. A continuación, se han reemplazado los nombres de las estaciones por valores numéricos, para poder seleccionar después los distintos datasets de cada estación. Cuando se han seleccionado los tres datasets de cada estación, se debe seleccionar los tres contaminantes para cada estación. La distribución de los datasets está formado de la siguiente manera:

- Sants
    - NOX
    - NO2
    - NO
- Eixample
    - NOX
    - NO2
    - NO
- Sant Gervasi-Gràcia
    - NOX
    - NO2
    - NO

Para que estos datasets fueran series temporales se han ordenado primero, los valores de datetime y luego, se han reemplazado los valores de False por True, debido a que se han cambiado cuando se ha hecho el pivot table. A continuación, la columna de *H* se ha transformado a float para poder hacer las series temporales; después se ordena por la columna de *H*; se agrupa por el *código de municipio* y la columna de *datetime y,* a continuación, agregamos una media del head de esa columna, para luego resetear el índice de la serie. Después, agrupamos otra vez por el código del municipio, a continuación, usamos la función rolling para hacer cálculos de ventana para poder crear la serie temporal. Se vuelve a resetear y se elimina la columna del código de municipio, luego se hace un set_index con la columna datetime como índice. Por último, se agrupa por data y se agrega en la columna *H*, el máximo. El código queda de la siguiente manera:

```
data_Sants_NO = dataSants[dataSants.CONTAMINANT == 'NO']

data_Sants_NO = data_Sants_NO.sort_values(['datetime'])

data_Sants_NO.replace(to_replace= False, value=True, inplace=True)

data_Sants_NO['H'] = data_Sants_NO['H'].astype(float)

seriecont_San_NO = data_Sants_NO[data_Sants_NO.V == True]
seriecont_San_NO = seriecont_San_NO .sort_values('H', ascending=False)
seriecont_San_NO  = seriecont_San_NO .groupby(['CODI MUNICIPI', 'datetime'])
seriecont_San_NO  = seriecont_San_NO .agg({'H': lambda x: x.head(2).mean()})
seriecont_San_NO = seriecont_San_NO .reset_index(drop=False)
seriecont_San_NO = seriecont_San_NO .groupby('CODI MUNICIPI')
seriecont_San_NO  = seriecont_San_NO .rolling(window= 2, on='datetime').mean()
seriecont_San_NO  = seriecont_San_NO .drop('CODI MUNICIPI', axis=1).reset_index()
seriecont_San_NO  = seriecont_San_NO .set_index('datetime')
seriecont_San_NO = seriecont_San_NO .groupby(lambda x: x.date)
seriecont_San_NO  = seriecont_San_NO .agg({'H':max})
```

Esto se ha repetido por los nueve datasets anteriormente nombrados, para poder conseguir las series temporales de los distintos contaminantes para cada estación.

3.4.2 Datos de las variables meteorológicas

Después de recopilar los datos viene la limpieza y la preparación de los datos para llegar a las series temporales.

Primero de todo, se ha creado un dataframe y se ha mirado que columnas tiene para ver cuales son útiles. Luego, se ha quitado la columna de *data extrem* y, a continuación, se ha ordenado el dataframe por la columna de *data lectura.* Después, se ha procedido a pasar la columna de *data lectura* como pd.to_datetime, para hacerlo en formato de data. A continuación, se mira si hay valores faltantes, pero no los hay. Luego, se ha renombrado la columna de *codi_variable* a *variable_meteo.* Después, en esta columna, se ha cambiado los valores numéricos de la columna por los correspondientes valores de las variables, que hay en otro archivo descargado con la descripción de los valores de las variables.

A continuación, se ha seleccionado cada dataset por variable meteorológica y por estación, la distribución de la datasets queda de la siguiente manera:

- Raval
    - Viento
    - Temperatura
    - Humedad relativa
    - Precipitación
    - Presión
- Zoo
    - Temperatura
    - Humedad relativa
    - Presión
- Zona Universitaria
    - Viento
    - Temperatura
    - Humedad relativa
    - Precipitación
    - Presión

Después de seleccionar los datasets, se ha procedido hacer las series temporales como se ha hecho anteriormente con las series temporales de la contaminación.

```
seriecontvent_zonauniversitaria = datavent_zonauniversitaria[datavent_zonauniversitaria.CODI_ESTAT == True]
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.sort_values('VALOR_LECTURA', ascending=False)
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.groupby(['CODI_BASE', 'DATA'])
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.agg({'VALOR_LECTURA': lambda x: x.head(1).mean()})
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.reset_index(drop=False)
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.groupby('CODI_BASE')
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.rolling(window= 1, on='DATA').mean()
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.drop('CODI_BASE', axis=1).reset_index()
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.set_index('DATA')
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.groupby(lambda x: x.date)
seriecontvent_zonauniversitaria = seriecontvent_zonauniversitaria.agg({'VALOR_LECTURA':max})
```

Se han creado las series temporales de las variables meteorológicas de igual forma que las series temporales de la contaminación. Para obtener los datasets de las variables meteorológicas, se ha creado la misma línea de código para cada estación y variable.

3.4.3 Datos de los sensores de tráfico

A continuación de la recopilación de los datos, se tiene que limpiar y preparar para explorarlos de la manera más adecuada.

En primer lugar, se han descargado tres csv de cada año de los valores numéricos y, a continuación, se ha creado un dataframe conjunto de los tres csv. Después, se ha mirado si hay valores faltantes y hemos comprobado que dentro del dataframe hay valores de medida no disponible. Luego, se ha renombrado la medida no disponible a zero y después a np.nan, anteriormente, se ha importado numpy. A continuación, se ha hecho la media en la columna de *Valor_IMD,* que es la columna donde había estos valores para después sustituirlos.

En segundo lugar, se ha creado una variable *fechas.* A partir de la suma de las columnas del dataframe que se ha creado, que son *Año*, *Mes* y *Código tipo dia.* Luego, se ha creado una columna *data,* con la variable *fechas* que se ha creado anteriormente. Después, se han borrado las columnas de *Mes*, *Codi_tipus_dia* y *Desc_tipus_dia.*

```
transit_total['Valor_IMD'].replace(to_replace='Mesura no disponible', value=0, inplace=True)

transit_total['Valor_IMD'].replace(to_replace=0, value=np.nan, inplace=True)

mean_IMD = transit_total['Valor_IMD'].astype(float).mean(skipna=True)

transit_total['Valor_IMD'].replace(to_replace=np.nan, value=mean_IMD, inplace=True)

fechas = transit_total['Any'].astype(str) + '-' + transit_total['Mes'] + '-' + transit_total['Codi_tipus_dia'].astype(str)

transit_total['VALIDO'] = 'V'

transit_total['VALIDO'] = transit_total.VALIDO == 'V'

transit_total['data'] = pd.to_datetime(fechas)

transit_total.drop(['Mes'],  axis=1, inplace=True)

transit_total.drop(['Codi_tipus_dia'], axis=1, inplace=True)

transit_total.drop(['Desc_tipus_dia'], axis=1, inplace=True)
```

A continuación, se debe seleccionar los id de los sensores de tráfico para proceder a crear los datasets. Con el dataframe que se ha trabajado hasta ahora, no tenemos todos los días, se debe crear una nueva columna para crear todos los días:

```
ultimo_fecha = max(transit_4071.index)
ultimo_dia_mes = calendar.monthrange(ultimo_fecha.year, ultimo_fecha.month)[1]
inicio = min(transit_4071.index).replace(day=1)
fin = ultimo_fecha.replace(day=ultimo_dia_mes)

index =pd.DatetimeIndex(start=inicio, end=fin, freq= '24h')
transit_4071 = transit_4071.reindex(index)
```

Así con todos los datasets que hemos creado. Los datasets de los sensores de tráfico se distribuyen de la siguiente manera:

- Sensores de tráfico
    - 20013
    - 2020
    - 2021
    - 4071
    - 4095
    - 8001
    - 8039

Después de crear las fechas, se ha creado una media para el *Valor_IMD* para reemplazarlo por np.nan. A continuación, se han eliminado las columnas de Año, Id aforo y Válido.

```
mean_IMD_1 = transit_4071['Valor_IMD'].astype(float).mean(skipna=True)

transit_4071['Valor_IMD'].replace(to_replace=np.nan, value=mean_IMD_1, inplace=True)

transit_4071.drop(['Any'], axis=1, inplace=True)

transit_4071.drop(['Id_aforament'], axis=1, inplace=True)

transit_4071.drop(['VALIDO'],axis=1, inplace=True)

transit_4071.index.names = ['DATA']
transit_4071= transit_4071.reset_index()

transit_4071.head()
```

	DATA	Valor_IMD
0	2017-01-01	32776
1	2017-01-02	33574
2	2017-01-03	34238
3	2017-01-04	26060
4	2017-01-05	21083

```
transit_4071.to_csv('transit-4071', index=False)
```

Así se crea las series temporales de los sensores de tráfico, se ha tenido que hacer para todos los sensores de tráfico.

Capítulo 4

Predicción de la contaminación

4.1 Introducción

En el capítulo anterior, se han creado todos los datasets que se necesita para las predicciones. Cada dataset contiene los datos de los tres contaminantes para cada estación, la intensidad de los sensores de tráfico que se ha seleccionado y las variables meteorológicas para cada estación meteorológica, desde enero 2017 hasta diciembre 2019. A partir de aquí, el objetivo es determinar la predicción de dichos contaminantes para cada estación, se ha estudiado con diferentes modelos para ver que tipo de modelo ajusta y predice mejor.

4.2 Modelos

4.2.1 Arima

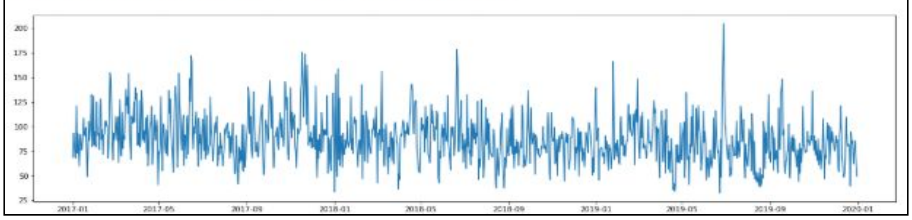
Este modelo estadístico usa las variaciones y regresiones de datos con el objetivo de hallar patrones para hacer una predicción. Es un modelo dinámico quiere decir que estas predicciones futuras se explican por los datos del pasado no por variables independientes.

Primero de todo, para ver como son los datos se ha hecho un plot para visualizarlos y ver cómo tratarlos:

```
plt.figure(figsize=(20,5))
plt.plot(data)
```

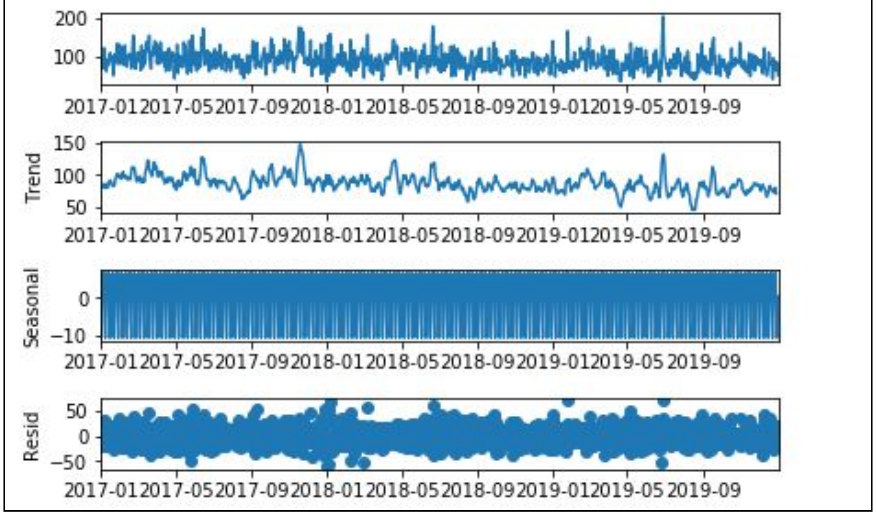
```
/home/marc/[/Usuarios/marcj/Descargas]/envs/masterds/lib/python3.7/site-packages/pandas/plotting/_matplotlib/converter.py:103: FutureWarning: Using an implicitly registered datetime converter for a matplotlib plotting method. The converter was registered by pandas on import. Future versions of pandas will require you to explicitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```

```
[<matplotlib.lines.Line2D at 0x7f9a154c9f90>]
```



En segundo lugar, importamos de la librería de statsmodels, el módulo seasonal descompose. A continuación, se realizó otro plot descomponiendo los datos:

```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(data)
fig= decomposition.plot()
```



A continuación, se ha creado un modelo ARIMA importando desde statsmodels, el módulo de Arima. Para ver cómo ajusta el modelo, se tiene que tener en cuenta que en los modelos arima no necesita features sino, que los modelos ARIMA son de los contaminantes y su respectivas estaciones.

```
from statsmodels.tsa.arima_model import ARIMA

model_conf = ARIMA(data, order= (2,1, 3))
model_fit = model_conf.fit()
model_fit.summary()
```

ARIMA Model Results

Dep. Variable:	D.H	**No. Observations:**	1094
Model:	ARIMA(2, 1, 3)	**Log Likelihood**	-4927.932
Method:	css-mle	**S.D. of innovations**	21.818
Date:	Thu, 30 Jul 2020	**AIC**	9869.863
Time:	12:33:06	**BIC**	9904.846
Sample:	01-02-2017 - 12-31-2019	**HQIC**	9883.101

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0185	0.003	-5.659	0.000	-0.025	-0.012
ar.L1.D.H	-0.4821	0.172	-2.807	0.005	-0.819	-0.146
ar.L2.D.H	0.3567	0.079	4.524	0.000	0.202	0.511
ma.L1.D.H	-0.1620	0.175	-0.927	0.354	-0.505	0.181
ma.L2.D.H	-0.9054	0.114	-7.914	0.000	-1.130	-0.681
ma.L3.D.H	0.0675	0.095	0.710	0.478	-0.119	0.254

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-1.1299	+0.0000j	1.1299	0.5000
AR.2	2.4814	+0.0000j	2.4814	0.0000
MA.1	1.0000	+0.0000j	1.0000	0.0000
MA.2	-1.0966	+0.0000j	1.0966	0.5000
MA.3	13.5184	+0.0000j	13.5184	0.0000

A continuación, se han creado tres variables: ini_forecast es una variable donde hay un período desde principios de 2019 hasta finales y su frecuencia es en días; ndays es una variable que tiene sólo asignado un número, 310 y end_forecast es una variable que es la suma entre ini_forecast y ndays, el resultado se le resta uno.

