

Universidad  
Isabel I

**TRABAJO FIN DE MÁSTER**

CURSO 2019/20

**Estudio de la relación entre tráfico aéreo y niveles de contaminación**

Alumno:

Daniel Arnaiz Gutierrez

Tutor:

Rubén Ruiz González

UNIVERSIDAD ISABEL I

FACULTAD DE CIENCIAS Y TECNOLOGÍA

**MÁSTER OFICIAL EN ANÁLISIS INTELIGENTE DE DATOS MASIVOS (BIG DATA)**

## Resumen

En los últimos años, el aumento de noticias y estudios acerca de la calidad del aire o los efectos del cambio climático ha sido constante. Ciudades como Madrid, se han visto obligadas incluso a limitar el transporte terrestre para así tratar de regular las emisiones de contaminantes que contribuyen a la bajada de la calidad del aire. Sin embargo, las grandes ciudades cuentan con aeropuertos por los que pasan cientos de aviones cada día y para los que no parece haber una regulación en términos de emisiones de contaminantes.

El principal objetivo de este proyecto es tratar de encontrar una posible relación entre la contaminación del aire y las rutas que siguen los vuelos de aviones comerciales en una determinada zona. Para ello, tras delimitar un área y un rango de fechas determinados, y obtener los datos tanto de la contaminación del aire, como de todos los vuelos cuyas rutas han pasado por este área, se ha realizado un estudio para determinar si existe una relación entre estos dos factores.

El proceso de extracción, transformación y carga de datos seguido para realizar este estudio ha sido completado en su totalidad bajo el entorno de Amazon Web Services haciendo uso de múltiples herramientas para la extracción, almacenamiento y procesado de la información como Amazon S3, Amazon EC2 o Amazon EMR. Así mismo, se han empleado librerías de Python para tratamiento de datos como pandas y NumPy, o de visualización de datos como folium, matplotlib o seaborn.

En cuanto a los resultados obtenidos, no se ha encontrado una clara relación directa entre la contaminación del aire y el tráfico aéreo, dados los datos y limitaciones propuestas en este estudio. Sin embargo, la consistencia de los resultados obtenidos de la correlación entre ambos conjuntos de datos sugiere la posibilidad de que, al tener en cuenta contaminantes emitidos por fuentes externas, los efectos de compuestos como óxidos de nitrógeno ( $NO_X$ ) o materias particuladas ( $PM_{10}$  y  $PM_{2.5}$ ) se vean más definidos en los datos observados sobre el tráfico aéreo.

**Palabras clave:** contaminación, aire, tráfico aéreo, Amazon Web Services (AWS), Python, correlación, datos.

## Abstract

In recent years, the increase in news and studies about air quality or the effects of climate change has been constant. Cities like Madrid have even been forced to limit land transport in an attempt to regulate the emissions of pollutants that contribute to the decline in air quality. However, large cities have airports through which hundreds of planes pass every day and for which there seems to be no regulation in terms of pollutant emissions.

The main objective of this project is to try to find a possible relationship between air pollution and the routes followed by commercial aircraft flights in a given area. For that purpose, after defining an area and a range of specific dates, and obtaining data both on air pollution and on all flights whose routes have passed through this area, a study has been carried out to determine whether there is a relationship between these two factors.

The process of data extraction, transformation and loading carried out this study has been completed entirely under the Amazon Web Services environment making use of multiple tools for the extraction, processing and storage of data such as Amazon S3, Amazon EC2 or Amazon EMR. As well as Python libraries for data processing such as pandas and NumPy, or data visualization such as folium, matplotlib or seaborn.

As for the results obtained, no clear direct relationship has been found between air pollution and air traffic, given the data and limitations proposed in this study. However, the consistency of the results obtained from the correlation between both sets of data suggests the possibility that, when taking into account pollutants emitted by external sources, the effects of compounds such as nitrogen oxides ( $NO_x$ ) or particulate matter ( $PM_{10}$  y  $PM_{2.5}$ ) are more defined in the data observed on air traffic.

**Keywords:** pollution, air, air traffic, Amazon Web Services (AWS), Python, correlation, data.

## Índice

1.	Introducción . . . . .	1
2.	Objetivos . . . . .	2
2.1.	Objetivos de carácter general . . . . .	2
2.2.	Objetivos de carácter técnico . . . . .	2
3.	Metodología . . . . .	3
3.1.	Fases del proyecto . . . . .	3
3.2.	Técnicas y herramientas . . . . .	4
3.2.1.	Lenguajes y librerías de programación . . . . .	4
3.2.2.	Herramientas de desarrollo . . . . .	6
3.2.3.	Herramientas de documentación . . . . .	7
3.2.4.	Otras técnicas y herramientas . . . . .	7
4.	Conceptos teóricos . . . . .	9
4.1.	Contaminación del aire . . . . .	9
4.1.1.	Contaminantes . . . . .	9
4.1.2.	Monitorización del aire . . . . .	10
4.2.	Seguimiento de vuelos . . . . .	12
4.2.1.	ADS-B . . . . .	12
4.3.	Trabajos relacionados . . . . .	13
5.	Origen de los datos . . . . .	15
5.1.	Fuentes de datos sobre la contaminación del aire . . . . .	15
5.1.1.	OpenAQ . . . . .	15
5.1.2.	AQICN . . . . .	16
5.1.3.	<i>European Environment Agency</i> . . . . .	17
5.2.	Obtención de los datos sobre la contaminación del aire . . . . .	18
5.3.	Fuentes de datos sobre vuelos . . . . .	19
5.3.1.	OpenFlights . . . . .	19
5.3.2.	FlightRadar24 . . . . .	20
5.3.3.	The OpenSky Network . . . . .	21
5.4.	Obtención de los datos sobre el tráfico aéreo . . . . .	22
6.	Obtención de los datos . . . . .	24
6.1.	Extracción de datos sobre contaminación del aire . . . . .	24
6.2.	Extracción de datos sobre tráfico aéreo . . . . .	26
7.	Ánalisis y procesado de los datos . . . . .	29
7.1.	Infraestructura empleada . . . . .	29
7.1.1.	Configuración . . . . .	29
7.2.	Análisis exploratorio . . . . .	32
7.2.1.	Contaminación del aire . . . . .	32
7.2.2.	Tráfico aéreo . . . . .	38
7.3.	Transformación de los datos . . . . .	40
7.3.1.	Contaminación del aire . . . . .	40
7.3.2.	Tráfico aéreo . . . . .	44
8.	Resultados . . . . .	46
9.	Conclusiones . . . . .	49
10.	Líneas de trabajo futuras . . . . .	50
	Referencias bibliográficas . . . . .	52
	Anexos . . . . .	53