```
ini_forecast = pd.Timestamp('2019-01-01', freq='D')
ndays= 310

end_forecast = ini_forecast + ndays -1
```

Después se crea otro modelo Arima con las variables anteriormente nombradas para poder crear un modelo que ajuste mejor.

```
model_conf = ARIMA(data[:ini_forecast-1], order=(2,1,2), freq='D')
model_fit = model_conf.fit()
forecast = model_fit.forecast(ndays)[0]
forecast = pd.DataFrame(forecast, columns=['forecast'],
                        index=data[ini_forecast:end_forecast].index)
```

```
/home/marc/[/Usuarios/marcj/Descargas]/envs/masterds/lib/python3.7/site-package
s/ipykernel_launcher.py:1: FutureWarning: Addition/subtraction of integers and
integer-arrays to Timestamp is deprecated, will be removed in a future version.
Instead of adding/subtracting `n`, use `n * self.freq`
  """Entry point for launching an IPython kernel.
```

Luego, para poder hacer un plot y obtener un dataframe del modelo se crea un diccionario, después el modelo y, a continuación, se crea un dataframe. En este caso, el modelo se ha creado dentro de un bucle, dentro de ndays. Donde se ha creado una nueva variable ini test, que es la suma de ini_forecast con d. A partir de aquí, todos los valores que se han creado se añaden al forecast daily.

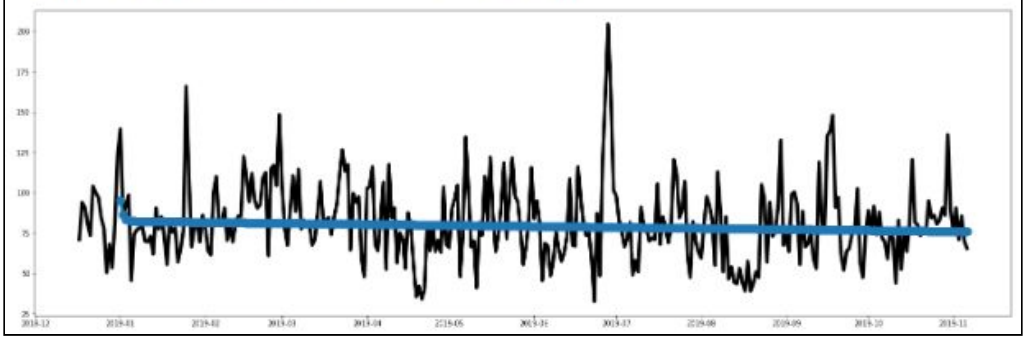
```
forecast_daily = []
for d in range (ndays):
    ini_test = ini_forecast + d
    modelconf_day =ARIMA(data[:ini_test - 1], order = (2,1,1), freq='D')
    modelfit_day = modelconf_day.fit()
    forecast_daily.append(modelfit_day.forecast(1)[0])

forecast_daily = pd.DataFrame(forecast_daily, columns=['forecast'],
                              index=data[ini_forecast:end_forecast].index)
```

A continuación, se ha hecho un plot del primer modelo, con las variables para ver si se ha ajustado bien.

```
plt.figure(figsize=(25, 8))
plt.plot(data[ini_forecast-15:end_forecast], linewidth=5, color='black')
plt.plot(forecast, linewidth=5, linestyle='--', markersize=12, marker='o')
```

```
/home/marc/[/Usuarios/marcj/Descargas]/envs/masterds/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWarning: Addition/subtraction of integers and integer-arrays to Timestamp is deprecated, will be removed in a future version.
Instead of adding/subtracting `n`, use `n * self.freq`
```

```
[<matplotlib.lines.Line2D at 0x7f9a0c5e5690>]
```

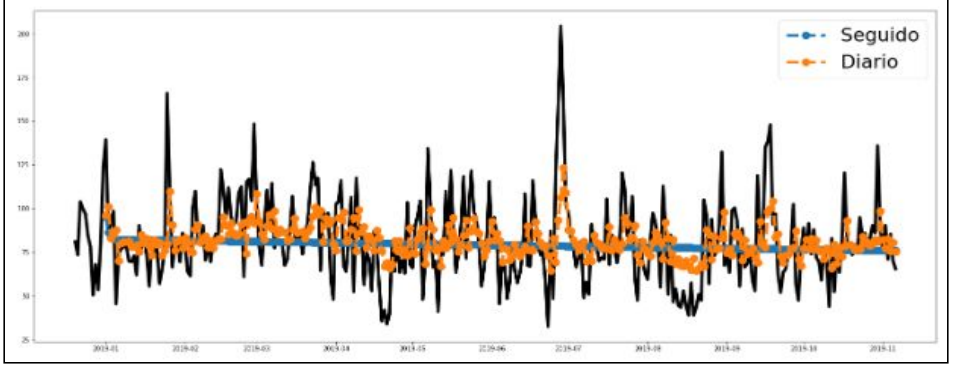


Como se puede ver, el modelo no se ajusta mucho a los datos de inicio, por eso se ha hecho otro plot del segundo modelo con forecast daily.

```
plt.figure(figsize=(25, 10))
plt.plot(data[ini_forecast-12:end_forecast], linewidth=5, color='black')
plt.plot(forecast, linewidth=5, linestyle='--', markersize=12, marker='o', label
plt.plot(forecast_daily, linewidth=5, linestyle='--', markersize= 12, marker='o'
plt.legend(prop={'size':30})
```

```
/home/marc/[/Usuarios/marcj/Descargas]/envs/masterds/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWarning: Addition/subtraction of integers and integer-arrays to Timestamp is deprecated, will be removed in a future version.
Instead of adding/subtracting `n`, use `n * self.freq`
```

```
<matplotlib.legend.Legend at 0x7f9a0c3ea9d0>
```



Se observa como el segundo modelo se ajusta mejor que el anterior, la conclusión de este modelo es que el modelo Arima sirve para tener un buen baseline.

Se ha procedido hacer lo mismo con todos los datasets de los contaminantes con sus respectivas estaciones, que se han comentado anteriormente porque no necesita features el modelo Arima.

4.2.2 Regresión lineal

La regresión lineal es un modelo matemático que se utiliza para aproximar la relación de dependencia entre las variables dependientes y las independientes y otro término incierto.

Para empezar con el modelo de regresión lineal, primero se ve la forma de la data, se observa que es un (1095, 1). A partir de aquí creamos train data y test data de la siguiente manera:

```
train_data, test_data = data.iloc[:-TEST_SIZE], data.iloc[-TEST_SIZE:]
```

Después de crear estos dataframes, se han creado x_train_data y x_test_data para poder hacer correctamente la técnica de la regresión lineal.

```
x_train_data = np.array(range(train_data.shape[0]))
x_test_data = np.array(range(train_data.shape[0], data.shape[0]))
```

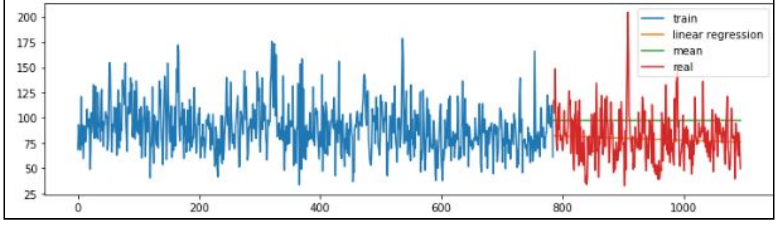
A continuación, importamos sklearn, el módulo de Linear Regression, para poder realizar la técnica de la regresión lineal. Para poder ver el ajuste de la regresión lineal, se ha hecho una media de los valores de train data. A continuación, se ha transformado el dataframe en una lista para poder introducirlo en el modelo. Después se ha creado un modelo con el Linear Regression de sklearn, se ha ajustado con los datos de train y se ha predecido con los datos de test. A continuación, se ha hecho un plot para ver cómo ajusta el linear regression.

```
from sklearn.linear_model import LinearRegression

y_pred_mean = [train_data.values[-12:].mean()] * 310

train_data = train_data.values

lr_model = LinearRegression()
lr_model.fit(x_train_data.reshape(-1,1), train_data)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_pred_lr = lr_model.predict(x_test_data.reshape(-1,1))
```

```
fig, ax = plt.subplots(1, 1, figsize=(10,3))
ax.plot(x_train_data, train_data, label='train')
ax.plot(x_test_data, y_pred_lr, label ='linear regression')
ax.plot(x_test_data, y_pred_mean, label='mean')
ax.plot(x_test_data, test_data, label ='real')
plt.legend()
plt.tight_layout()
```



Se observa la linea de color naranja, es la linear regression, como se puede ver no se ajusta a los datos, por eso motivo se ha hecho una linear regression weight. Para poder observar cómo se ajusta mejor la linear regression weight, se ha hecho el mae, mse, rmse, mape y mpe. Primero, veremos estos parámetros para la primera linear regression.

```
def mae(y_true, y_pred):
    return np.abs(y_true - y_pred).sum() / y_true.size
```

```
print('MAE in a mean model:', mae (test_data.values, y_pred_mean))
```

```
MAE in a mean model: 7586.625228519196
```

```
print('MAE in linear regression model:', mae(test_data.values, y_pred_lr))
```

```
MAE in linear regression model: 17.650571779284537
```

```
def mse(y_true, y_pred):
    return((y_true - y_pred)** 2).sum() / y_true.size
```

```
print('MSE in mean model:', mse(test_data.values, y_pred_mean))
print('MSE in linear regression model:', mse(test_data.values, y_pred_lr))
```

```
MSE in mean model: 265578.7079239553
MSE in linear regression model: 552.8469073204955
```

```
def rmse(y_true, y_pred):
    return np.sqrt(((y_true - y_pred)** 2).sum() / y_true.size)
```

```
print('RMSE in mean model:', rmse(test_data.values, y_pred_mean))
print('RMSE in linear regression model:', rmse(test_data.values, y_pred_lr))
```

```
RMSE in mean model: 515.3432913349657
RMSE in linear regression model: 23.512696725822316
```

```
def mape (y_true, y_pred):
    return 2 * np.abs((y_true - y_pred) / y_true).sum() / y_true.size
```

```
print('MAPE in mean model:', mape(test_data.values, y_pred_mean))
print('MAPE in linear regression model:', mape(test_data.values, y_pred_lr))
```

```
MAPE in mean model: 234.9252952092504
MAPE in linear regression model: 0.48192918123203243
```

```
def mpe(y_true, y_pred):
    return 100 * ((y_true - y_pred) / y_true).sum() / y_true.size
```

```
print('MPE in mean model: ', mpe(test_data.values, y_pred_mean))
print('MPE in linear regression model: ', mpe(test_data.values, y_pred_lr))
```

```
MPE in mean model:  -10032.774816684085
MPE in linear regression model:  -6.996071270899712
```

A continuación, se ha hecho la linear regression weight, donde dentro del modelo se ha creado un variable sample weight, para añadirle más peso dentro de x_train_data. A parte, el modelo se ajusta con los datos de train y se ha predecido con los datos de test.

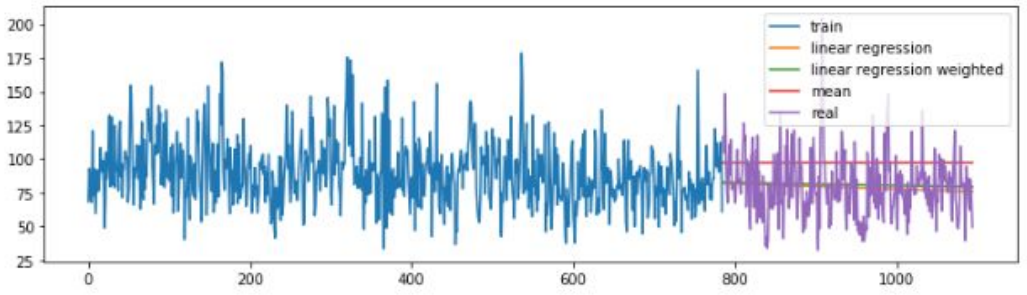
```
lrw_model = LinearRegression()
lrw_model.fit(x_train_data.reshape(-1,1), train_data,
              sample_weight= [i**5 for i in x_train_data])
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_pred_lr_w = lrw_model.predict(x_test_data.reshape(-1, 1))
```

Para poder ver la comparación con la otra linear regression, se ha hecho un plot, los parámetros mae y rmse para todos los otros modelos para poder ver la comparación entre todos.

```
fig, ax = plt.subplots(1, 1, figsize=(10, 3))
ax.plot(x_train_data, train_data, label='train')
ax.plot(x_test_data, y_pred_lr, label='linear regression')
ax.plot(x_test_data, y_pred_lr_w, label='linear regression weighted')
ax.plot(x_test_data, y_pred_mean, label='mean')
ax.plot(x_test_data, test_data, label='real')
plt.legend()
plt.tight_layout()
```




```
print('MAE in mean model: ', mae(test_data.values, y_pred_mean))
print('MAE in linear regression model: ', mae(test_data.values, y_pred_lr))
print('MAE in weighted linear regression model: ', mae(test_data.values, y_pred_

print('RMSE in mean model: ', rmse(test_data.values, y_pred_mean))
print('RMSE in linear regression model: ', rmse(test_data.values, y_pred_lr))
print('RMSE in weighted linear regression model: ', rmse(test_data.values, y_pre
```

```
MAE in mean model:  7586.625228519196
MAE in linear regression model:  17.650571779284537
MAE in weighted linear regression model:  17.75663835703781
RMSE in mean model:  515.3432913349657
RMSE in linear regression model:  23.512696725822316
RMSE in weighted linear regression model:  23.5308045232503
```

Como se puede observar, la linear regression weight no se ajusta bien a los datos, por lo que no nos quedaremos con este modelo.

Se ha hecho para todos los datasets de los contaminantes con sus respectivas estaciones.

4.2.3 Regresión polinómica

Este modelo es como el modelo de regresión lineal, pero con un número de grado de polinomio. Este modelo se ajusta a un modelo no lineal pero la función es lineal, los parámetros desconocidos se ajustan a partir de los datos, esto provoca que no sea lineal.

Para esta técnica, se tiene en cuenta que se puede poner features. Se ha creado un dataframe con las correspondientes features de las variables meteorológicas y, de los sensores de tráfico para cada instante y su correspondiente valor de contaminación. Para tener valores en el

mismo intervalo, y que no tenga los modelos màximos muy dispersos en algunas variables. Se ha procedido hacer un log de estas variables. Estas son los sensores de tráfico, las variables de humedad relativa del Zoo, Zona Universitaria, la presión de Zona Universitaria y el Raval. La columna *DATA* que contiene la fecha, se ha convertido en entero para poder tener más libertad para su exploración.