## Índice de figuras

1.	Comparación del tamaño de $PM_{10}$ y $PM_{2.5}$ con un cabello humano . . . . .	10
2.	Diagrama del funcionamiento de un filtro de partículas PM2.5 y PM10 . . . . .	11
3.	Estación de monitorización de contaminación del aire . . . . .	11
4.	Diagrama del funcionamiento del sistema ADS-B . . . . .	12
5.	Visualización de datos de OpenAQ en un mapa . . . . .	15
6.	Visualización de datos de AQICN en un mapa . . . . .	16
7.	Página de descarga del <i>dataset</i> de la web de EEA . . . . .	17
8.	Visualización del <i>dataset</i> de rutas del proyecto Openflights . . . . .	20
9.	Visualización los datos de Flightradar24 sobre Europa . . . . .	21
10.	Visualización de la cobertura de receptores a fecha 20-01-2010 por OpenSky Network . . . . .	22
11.	Tablas disponibles en la base de datos de OpenSky Network . . . . .	22
12.	Estructura de la tabla <i>state_vectors_data4</i> . . . . .	23
13.	Descripción de los campos de la URL utilizados para filtrar los datos sobre la contaminación del aire . . . . .	25
14.	Estructura JSON de los datos de las estaciones de calidad del aire . . . . .	25
15.	Resumen de la configuración de la instancia EC2 utilizada . . . . .	27
16.	Explicación de la <i>query</i> utilizada para extraer información desde la base de datos de OpenSky . . . . .	27
17.	Mapa que muestra el área sobre el que se extraen los datos del tráfico aéreo . .	28
18.	Apartado de configuración de <i>software</i> del clúster EMR utilizado . . . . .	30
19.	Apartado de configuración de <i>hardware</i> del clúster EMR utilizado . . . . .	30
20.	Resumen de la configuración del clúster EMR . . . . .	31
21.	Entorno de JupyterLab con los <i>notebooks</i> usados en el proyecto . . . . .	31
22.	Contenido del <i>dataframe</i> con las estaciones de contaminación del aire . . . .	32
23.	Contenido del <i>dataframe</i> final con las estaciones de contaminación del aire españolas . . . . .	32
24.	Mapa que muestra la localización de las estaciones seleccionadas para la realización del estudio . . . . .	33
25.	Contenido del <i>dataframe</i> con mediciones sobre la contaminación del aire . . .	34
26.	Contenido del <i>dataframe</i> con mediciones sobre la contaminación del aire después del filtrado . . . . .	34
27.	Contenido del <i>dataframe</i> sobre la contaminación del aire e información de su posición . . . . .	34
28.	Gráfico de barras que muestra el número de entradas para cada tipo de contaminante	35
29.	Conjunto de histogramas que muestran la distribución de los valores medidos para cada uno de los contaminantes . . . . .	36
30.	Mapa en el que se muestra la posición de las observaciones de monóxido de carbono junto con su concentración . . . . .	37
31.	Mapa en el que se muestra la posición de las observaciones de dióxido de nitrógeno junto con su concentración . . . . .	37
32.	Contenido del <i>dataframe</i> con los datos sobre tráfico aéreo . . . . .	38
33.	Contenido del <i>dataframe</i> con los datos sobre tráfico aéreo una vez preprocesados	38
34.	Contenido del <i>dataframe</i> con los datos sobre tráfico aéreo una vez preprocesados	39
35.	Mapa de calor que muestra el tráfico aéreo sobre la Península Ibérica el día 1 de enero de 2020 . . . . .	40

36.	Histogramas con los datos sobre contaminación del aire después de eliminar los valores atípicos encontrados . . . . .	41
37.	Ejemplo del resultado de la interpolación de valores de Ozono para un día concreto sobre el mapa de la Península Ibérica . . . . .	42
38.	<i>Dataframe</i> final sobre la contaminación del aire después de aplicar todas las transformaciones requeridas . . . . .	43
39.	Histograma de los campos <i>Altitude</i> y <i>Velocity</i> después de haber eliminado sus valores atípicos . . . . .	44
40.	<i>Dataframe</i> final sobre el tráfico aéreo después de aplicar todas las transformaciones requeridas . . . . .	45
41.	Conjunto de diagramas de dispersión entre los campos del <i>dataset</i> final . . . . .	47
42.	Matriz de correlación que muestra los coeficientes calculados para todos los pares de campos del <i>dataset</i> . . . . .	48

## Índice de tablas

1.	Estructura del <i>dataset</i> sobre la contaminación del aire . . . . .	19
2.	Estructura del <i>dataset</i> sobre el tráfico aéreo . . . . .	23
3.	Estructura del <i>dataset</i> final de las estaciones de medición de la contaminación del aire en España . . . . .	33
4.	Estructura del <i>dataset</i> final sobre la contaminación del aire . . . . .	35
5.	Estructura del <i>dataset</i> final sobre el tráfico aéreo . . . . .	39
6.	Estructura del dataset sobre la contaminación del aire después de aplicar las transformaciones sobre sus datos . . . . .	43
7.	Estructura del dataset sobre el tráfico aéreo después de aplicar las transformaciones sobre sus datos . . . . .	45
8.	Estructura del dataset resultante de la unión entre los datos sobre contaminación del aire y tráfico aéreo . . . . .	46