```
prueba = pd.merge(data, vent_raval, on = 'DATA')

prueba = pd.merge(prueba, temp_raval, on = 'DATA')

prueba = pd.merge(prueba, hum_raval, on = 'DATA')

prueba = pd.merge(prueba, pre_raval, on = 'DATA')

prueba = pd.merge(prueba, prep_raval, on = 'DATA')

prueba = pd.merge(prueba, temp_zoo, on = 'DATA')

prueba = pd.merge(prueba, hum_zoo, on = 'DATA')

prueba = pd.merge(prueba, pre_zoo, on = 'DATA')

prueba = pd.merge(prueba, trans_4095, on= 'DATA')

prueba = pd.merge(prueba, trans_4071, on = 'DATA')

prueba = pd.merge(prueba, trans_20013, on = 'DATA')
```

```
prueba['DATA'] = prueba['DATA'].astype(int)

prueba['DATA'] = np.arange(1094)

prueba.head()
```

	DATA	H	VENT_RAVAL	TEMP_RAVAL	HUM_RAVAL	PRE_RAVAL	PREP_RAVAL	TEMP_ZOO
0	0	68.5	2.6	13.1	13.1	6.931862	0.0	12.7
1	1	93.0	3.0	15.7	15.7	6.921658	0.0	16.0
2	2	82.5	4.0	15.6	15.6	6.923530	0.0	16.5
3	3	69.5	3.2	14.9	14.9	6.920770	3.3	14.4
4	4	68.0	4.0	17.5	17.5	6.924121	0.0	18.2

```
prueba.set_index('DATA', inplace = True)
```

Cuando se hayan realizado los anteriores pasos, se debe crear la partición de train y test del dataframe. Con sus respectivas, x_train y x_test para poder realizar los modelos y, a continuación, se ha procedido hacer un values de los dataframes divididos en train y en test, para poder introducirlos dentro del modelo.

```
x_prueba = np.arange(13128)
x_prueba = np.reshape(x_prueba, (1094,12))

x_train_prueba =  x_prueba[:-TEST_SIZE]

x_test_prueba = x_prueba[-TEST_SIZE:]

prueba_train, prueba_test = prueba[:-TEST_SIZE], prueba[-TEST_SIZE:]

prueba_train_1 = prueba_train.values

prueba_test_1 = prueba_test.values
```

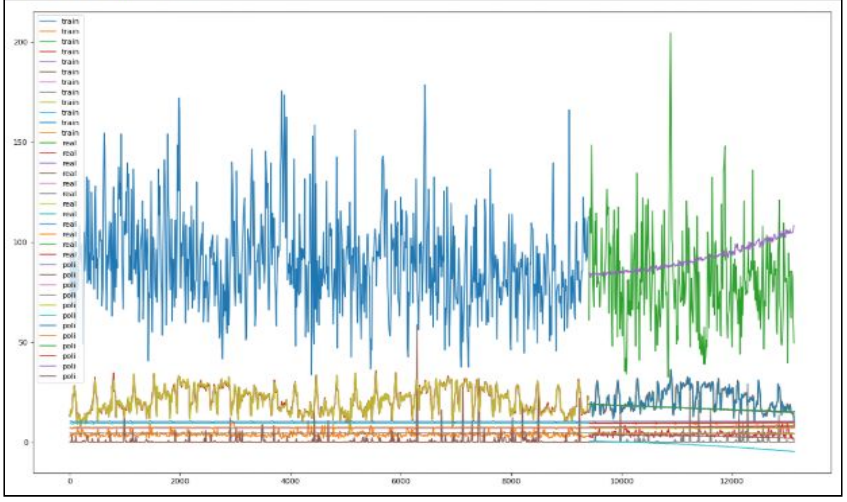
A continuación, se ha importado de sklearn, el módulo de PolynomialFeatures, para poder realizar la técnica. Después, se ha creado la variable que contiene el polinomio de tercer grado y se ha procedido hacer x_train y x_test transformados para el polinomio que se ha creado anteriormente. A continuación, se debe crear una variable con el LinearRegression para después ajustar el train con el respectivo x_train transformado con el polinomio de tercer grado. Por último, se hace un predict con los datos de test y un plot para comprobar el polinomio de tercer grado.

```
from sklearn.preprocessing import PolynomialFeatures

poli_reg = PolynomialFeatures(degree = 3)

x_train_poli = poli_reg.fit_transform(x_train_prueba)
```

```
x_test_poli = poli_reg.fit_transform(x_test_prueba)
```

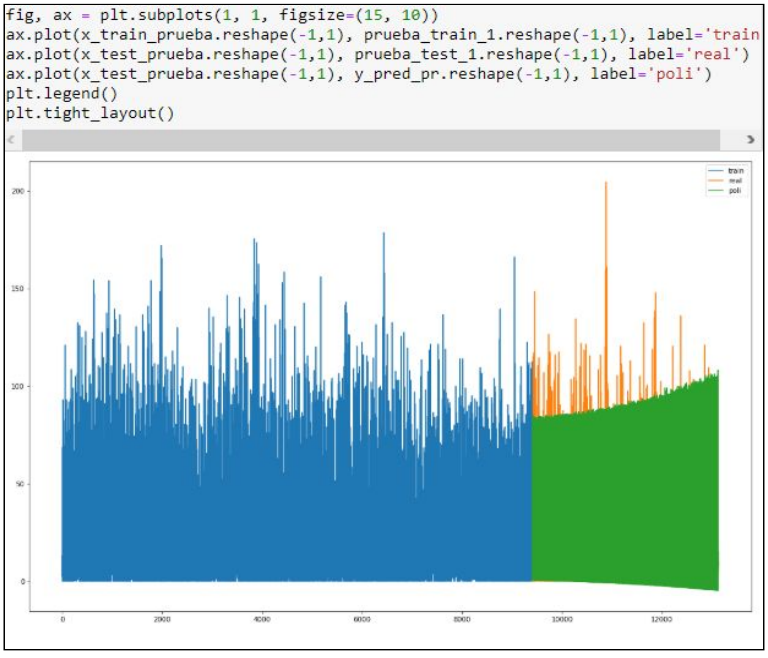
```
pr = LinearRegression()

pr.fit(x_train_poli, prueba_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

y_pred_pr = pr.predict(x_test_poli)
```

```
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
ax.plot(x_train_prueba, prueba_train_1, label='train')
ax.plot(x_test_prueba, prueba_test_1, label='real')
ax.plot(x_test_prueba, y_pred_pr, label='poli')
plt.legend()
plt.tight_layout()
```



Para tener todo en la misma curva del polinomio en el plot se debe hacer un reshape de todas las variables que lo componen.

```
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
ax.plot(x_train_prueba.reshape(-1,1), prueba_train_1.reshape(-1,1), label='train
ax.plot(x_test_prueba.reshape(-1,1), prueba_test_1.reshape(-1,1), label='real')
ax.plot(x_test_prueba.reshape(-1,1), y_pred_pr.reshape(-1,1), label='poli')
plt.legend()
plt.tight_layout()
```



Como se puede ver en el plot, el polinomio quiere ajustarse pero no lo hace de una manera correcta, por lo que continuaremos con los árboles, para ver si pueden ajustarse de la manera correcta.

Como en los anteriores apartados, se ha hecho para los correspondientes contaminantes con las estaciones, para las variables meteorológicas y los sensores que van relacionados que se ha comentado anteriormente.

4.2.4 Árboles de decisión

Para entender los árboles de decisión, primero tenemos que entender el árbol. El algoritmo del árbol de decisión pertenece a la familia de los algoritmos de aprendizaje supervisados.

A diferencia de otros algoritmos de aprendizaje supervisados, el algoritmo del árbol de decisión también se puede utilizar para resolver problemas de regresión y de clasificación. El objetivo de utilizar un árbol de decisión es crear un modelo de train, que se pueda utilizar para predecir la clase o el valor de la variable objetivo, mediante el aprendizaje de las reglas de decisión simples inferidas de datos anteriores. Los árboles de decisión para predecir una etiqueta de clase para un registro, se parte de la raíz del árbol. En base a la comparación, se sigue la rama correspondiente a este valor y salta al siguiente nodo.

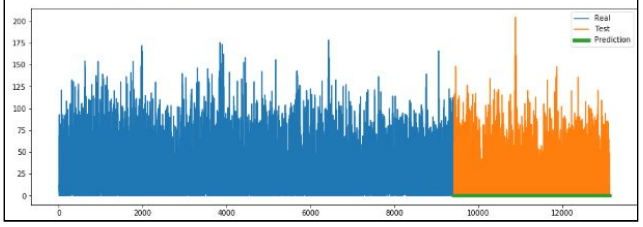
Hay distintos tipos de árboles de decisión, que se basan en el tipo de variable objetivo que hay. Se puede distinguir en dos tipos de árboles: variables categóricas y variables continuas.
Los árboles de decisión de variables categóricas tienen una variable de orientación categórica y los árboles de decisión de la variable continua se caracterizan por tener una variable de destinación continua.

Es una forma jerárquica de estructurar los datos. Los árboles tienen raíces, hojas y ramas como los árboles de la naturaleza. Sin embargo, la asignación de estas partes es de abajo hacia arriba, en comparación con los árboles normales. Las raíces se encuentran en la parte superior del árbol y las hojas en el extremo inferior. El nodo padre es el nodo por encima de otro nodo. El nodo raíz es el nodo padre por los nodos internos que hay a continuación. Después tenemos los nodos hijos, tal como indica su nombre , son los hijos de un nodo progenitor y, seguidamente, los nodos situados debajo de un nodo padre. Por otro lado, está el nodo raíz, es el nodo superior, es decir, el origen del árbol. Después de lo cual, hay un nodo hoja, llamado también nodo externo, y se puede ver un punto muerto porque es el último nodo y no tiene nodos secundarios, por debajo. Por otra parte, el proceso de dividir un nodo en dos o más subnodos, de ahí se obtiene los nodos padres, hijos y hojas. A continuación, la poda es cuando se elimina los subnodos de un nodo de decisión. Se puede decir que es el proceso contrario a dividir. Por último, están los nodos de interior o de ramas que tienen una conexión por encima y por debajo, es decir, entre un nodo hijo y padre.

Teniendo en cuenta el problema, se utiliza los árboles de decisión como regresión debido a que hay que predecir una variable continua. Cuando los valores son continuos se discretizan antes de construir el modelo. Los registros se distribuyen de manera recursiva a partir de los valores de los atributos. El orden de situar los atributos como la raíz o el nodo interno del árbol se hace mediante un enfoque estadístico.Para medir la calidad de esta regresión se hace la suma de los residuos cuadrados. Se puede ir dividiendo con los árboles, el espacio, pero hilar muy fino con esto te puedo dar overfitting.

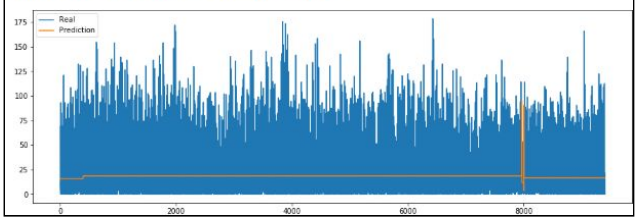
En primer lugar, se ha importado el módulo DecisionTreeRegression de la librería sklearn. A continuación se ha creado una variable tree_model que contiene el algoritmo del árbol. Luego, se ha ajustado el modelo con los datos de train y se ha hecho un predict con los datos de test, para ver el resultado se ha hecho un plot. Para hacer el modelo y que pueda tener en cuenta todos los valores, se ha hecho un reshape para tener todos los datos en las misma columna.

```
from sklearn.tree import DecisionTreeRegressor

tree_model = DecisionTreeRegressor()

tree_model.fit(x_train.reshape(-1,1), train_1.reshape(-1,1))

DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')

y_pred_tree = tree_model.predict(x_test.reshape(-1,1))

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train.reshape(-1,1), train_1.reshape(-1,1), label='Real')
ax.plot(x_test.reshape(-1,1), test_1.reshape(-1,1), label = 'Test')
ax.plot(x_test.reshape(-1,1), y_pred_tree, label='Prediction', linewidth =5)
ax.legend()

<matplotlib.legend.Legend at 0x7f9a14320150>
```



Se observa que el modelo decision tree no ajusta los datos de train en test, pero este algoritmo tiene un parámetro que se llama max depth, que controla la profundidad máxima del árbol que se creará. Básicamente, se puede describir como la longitud del camino más largo desde la raíz hasta la hoja. En este caso, se combinan diferentes clasificadores para agrupar diferentes grupos por medias. En el apartado anterior, se obtiene una línea debido a la naturaleza del árbol. Esto se debe a que los árboles de decisión son buenos para hacer una interpolación, pero no una extrapolación. Por tanto, sólo puede predecir lo que puede ver. En este caso, se ha añadido el valor de tres en max depth, para comprobar si se ajusta mejor a los datos de train y predice mejor en los datos de test.