## 1. Introducción

A pesar de décadas de progreso, la calidad del aire en todo el mundo ha empeorado en los últimos años. Según datos del último informe de la Agencia Europea del Medio Ambiente (Agency, 2019), en 2017 las estaciones pertenecientes a la Unión Europea registraron un aumento del 22 % de los días en los que el límite saludable recomendable de contaminación fue sobrepasado (Agency, 2020).

La contaminación del aire es la mezcla de una serie de gases y partículas que, en determinada concentración, pueden llegar a ser perjudiciales para la salud tanto en entornos abiertos como cerrados. A parte de los riesgos saludables que suponen, también pueden llegar a afectar al medio ambiente con subidas en las temperaturas (Parra, 2020).

Aunque existen diversos criterios para definir los contaminantes con mas repercusión sobre la calidad del aire, los principales compuestos a tener en cuenta son el monóxido de carbono (CO), varias composiciones de óxidos de nitrógeno (NOX), ozono (O<sub>3</sub>), óxido de azufre (SO) y materia particulada (PM).

Las razones de la reciente disminución en la calidad del aire en todo el mundo siguen sin estar claras. El informe mencionado hace alusiones a incendios forestales, al cambio climático y al aumento de la población mundial. Sin embargo en otros informes y medios se relaciona con otros factores como la industria, la incineración de desechos o el uso de combustibles fósiles en vehículos.

Por otro lado, el continuo incremento en la demanda del transporte de la aviación comercial genera nuevas cuestiones sobre los efectos que supone en el medio ambiente y en concreto en la calidad y contaminación del aire.

La tecnología usada actualmente para controlar el tráfico aéreo, llamada ADS-B (Sistema de Vigilancia Dependiente Automática), se basa en la emisión periódica de datos desde los propios aviones posibilitando así la extracción de grandes cantidades de datos precisos sobre la posición, altitud y velocidad de la aeronave en todo momento.

En cuanto a la disponibilidad de los datos sobre ambos campos, existen multitud de organizaciones y repositorios *open data* desde los que acceder a datos de diversas fuentes y tipos con el objetivo de ser analizados y estudiados. Algunos ejemplos son FlightRadar24 y OpenSky Network para datos sobre el tráfico aéreo, o OpenAQ y *European Environment Agency* para conjuntos de datos sobre la contaminación del aire.

Este es el caso de proyectos como este, en el que se relacionarán datos provenientes de ambos campos con el objetivo de encontrar patrones que puedan ayudar a entender los posibles efectos del tráfico aéreo en la calidad del aire y en los niveles de los distintos contaminantes encontrados en éste.

## 2. Objetivos

Este apartado explica de forma precisa y concisa cuáles son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

### 2.1. Objetivos de carácter general

El principal objetivo de este proyecto es estudiar la relación entre el tráfico aéreo y los niveles de contaminación del aire identificando cómo afectan las rutas que toman estos vuelos a los resultados que las estaciones encargadas de medir la calidad del aire recogen.

A su vez, este proceso está dividido en diferentes fases que podemos identificar como objetivos secundarios:

- Identificar y extraer los datos necesarios para realizar el estudio.
- Preparar los datos extraídos seleccionando y transformando los campos que se consideren determinantes.
- Calcular la correlación entre los datos una vez hayan sido procesados.
- Identificar los efectos, si existiesen, del tráfico aéreo sobre la contaminación del aire en función de los resultados de la correlación obtenida gracias a los pasos anteriores.