```
tree_model = DecisionTreeRegressor(max_depth=3)

tree_model.fit(x_train.reshape(-1,1), train_1.reshape(-1,1))

DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')

y_pred_tree = tree_model.predict(x_train.reshape(-1,1))

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train.reshape(-1,1), train_1.reshape(-1,1), label='Real')
ax.plot(x_train.reshape(-1,1), y_pred_tree, label='Prediction')
ax.legend()

<matplotlib.legend.Legend at 0x7f99f7ffc950>
```



En el plot, se puede comprobar que con el parámetro max depth quiere ajustarse pero aún no lo consigue, se tendrá que continuar con la siguiente prueba de machine learning que es el Random Forest.

Como en los anteriores apartados, se ha hecho para los correspondientes contaminantes con las estaciones, para las variables meteorológicas y los sensores que van relacionados comentados anteriormente.

4.2.5 Random Forest

El Ensemble es el proceso mediante el cual múltiples modelos, como clasificadores o expertos, se generan y combinan estratégicamente para resolver un problema de inteligencia computacional en particular. El aprendizaje en conjunto se utiliza principalmente para mejorar el rendimiento (clasificación, predicción, aproximación de funciones, etc.) de un modelo o reducir la probabilidad de una selección desafortunada de uno deficiente. Otras aplicaciones del aprendizaje por conjuntos incluyen asignar confianza a la decisión tomada por el modelo, seleccionar características óptimas (o casi óptimas), fusión de datos, aprendizaje incremental, aprendizaje no estacionario y corrección de errores. Este artículo se centra en las aplicaciones del aprendizaje por conjuntos relacionadas con la clasificación; sin embargo, todas las ideas principales que se describen a continuación se pueden generalizar fácilmente a problemas de aproximación de funciones o de predicción.[14]

El Random Forest es una técnica de Bagging es unos de los tipos de ensemble , que significa agregación bootstrap, es uno de los algoritmos basados en conjuntos más tempranos, más intuitivos y quizás más simples, con un rendimiento sorprendentemente bueno (Breiman 1996). La diversidad de clasificadores en ensacado se obtiene utilizando réplicas de arranque de los datos de entrenamiento. Es decir, se extraen aleatoriamente diferentes subconjuntos de datos de entrenamiento, con reemplazo, de todo el conjunto de datos de entrenamiento. Cada subconjunto de datos de entrenamiento se usa para entrenar un clasificador diferente del mismo tipo. Luego, los clasificadores individuales se combinan tomando una mayoría simple de votos de sus decisiones. Para cualquier caso dado, la clase elegida por la mayoría de clasificadores es la decisión de conjunto. Dado que los conjuntos de datos de entrenamiento pueden superponerse sustancialmente, se pueden usar medidas adicionales para aumentar la diversidad, como usar un subconjunto de los datos de entrenamiento para entrenar a cada clasificador, o usar clasificadores relativamente débiles (como los tocones de decisión).[14]

En esta técnica, el objetivo es descorrelacionar los diferentes árboles que se ha generado, y después se reduce la varianza promediándolos con los árboles de decisión. Promediar los árboles de decisión ayuda a tener un mejor rendimiento en test, para así prevenir el overfitting. Con Random Forest se puede construir muchos árboles para tener una correlación más pequeña entre estos. Por tanto, la técnica de Random Forest sirve para ajustar un modelo mucho más preciso al hacer el promedio entre muchos árboles de decisión, reducir la varianza y así eludir el problema del overfitting.

Para empezar con el Random Forest como en los modelos anteriores, se ha importado desde la librería de sklearn ensemble. A continuación, se ha creado una variable con el algoritmo de random forest con los parámetros de n_estimators y max_depth con los valores cien y veinte,

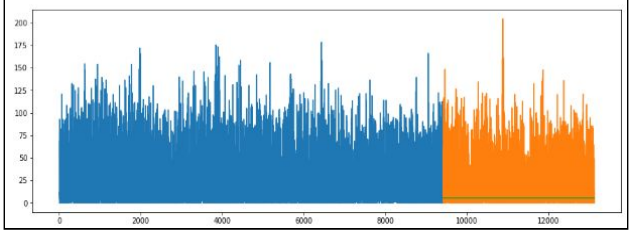
respectivamente. Después, se ha creado el modelo para poder ver cómo se ajusta y como predice el modelo. Con consiguiente para comprobar cómo ha salido el modelo se ha hecho un plot.

```
from sklearn import ensemble

rf_model = ensemble.RandomForestRegressor(n_estimators=100, max_depth=20)
rf_model.fit(x_train.reshape(-1,1), train_1.reshape(-1,1))
y_pred_rf = rf_model.predict(x_test.reshape(-1,1))
```

```
/home/marc/[/Usuarios/marcj/Descargas]/envs/masterds/lib/python3.7/site-packages/ipykernel_launcher.py:2: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
```

```
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train.reshape(-1,1), train_1.reshape(-1,1))
ax.plot(x_test.reshape(-1,1), test_1.reshape(-1,1), label='Real')
ax.plot(x_test.reshape(-1,1), y_pred_rf, label='Prediction')
```

```
[<matplotlib.lines.Line2D at 0x7f9a0c231310>]
```



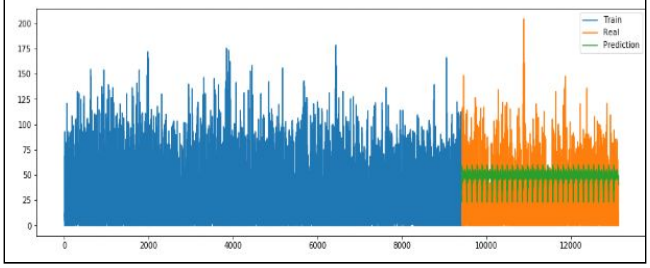
Como se puede comprobar el modelo con random forest tampoco ajusta de la manera correcta. Para poder hacer que se ajuste, se ha creado dos features una para train y otra para test, llamadas x_train_month y x_test_month. Estas features permiten ajustar los datos en meses para que así, los datos pueden ajustarse mejor. Para comprobar si con estas features ajusta mejor, se ha creado de nuevo un modelo pero ajustando con la nueva feature de train y prediciendo con la nueva feature de test y, a continuación, se ha hecho un plot para comprobarlo.

```
x_train_month = np.hstack((x_train.reshape(-1,1), np.array(list(range(147)) * 6
x_test_month = np.hstack((x_test.reshape(-1,1), np.array(list(range(120)) * 31)
```

```
rf_model = ensemble.RandomForestRegressor(n_estimators=100, max_depth=20)
rf_model.fit(x_train_month, train_1.reshape(-1,1))
y_pred_rf = rf_model.predict(x_test_month)

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train.reshape(-1,1), train_1.reshape(-1,1), label='Train')
ax.plot(x_test.reshape(-1,1), test_1.reshape(-1,1), label='Real')
ax.plot(x_test.reshape(-1,1), y_pred_rf, label='Prediction')
ax.legend()
```

```
/home/marc/[/Usuarios/marcj/Descargas]/envs/masterds/lib/python3.7/site-packages/ipykernel_launcher.py:2: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
```

```
<matplotlib.legend.Legend at 0x7f9a14306fd0>
```



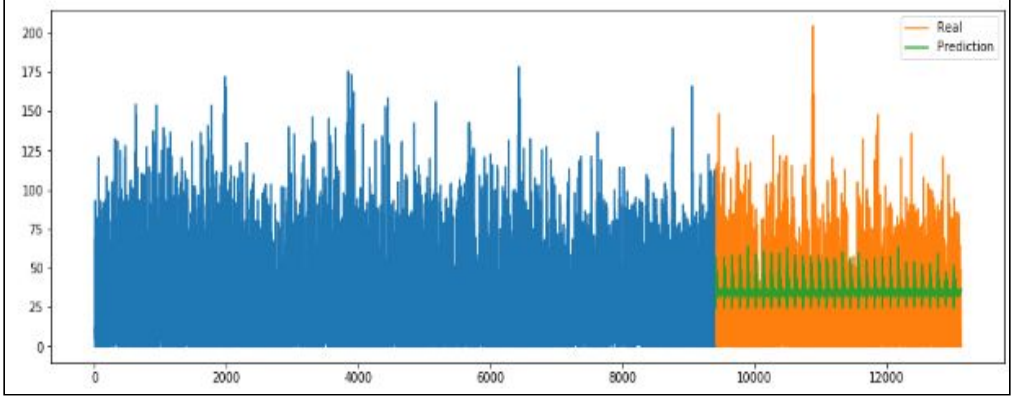
Como se puede ver en el plot, gracias a las features que se han hecho anteriormente, se comprueba que se ajusta mejor que antes. Pero se debe ajustar de una manera más correcta por lo que se han creado dos nuevas features de train y test, x_train_month_year y x_test_month_year. Estas features se han ajustado para año y mes, en teoría debería ajustar mejor que las anteriores features. A continuación, se ha creado el modelo ajustando con los mismos estimadores y con el mismo valor de max depth, pero ajustando el modelo con la nueva feature de train y prediciendo con la nueva feature de test y para comprobar el resultado, se ha hecho un plot.

```
x_train_month_year = np.hstack((x_train_month, np.repeat(list(range(147)), 64).
x_test_month_year = np.hstack((x_test_month, np.repeat(list(range(1,121)), 31).
```

```
rf_model = ensemble.RandomForestRegressor(n_estimators=100, max_depth=20)
rf_model.fit(x_train_month_year, train_1.reshape(-1,1))
y_pred_rf = rf_model.predict(x_test_month_year)

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train.reshape(-1,1), train_1.reshape(-1,1))
ax.plot(x_test.reshape(-1,1), test_1.reshape(-1,1), label='Real')
ax.plot(x_test.reshape(-1,1), y_pred_rf, label='Prediction')
ax.legend()
```

```
/home/marc/[/Usuarios/marcj/Descargas]/envs/masterds/lib/python3.7/site-package
s/ipykernel_launcher.py:2: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples,), for
example using ravel().
```

```
<matplotlib.legend.Legend at 0x7f99f6b01a50>
```



Se puede ver que ajusta mejor los datos pero no es suficiente como para tener una buena predicción, por lo cual no es el modelo que se escogerá.

Como se ha dicho anteriormente, se ha realizado este técnica con todos los contaminantes y sus respectivas estaciones como sus respectivas variables meteorológicas como los sensores relacionados.

4.2.6 XGBoost

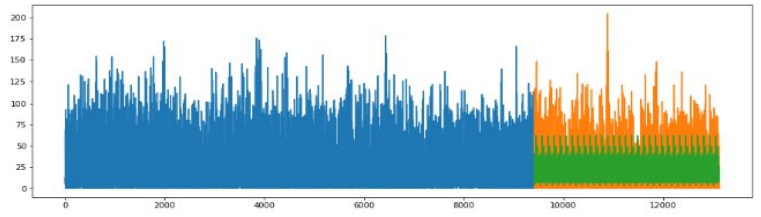
De manera similar al bagging, el boosting también crea un conjunto de clasificadores volviendo a muestrear los datos, que luego se combinan mediante votación por mayoría. Sin embargo, en el impulso, el remuestreo está estratégicamente orientado a proporcionar los datos de entrenamiento más informativos para cada clasificador consecutivo. En esencia, cada iteración de refuerzo crea tres clasificadores débiles: el primer clasificador C1 se entrena con un subconjunto aleatorio de los datos de entrenamiento disponibles. El subconjunto de datos de entrenamiento para el segundo clasificador C2 se elige como el subconjunto más informativo, dado C1. Específicamente, C2 se entrena con datos de entrenamiento, de los cuales solo la mitad está correctamente clasificada por C1, y la otra mitad está mal clasificada. El tercer clasificador C3 se entrena con instancias en las que C1 y C2 no están de acuerdo. Los tres clasificadores se combinan mediante un voto mayoritario a tres bandas.[14]

Schapire mostró que el error de este algoritmo tiene un límite superior: si el algoritmo A usado para crear los clasificadores C1, C2, C3 tiene un error de ϵ (calculado en S), entonces el error del conjunto está limitado por f (ϵ) = 3ϵ2−2ϵ3. Tenga en cuenta que f (ϵ) ≤ϵ para ϵ <1/2. Es decir, siempre que el algoritmo A original pueda funcionar al menos mejor que la adivinación aleatoria, entonces el conjunto de refuerzo que combina tres clasificadores generados por A en las tres distribuciones de S descritas anteriormente siempre superará a A. Además, el error de conjunto es un error de entrenamiento. Por lo tanto, se genera un clasificador más fuerte a partir de tres clasificadores más débiles. A continuación, se puede crear un clasificador fuerte en el sentido estricto de aprendizaje de PAC mediante aplicaciones recursivas de refuerzo. Una limitación particular del impulso es que se aplica sólo a problemas de clasificación binaria.[14]

Para empezar con el algoritmo de XGBoost, se ha importado desde la librería xgboost: los módulos de XGBRegressor, plot_tree y plot-importance. A continuación, se ha hecho una variable que se ha incorporado al algoritmo. Luego, se ha ajustado el modelo con los parámetros de train y la última feature de train creada y los parámetros del modelo son n_estimators y max_depth con los valores cien y cuatro, respectivamente . Después de hacer el predict con la última feature de test, se ha hecho un plot para comprobar si se ajusta como debería los datos de test. Se puede ver que los datos se ajustan mucho mejor pero este tipo de algoritmo tiene otro parámetro learning rate, que controla la ponderación de los nuevos árboles añadidos. Se ha creado un nuevo modelo con los parámetros n_estimators, max_depth y learning_rate con los valores cien, tres y uno, respectivamente. Para poder ver si ajusta mejor que el modelo anterior, se ha hecho un plot para comprobarlo y como se puede ver en la siguiente figura, ajusta mejor pero aún se puede ajustar un poco mejor nuestra predicción.

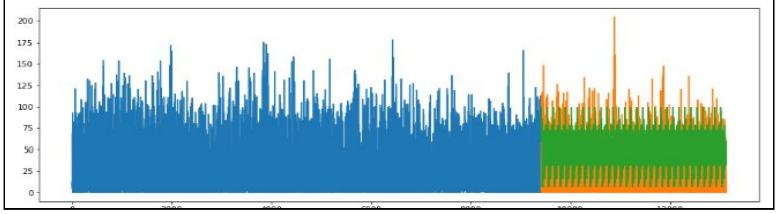
```
from xgboost import XGBRegressor, plot_tree, plot_importance

xgb_model = XGBRegressor(n_estimators=100, max_depth=4)
xgb_model.fit(x_train_month_year, train_1.reshape(-1,1))
y_pred_xgb = xgb_model.predict(x_test_month_year)

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train.reshape(-1,1), train_1.reshape(-1,1))
ax.plot(x_test.reshape(-1,1), test_1.reshape(-1,1), label='Real')
ax.plot(x_test.reshape(-1,1), y_pred_xgb, label='Prediction')
```

```
[<matplotlib.lines.Line2D at 0x7f99e0727890>]
```




```
xgb_model = XGBRegressor(n_estimators=100, max_depth=3, learning_rate=1)
xgb_model.fit(x_train_month_year, train_1.reshape(-1,1))
y_pred_xgb = xgb_model.predict(x_test_month_year)


fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train.reshape(-1,1), train_1.reshape(-1,1))
ax.plot(x_test.reshape(-1,1), test_1.reshape(-1,1), label='Real')
ax.plot(x_test.reshape(-1,1), y_pred_xgb, label='Prediction')
```

```
[<matplotlib.lines.Line2D at 0x7f99e00b7b10>]
```



Para poder ajustar mejor aún el modelo, se han creado dos features de train y test, x_train_month_2ylag y x_test_month_2ylag. Estas features son dos arrays que tienen dos años de lag para que se ajuste mejor a los datos.

```
x_train_month_year_2ylag = np.hstack(((train_1.reshape(-1,1)[24:] / train_1.reshape(-1,1)[:-24])[:-24],
                                      (train_1.reshape(-1,1)[24:] - train_1.reshape(-1,1)[:-24])[:-24],
                                      x_train_month_year[48:,:]))
```

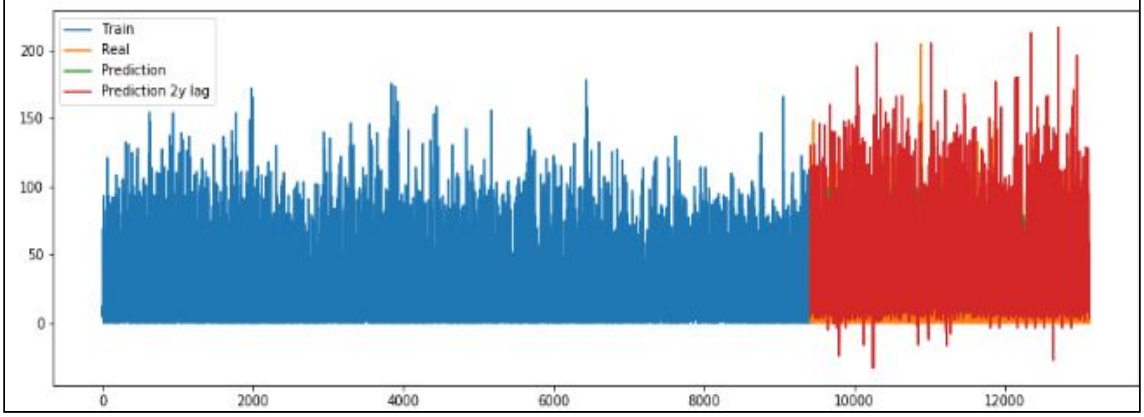
```
/home/marc/[/Usuarios/marcj/Descargas]/envs/masterds/lib/python3.7/site-packages/ipykernel_launcher.p
y:1: RuntimeWarning: divide by zero encountered in true_divide
  """Entry point for launching an IPython kernel.
/home/marc/[/Usuarios/marcj/Descargas]/envs/masterds/lib/python3.7/site-packages/ipykernel_launcher.p
y:1: RuntimeWarning: invalid value encountered in true_divide
  """Entry point for launching an IPython kernel.
```

```
x_test_month_year_2ylag = np.hstack(((train_1[100:].reshape(-1,1) / train_1[:-100].reshape(-1,1))[:372
                                     (train_1[100:].reshape(-1,1) - train_1[:-100].reshape(-1,1))[:372
                                     x_test_month_year))
```

A continuación, se ha hecho un nuevo modelo ajustando con la nueva feature de train y se ha hecho un predict con la feature de test. Para comprobar el resultado, se ha hecho un plot y se ha mirado el score del modelo, se ha dado este resultado.

```
xgb_model = XGBRegressor(n_estimators=300, max_depth=3, learning_rate= 1.5)
xgb_model.fit(x_train_month_year_2ylag, train_1.reshape(-1,1)[48:])
y_pred_xgb_2ylag = xgb_model.predict(x_test_month_year_2ylag)

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train.reshape(-1,1), train_1.reshape(-1,1), label='Train')
ax.plot(x_test.reshape(-1,1), test_1.reshape(-1,1), label='Real')
ax.plot(x_test.reshape(-1,1), y_pred_xgb, label='Prediction')
ax.plot(x_test.reshape(-1,1), y_pred_xgb_2ylag, label='Prediction 2y lag')
ax.legend()
```

```
<matplotlib.legend.Legend at 0x7f99e0081550>
```




```
print(xgb_model.score(x_train_month_year_2ylag, train_1.reshape(-1,1)[48:]))
```

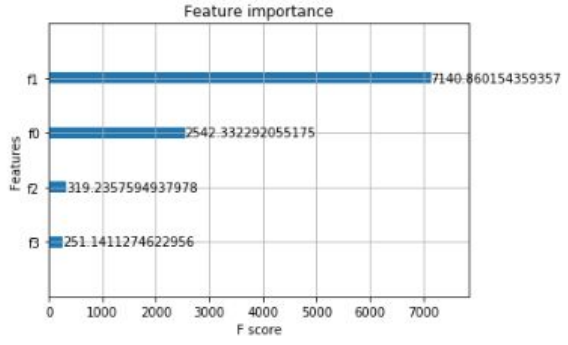
```
0.9967662905692417
```

Como se puede ver en este modelo hay una predicción muy notable de los datos, gracias a las nuevas features, debido a que el algoritmo de xgboost aprende muchísimo mejor con unas features con un retraso de dos años. Para ver la importancia de estas features, se ha creado un plot importance para ver esta clasificación.

En este punto, hay un modelo con una score alta pero se ha querido intentar hilar más fino, por lo que se ha creado una nueva variable que elimina la tendencia, restando los valores de train con los valores predichos por el modelo de regresión lineal.

```
plt.figure()
plot_importance(xgb_model, importance_type='gain')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f99d0061950>

<Figure size 432x288 with 0 Axes>
```



```
model = LinearRegression()
model.fit(x_train.reshape(-1, 1), train_1.reshape(-1,1))
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
train_log_trend = (train_1.reshape(-1,1) - model.predict(x_train.reshape(-1, 1))).squeeze()
```

A continuación, se ha creado otro modelo con xgboost y los parámetros n_estimators, max_depth y learning_rate con los valores 500, 3 y 1.2, respectivamente. Se ha procedido a pasar los valores del dataframe train_log_trend a values, para poder ajustarlo al modelo con el array de x_train_month_2ylag. Luego, se ha hecho un predict con el array de x_test_month_2ylag. El resultado de este modelo ha sido un score un poco más alto que el anterior.

```
xgb_model.score(x_train_month_year_2ylag, train_log_trend.reshape(-1,1)[48:])
```

```
0.9986389424336982
```

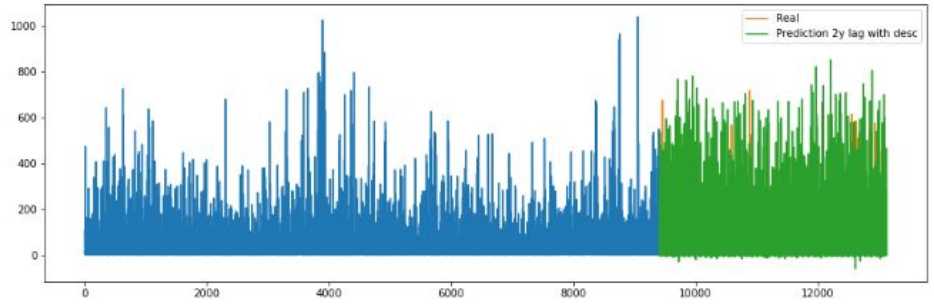
Después, se ha creado una nueva y, que se ha incluido otra vez la tendencia con la y que se ha predicho anteriormente.

```
y_pred_xgb_2ylag_desc = (y_pred_xgb_2ylag_wotrend + model.predict(x_test.reshape(-1, 1)).squeeze())
```

Luego, se ha hecho un plot y se ha mirado el score de x_test_month_year_2ylag con y_pred_2ylag_desc que es la última y que se ha predicho.

```
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(x_train.reshape(-1,1), train_1.reshape(-1,1))
ax.plot(x_test.reshape(-1,1), test_1.reshape(-1,1), label='Real')

#ax.plot(x_test, forecast_daily, linewidth='3', linestyle='--', markersize= 0.2, marker='o')
#ax.plot(x_test.reshape(-1,1), y_pred_xgb, label='Prediction seasonality')
#ax.plot(x_test.reshape(-1,1), y_pred_xgb_2ylag, label='Prediction 2y lag')
#ax.plot(x_test.reshape(-1,1), y_pred_xgb_2ylag_wotrend, label='wotrend')
ax.plot(x_test.reshape(-1,1), y_pred_xgb_2ylag_desc, label='Prediction 2y lag with desc')
ax.legend()
```

```
<matplotlib.legend.Legend at 0x7f96f03e35d0>
```




```
xgb_model.score(x_test_month_year_2ylag, y_pred_xgb_2ylag_desc)
```

```
0.9582201183467738
```

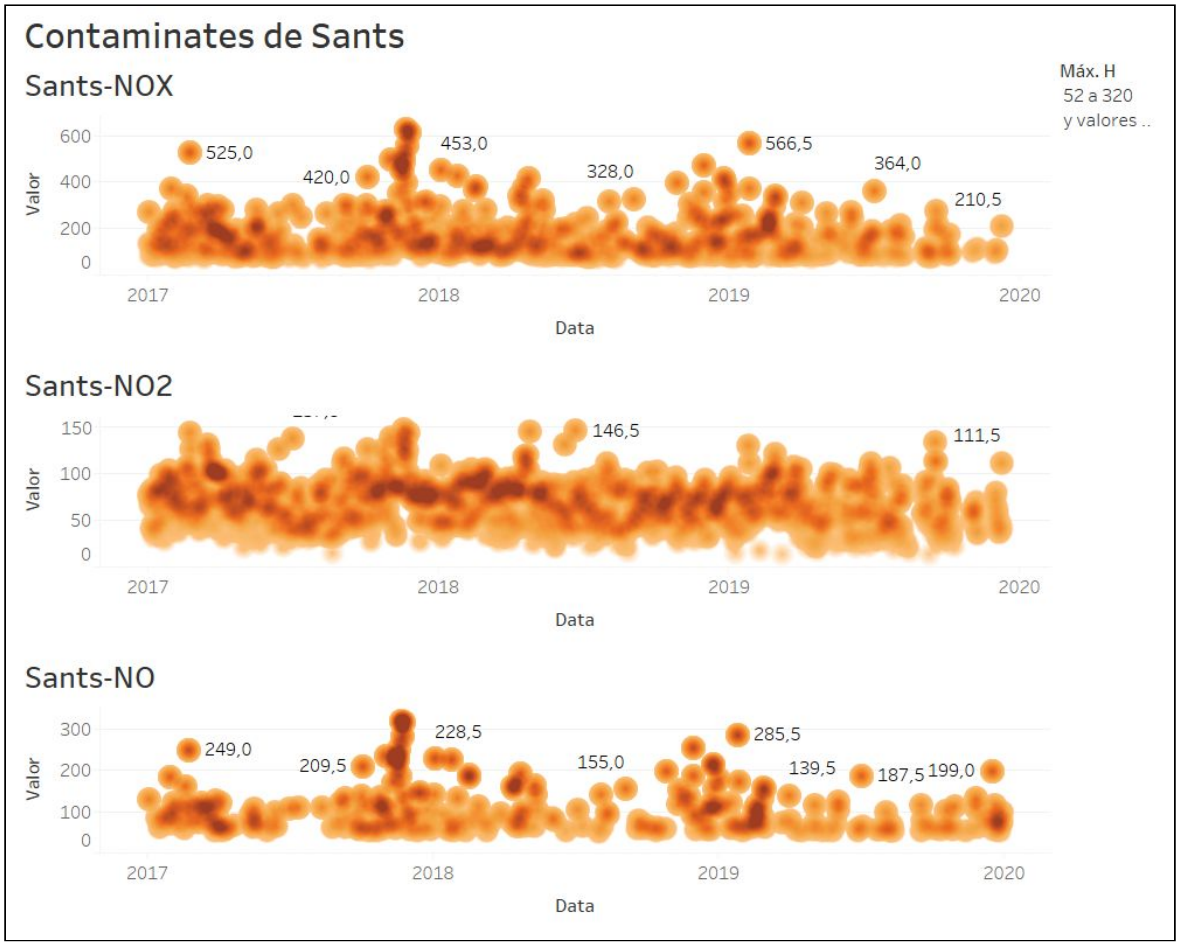
Se puede observar en el score, que ha disminuido desde el último que se ha hecho. Teniendo en cuenta que es la parte de test no la de train. Ha sucedido con todos los modelos que se han hecho. Esto puede ser debido a que la tendencia no estan pronunciada o que se ha perdido valor cuando se ha quitado la tendencia.

4.4 Discusión

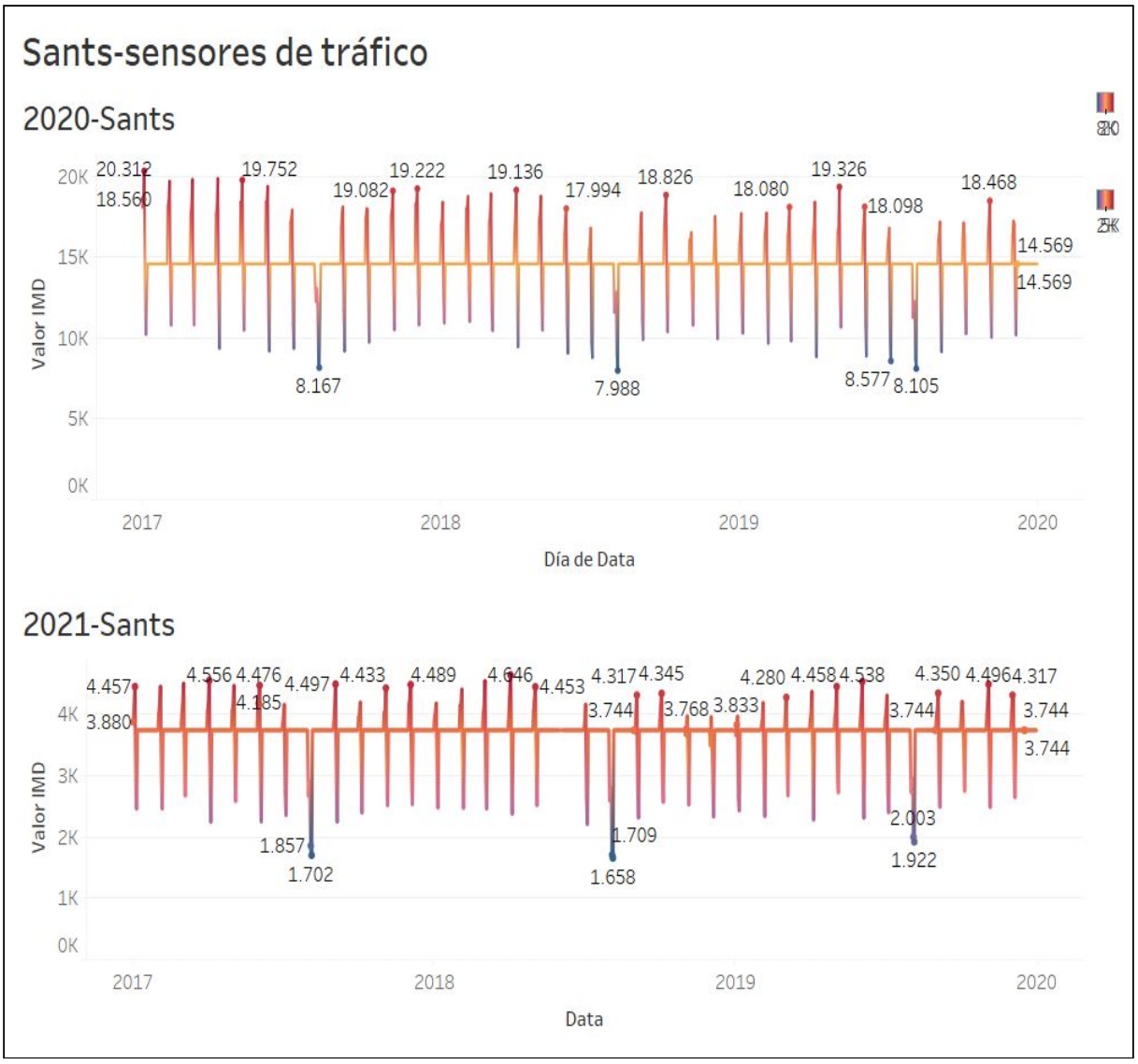
En este punto del proyecto se han visto todos los modelos, el modelo Arima se ha podido comprobar que es un buen baseline. El modelo de regresión lineal no se ajustaba correctamente y se ha considerado hacer un modelo de regresión con más peso pero tampoco funcionó. Para poder trabajar con un poco más de libertad se ha continuado con la técnica de regresión polinómica. En este punto, se han añadido los parámetros meteorológicos y de tráfico relacionados con cada estación y cada contaminante. En este punto se ha visto que la regresión polinómica, se ha empezado ajustar pero no de una manera coherente. A continuación, se ha empezado con los árboles de decisiones, con el parámetro max_depth ha ajustado mejor de lo que se había visto anteriormente. Con el Random Forest y la features se ha visto un cambio notable, pero no ha sido suficiente. Y por último, el XGBoost se ha podido observar que, gracias a las features y su algoritmo es el mejor que predice ,comprobado con los plots y los scores. Para comprobar que influencia tienen las variables meteorológicas y los sensores de tráfico en el repositorio de github, hay un notebook donde se ha hecho una predicción con todos los algoritmos probados pero con sólo datos de contaminantes. Los scores de los modelos son muchísimo inferior a los modelos con las variables meteorológicas y los sensores de tráfico. Por tanto, se ha podido observar la relación entre las variables meteorológicas y los sensores con los datos de los contaminantes. Para siguientes proyectos, se puede hacer esta predicción con otro tipo de algoritmos como el SVM o las redes neuronales, para poder profundizar más en las series temporales.

Capítulo 5

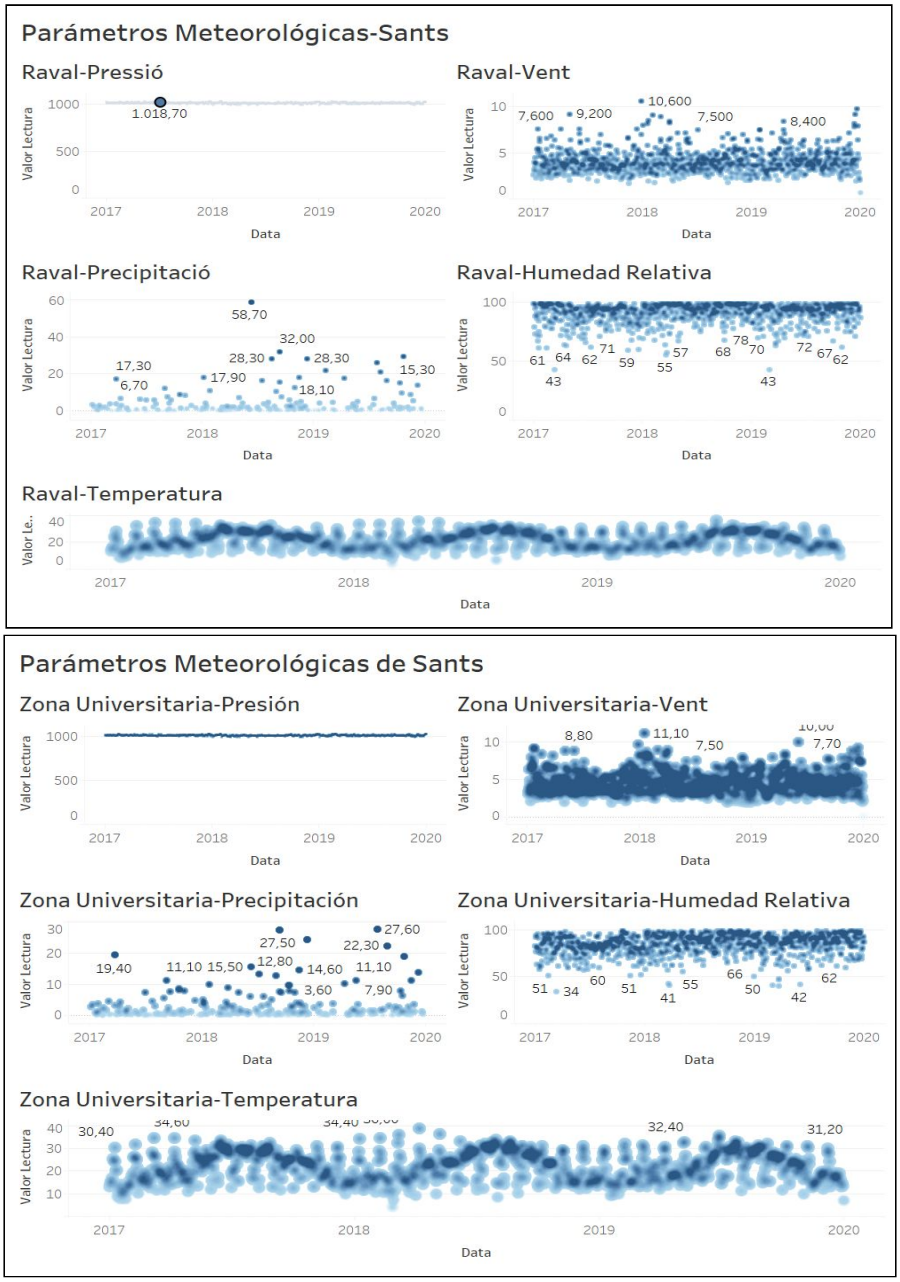
Visualizaciones



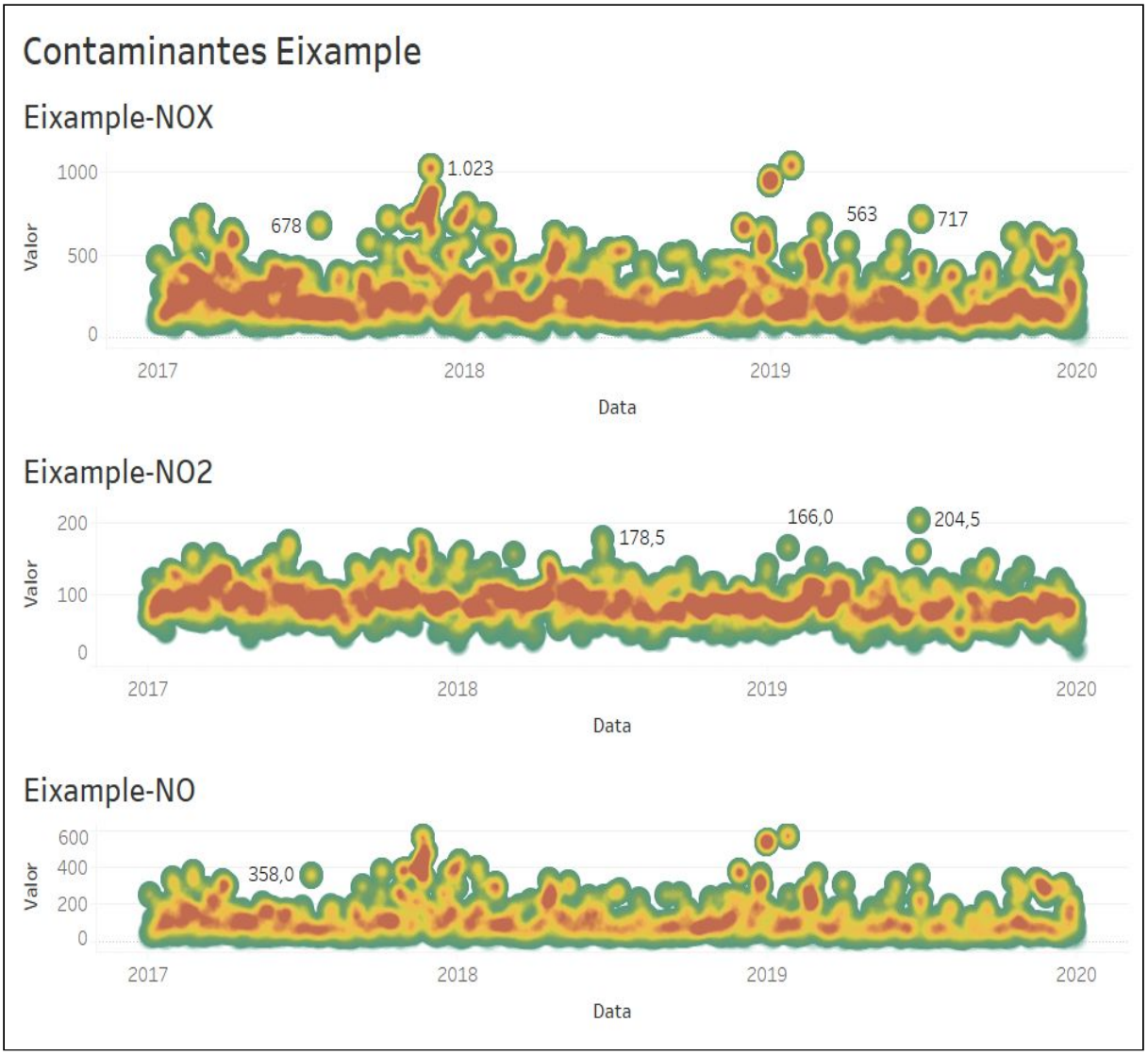
Para poder ver cómo actúan los datos que se han seleccionado, se han hecho unos dashboards con tableau para mostrarlos. El primer dashboard son los contaminantes de Sants, se observa que cuando empieza el frío después del verano aumenta la contaminación. En invierno el aire frío que está a nivel del suelo, se caliente cuando sale el sol y la capa de aire superior que está a otra temperatura impide que el aire frío contaminado de abajo no pueda subir. Por eso el invierno es la estación del año en la que hay más contaminación. Por tanto, estos datos están concordes con la literatura que hay sobre la contaminación. Otra cosa que se puede observar en el dashboard, es que el contaminante NOX tiene los valores más elevados y el contaminante NO2, los valores más pequeños. Por lo que en esta zona hay un factor de riesgo con el contaminante NOX.



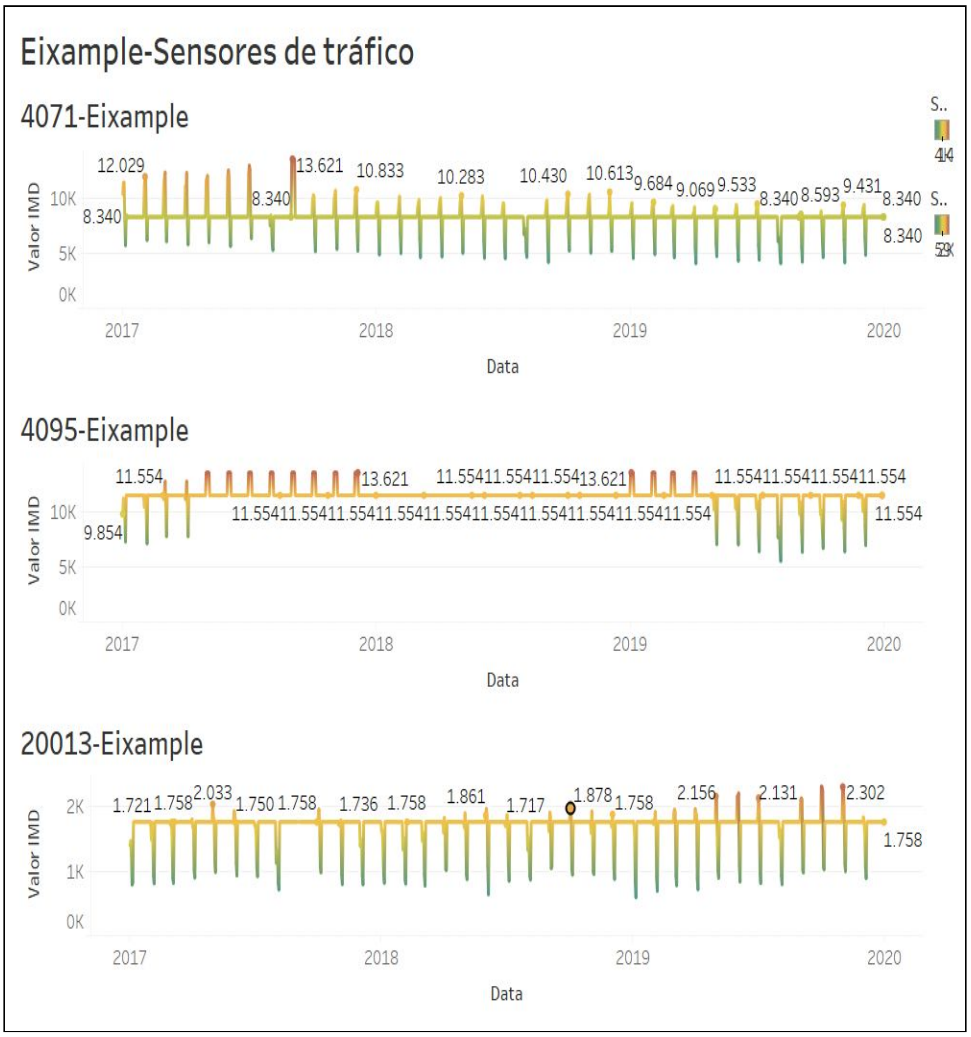
En el siguiente dashboard, están las visualizaciones de los sensores que se han asignado a Sants. Se puede comprobar, que hay una sintonía con los datos de los contaminantes, que en verano son los más bajos y con consiguiente después del verano suben. Por lo que hay una correlación de estos parámetros como se ha dicho. Por tanto, estos datos confirman la literatura sobre la relación entre contaminación y tráfico. El sensor 2020 tiene valores mucho más altos, esto indica que es una zona de mayor influencia de cotxes que en la zona del sensor 2021, pero tanto uno como el otro siguen la tendencia de la que se ha hablado. Otra tendencia que podemos observar, sobretodo en el gráfico del sensor 2020, es que del año 2017 al año 2019 ha disminuido el tráfico.



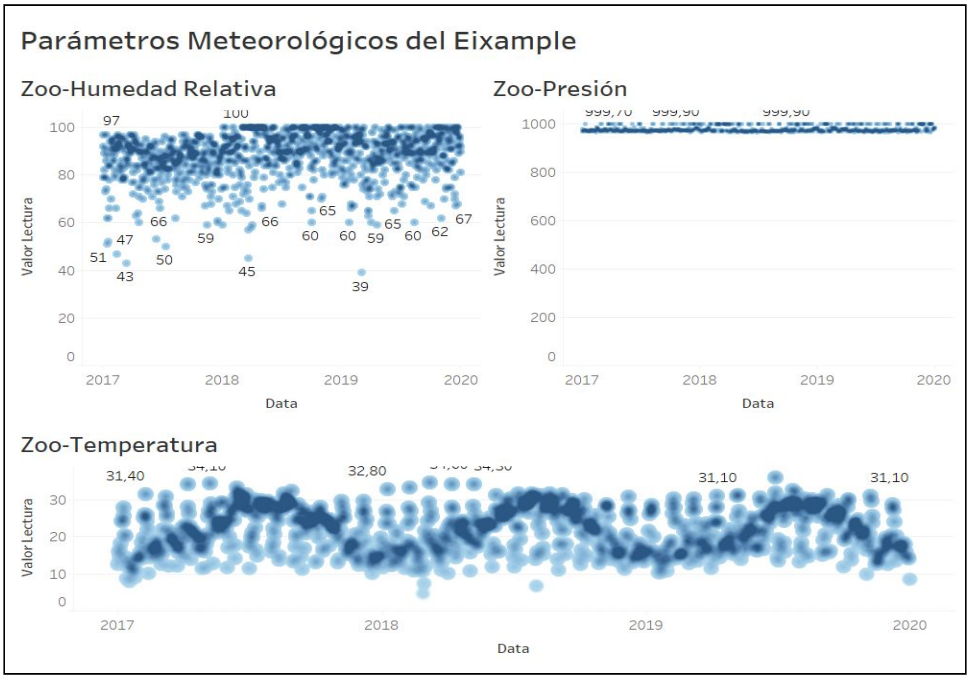
Este es el último dashboard relacionado con la zona de Sants, se compone de los parámetros meteorológicos de dos estaciones que son: la estación del Raval y Sants, donde hay dos parámetros bastante equilibrados que son la humedad relativa y la presión. La humedad tiene valores estables en el 80% y la presión tiene un valor estable de 1000 Pa. Pero hay dos variables que son el viento y la temperatura que siguen una tendencia. El viento sigue la misma tendencia que los contaminantes y los sensores, pero la temperatura sigue una tendencia contraria a esta. Esto es debido como se ha dicho en el dashboard de los contaminantes, a que en invierno al tener la capa de aire frío más cerca del suelo y tener una capa de aire caliente cuando amanece, el aire frío contaminado no puede subir esto se llama inversión térmica.

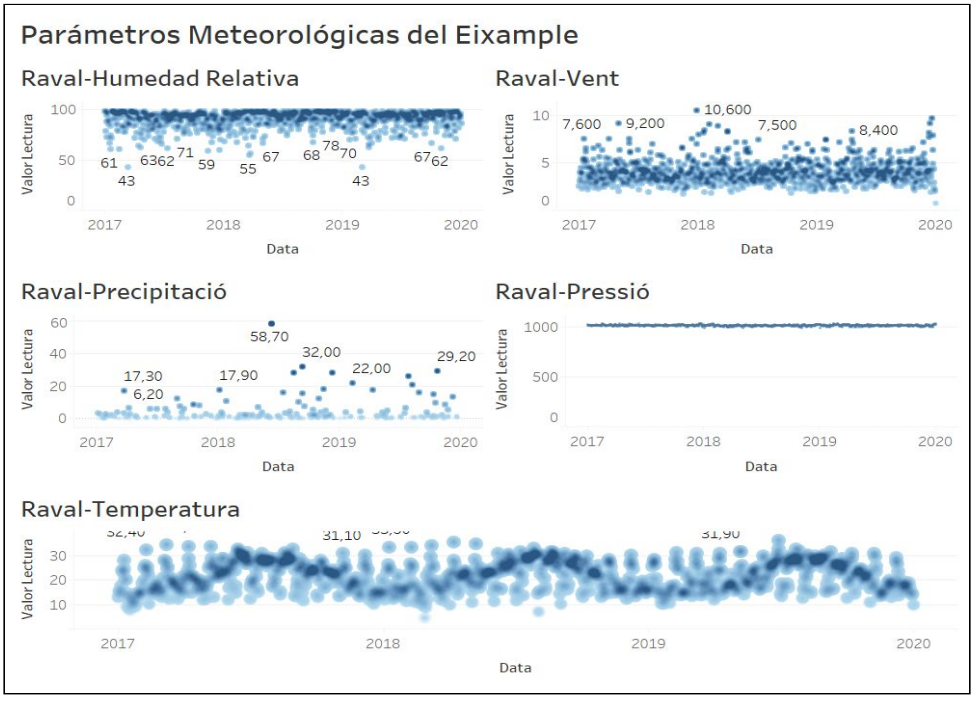


En este dashboard, se puede observar los contaminantes del Eixample con la misma tendencia de los contaminantes de Sants. Como en el caso anterior, esta tendencia es debido a la inversión térmica, por tanto está acorde con la literatura que habla sobre este tema. En este caso, los niveles más elevados de contaminación lo tiene el contaminante NOX, con valores más grandes de 1000. De momento, podemos observar que el contaminante más abundante de esta familia en Barcelona es el NOX. El contaminante NO2 tiene los valores más pequeños en comparación a los otros dos contaminantes, al igual que los contaminantes de Sants. Podemos comprobar que durante los años se mantiene bastante constante, esto quiere decir, que los contaminantes siguen estables y no están disminuyendo.

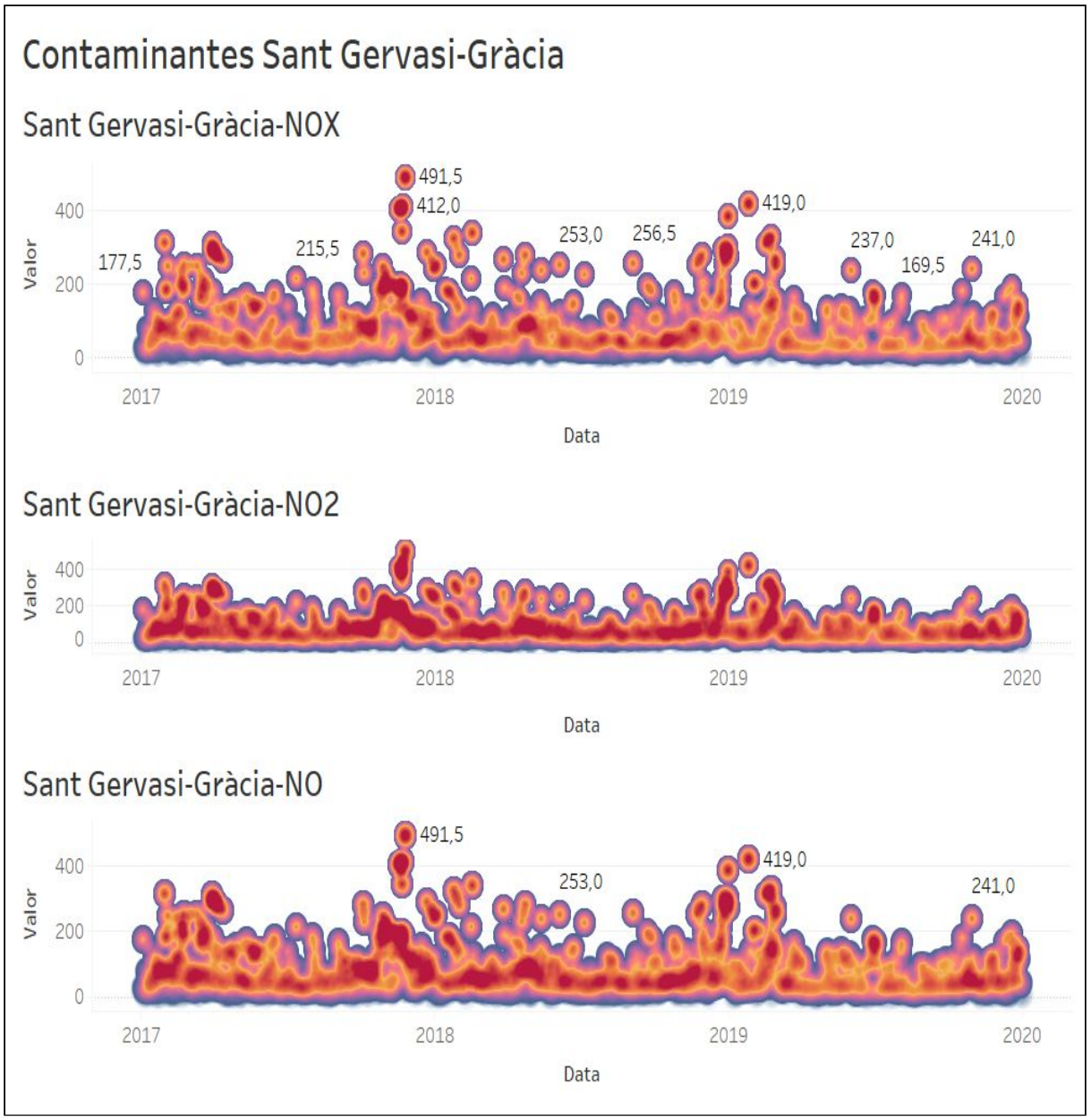


En el siguiente dashboard, se pueden observar las visualizaciones de los sensores asignados a la estación del Eixample. En este caso, los datos tienen distintos valores a los anteriores gráficos de los sensores de tráfico de Sants. En este podemos observar que no hay ninguna tendencia entre los tres, esto puede ser debido a que esta zona pueda tener algún tipo de obstáculo para tener una buena medición. La zona más transitada es la zona del sensor 4071 y la zona menos transitada es la del sensor 20013. Como se puede ver no hay ninguna tendencia clara en el sensor 4071, se puede ver que hay un repunte en septiembre pero luego los valores se mantienen estables. El sensor 4095 desde mediados del 2018 hasta mediados de 2019 son valores elevados, pero en los otros períodos, son más bajos. Y por último, el sensor 20013 tiene una tendencia muy estable sin ningún tipo de pico ni ningún bajón en su tendencia.

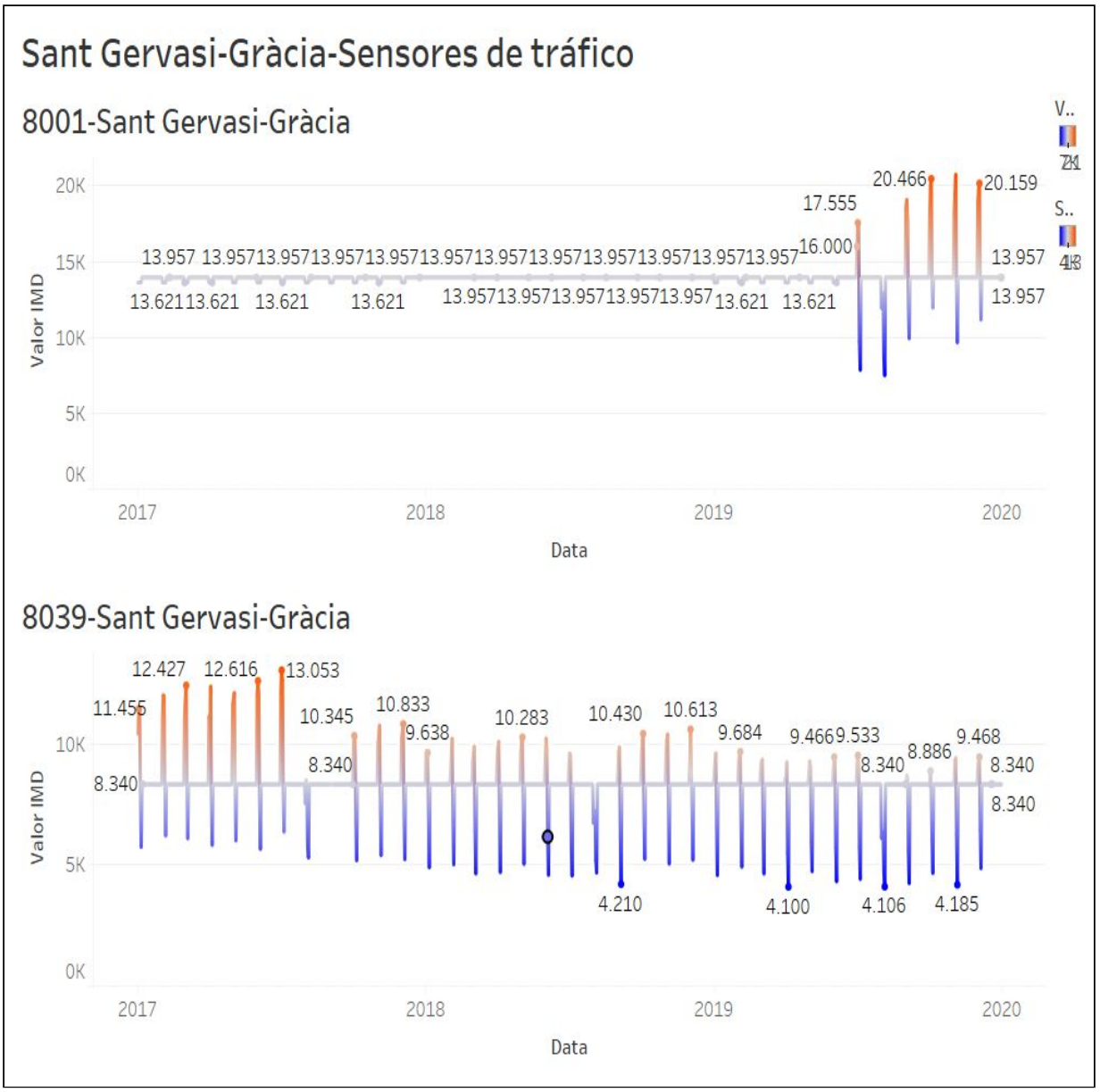




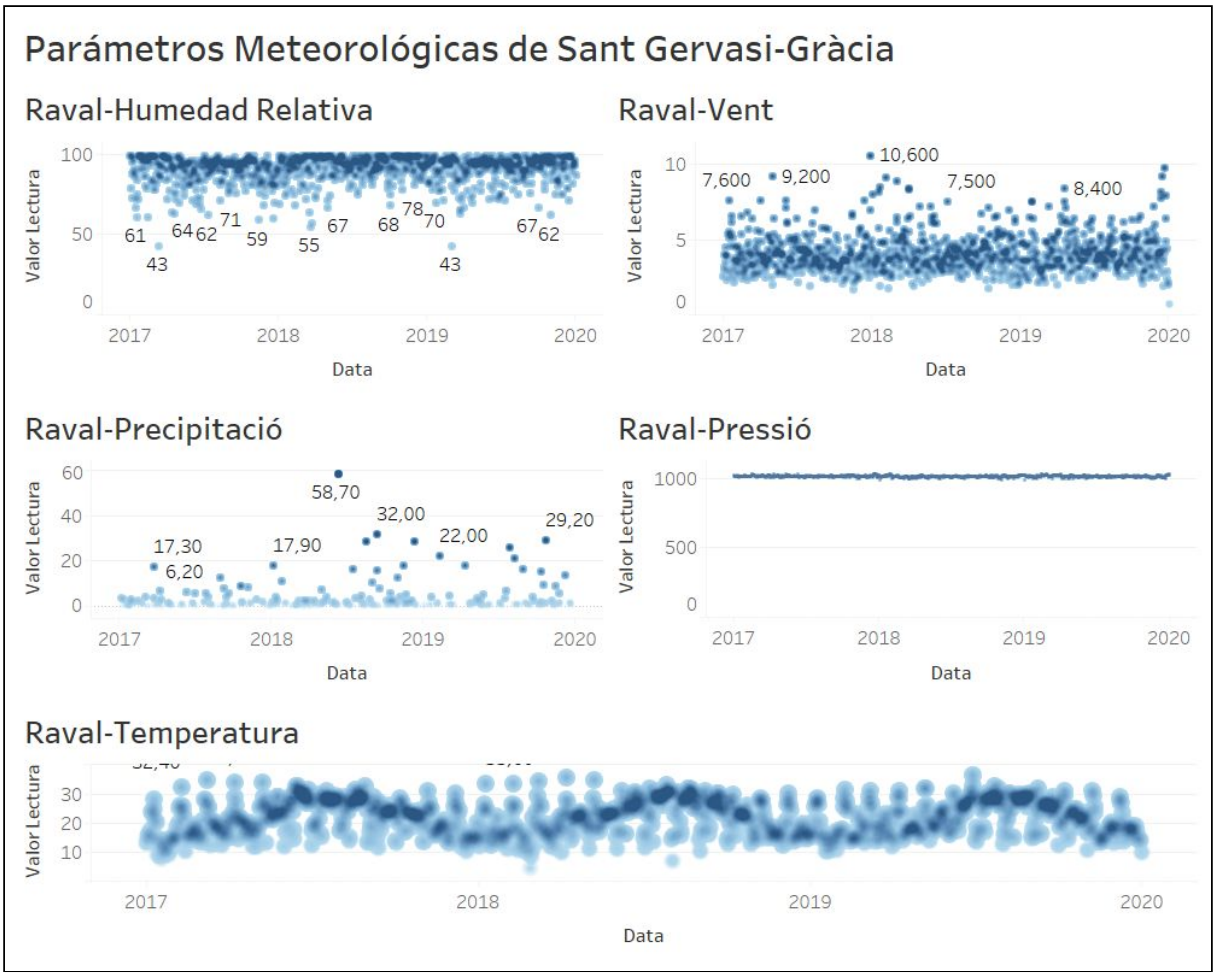
Este es el último dashboard relacionado con la estación del Eixample, donde se pueden ver los parámetros meteorológicos. En este caso, también hay dos estaciones, Raval y la ubicada en el Zoo. Como en el dashboard anterior que tiene que ver con estos parámetros tienen la misma tendencia. En el caso de estos parámetros tiene mucha importancia la temperatura, debido a que hay distintas capas con temperaturas diferentes, cosa que hace que se concentre la contaminación.



En este dashboard, se observa los contaminantes de la estación de Sant Gervasi-Gràcia. En este caso, podemos ver la misma tendencia que anteriormente. Pero en este dashboard hay cosas distintas, los parámetros tienen valores muy similares, esto puede ser por la zona en que se sitúa esta estación de medición, que tenga algunas condiciones que provoquen este tipo de datos. Por lo que se tendrá en cuenta una vista general de estos contaminantes, valores entre 0 y 500, con picos muy cerca de 500 y con la tendencia a la inversión térmica. También se puede observar que la intensidad de los contaminantes es mayor, dado que está en un intervalo de valores más pequeños, pero como se ha dicho anteriormente sigue la misma tendencia que los otros contaminantes.



Este es el último dashboard de sensores de tráfico que se comenta y se puede observar que son los datos que tienen una tendencia más irregular, puede ser debido a que se deban mejorar aún estos sensores o sus condiciones pueden ser mejorables. El primero de ellos, 8001, tiene una tendencia estable en los 13.900 pero a mediados de 2019 aumenta en invierno. En cambio, el otro sensor, 8039, tiene la tendencia que se ha comentado anteriormente, pero con valores más pequeños que el otro sensor y con una visible disminución del tráfico a lo largo de los años. El pico más alto en este caso, lo tiene en 13.053 y el más bajo en 4.100, pero este sensor nos enseña que en esta zona se ha conseguido la disminución del tráfico, no como en algunas otras zonas.



Este es el último dashboard con respecto a la estación de Sant Gervasi-Gràcia. Como en los anteriores comentarios, siguen la misma tendencia. En este caso, sólo está compuesto por una estación, el Raval. Esta estación se ha comentado anteriormente, tiene la misma tendencia en los parámetros que las otras estaciones, pero sobretodo el parámetro de la temperatura como se puede observar es el más importante, debido a la inversión térmica.

La conclusión de estas visualizaciones es clara, estos datos muestran la tendencia que hay sobre la contaminación y los distintos parámetros seleccionados. La contaminación tiene una tendencia aumentar después del invierno, debido a la inversión térmica. Los sensores se ha comprobado que son los datos más irregulares, se debería profundizar, pero seguramente sea por las condiciones de los sensores. Los parámetros meteorológicos que más influencia tienen sobre la contaminación son: el viento, pero sobretodo la temperatura debido a la inversión térmica, que contribuye a la concentración de la contaminación. Por último, con estas relaciones se ha podido comprobar que los parámetros escogidos para hacer el modelo han contribuido a mejorarlo.

Capítulo 6

Conclusiones

La baja calidad del aire es un factor que puede deteriorar la esperanza de vida, debido a que la contaminación intensifica el riesgo de tener enfermedades. A su vez, la contaminación destruye ecosistemas y el medioambiente. Existen estaciones de medición de contaminación para medir los niveles de contaminación, pero tienen un coste muy elevado, por este motivo están limitadas. Tienen otra limitación, el tiempo de validación de los datos, si estos fueran en tiempo real sería una herramienta muy útil para poder controlar la contaminación.

En este proyecto se han utilizado datos de Barcelona, es la ciudad de estudio, con el objetivo de obtener información que sea de utilidad, para predecir la contaminación para que esta información pudiera ser obtenida, antes que los datos reales observados. Para hacerlo, se han seleccionado distintos datos y parámetros, con un estudio previo para ver su relación con la contaminación y utilizarlos en este proyecto. Se han utilizado diferentes parámetros meteorológicas y datos de los sensores de la intensidad de tráfico de Barcelona, para predecir el nivel de contaminación de los óxidos de nitrógeno (NOX, NO2, NO).

Después de seleccionar los datos y tener creados los datasets, se han comparado técnicas de machine learning para predecir los niveles de los contaminantes, para tres estaciones de medición de contaminación de Barcelona. Luego de comparar las distintas técnicas, se ha podido ver el resultado, donde las mejores predicciones son a partir de la técnica del XGBoost y de las features que se han creado. Por tanto, la técnica del machine learning del Boosting, es la mejor en este caso, pero para futuros proyectos se pueden incorporar nuevas técnicas como el SVM y las redes neuronales, que pueden ajustar mejor las predicciones como también añadir nuevos parámetros, como la dirección del viento que son más difíciles de ajustar.

Para comprobar que los parámetros que se han seleccionado tiene relación, se han hecho los distintos modelos sin estos parámetros y se ha comprobado que la predicción es más aproximada con los parámetros que se han seleccionado.

Por lo tanto la conclusión final del proyecto, es que la mejor técnica de las probadas, (ARIMA, regresión lineal, regresión polinómica, decision tree, random forest y XGBoost) y con las features pertinentes, és XGBoost. Es la técnica que mejor ajusta los datos obtenidos de las distintas fuentes, esto es debido al algoritmo que usa el boosting, que combina los resultados de varios clasificadores débiles para hacerlo robusto. Y que los modelos con parámetros ajustan mucho mejor que los modelos sin parámetros. Como se ha podido ver en las visualizaciones, hay una gran correlación con algunos parámetros, por tanto, se cumple las relaciones de los parámetros que se han seleccionado.

Capítulo 7

Bibliografia

[1] https://www.who.int/topics/air_pollution/es/

[2] https://concepto.de/contaminacion-atmosferica/

[3] https://www.who.int/phe/health_topics/outdoorair/databases/health_impacts/en/

[4]https://ajuntament.barcelona.cat/qualitataire/es/qualitat-de-laire/com-es-lluita-contra-la-contaminacio/la-red-de-vigilancia-y-prevision-de-la

[5] José Hernández Orallo, Maria Jóse Ramírez Quintana, and César Ferri Ramírez. Introducción a la Minería de Datos. Pearson Prentice Hall, 2004.

[6] https://ajuntament.barcelona.cat/qualitataire/es/la-contaminacio/contaminacion-diaria-y-episodios

[7] Jeanne Mager Stellman et al. Enciclopedia de salud y seguridad en el trabajo. Ministerio de Trabajo y Asuntos Sociales, Subdirección General de Publicaciones, 1999

[8] https://blog.oxfamintermon.org/contaminacion-de-la-atmosfera-causas-y-soluciones/

[9] https://www.eea.europa.eu/es/themes/air/intro

[10] https://www.greenfacts.org/es/particulas-suspension-pm/index.htm

[11] https://enviraiot.es/cuales-son-gases-contaminantes-de-la-atmosfera/

[12] http://www.cepis.ops-oms.org/bvsci/e/fulltext/meteoro/frame_m2.html

[13] https://www.ptcarretera.es/wp-content/uploads/2015/09/Cuaderno-PTC_1-2011_Sistemas-de-adquisición-de-control-de-tráfico.pdf

[14] http://www.scholarpedia.org/article/Ensemble_learning