### 2.2. Objetivos de carácter técnico

A continuación se listan los objetivos técnicos utilizados para la correcta implementación de los objetivos de carácter más general expuestos en el anterior apartado:

- Hacer uso del entorno de computación en la nube de Amazon AWS (*Amazon Web Services*) para varias tareas del proyecto:
  - Amazon EC2 para la creación de instancias desde las cuales se extraerá parte de los *datasets* utilizados.
  - Amazon EMR para la creación de clústeres Hadoop donde se realizarán la mayoría del procesamiento de los datos.
  - Amazon S3 para el almacenamiento de los *datasets* empleados así como de los resultados finales e intermedios.
- Utilizar Python como lenguaje de programación para la totalidad del proyecto.
- Usar JSON y CSV como formas de almacenamiento temporal de los datos debido a la simplicidad de sus estructuras y la facilidad de su utilización.

### 3. Metodología

En este apartado se va a tratar de explicar el proceso seguido para la realización de este proyecto y las principales fases que lo conforman, así como las técnicas y herramientas utilizadas.

#### 3.1. Fases del proyecto

##### Investigación

La primera fase de todo proyecto es la investigación acerca de los temas a tratar. Por un lado, se ha necesitado recavar información sobre los posibles contaminantes de aire, así como las formas en que se puede medir y los riesgos que suponen tanto para las personas como para el medio ambiente.

Por otro lado, en cuanto al tráfico aéreo, es necesario entender la tecnología utilizada para emitir y recibir la información sobre los aviones, e identificar los diversos campos de información que serán utilizados para la realización del proyecto.

Esta fase también incluye la selección de las fuentes de datos desde las cuales se descargará los *datasets* que más tarde serán procesados y analizados.

##### Obtención de los conjuntos de datos

La obtención de los conjuntos de datos se basa en las opciones que ofrece cada una de las fuentes de datos seleccionadas. En esta fase se diseña el proceso de descarga de los datos y de su posterior almacenamiento.

##### Análisis exploratorio de los datos

El siguiente paso es entender los datos que hemos obtenido. En esta fase del proyecto se han de analizar los *datasets* e identificar sus principales características con ayuda de visualizaciones de datos y otras técnicas.

El objetivo de esta fase es decidir las técnicas y herramientas que se van a utilizar para procesar ambos conjuntos de datos.

##### Transformación de los datos

En esta fase se van a procesar los conjuntos de datos de tal forma que los *datasets* resultantes tengan una estructura similar y puedan ser analizados y visualizados conjuntamente. Algunas de las técnicas utilizadas en esta fase han sido la interpolación de datos o la identificación de *outliers*.

## Obtención de resultados

Por último, una vez que se dispone de los datos limpios y preparados, es el momento de tratar de responder a la cuestión principal de este trabajo. Para ello, se han de correlar ambos conjuntos de datos y evaluar los resultados obtenidos.

### 3.2. Técnicas y herramientas

#### 3.2.1. Lenguajes y librerías de programación

##### Python

Python<sup>1</sup> es un lenguaje de programación nacido en los años 90 y ampliamente utilizado en el presente. Se trata de un lenguaje interpretado, dinámicamente tipado y multiplataforma. Soporta programación orientada a objetos y sigue una filosofía de legibilidad y transparencia.

Ha sido el lenguaje utilizado en la mayor parte del proyecto, desde el proceso de carga y transformación de los datos, hasta la creación y visualización de los resultados obtenidos.

##### Pandas

Pandas<sup>2</sup> es una potente librería de Python de manipulación y análisis de datos totalmente *open-source*. Su desarrollo comienza en el año 2008, y entre sus principales características encontramos el uso de objetos *Dataframe*<sup>3</sup>, la capacidad leer y escribir ficheros de diversos tipos o funcionalidades de marcas de tiempo.

El nivel de optimización y rendimiento que ofrece, junto con las capacidades ya mencionadas, hace de esta librería una herramienta indispensable para el desarrollo de proyectos de esta índole.

##### NumPy

Esta librería está basada en un proyecto *open-source* cuya finalidad es la computación numérica en Python. Fue creado en 2005 basándose en las librerías Numerical y Numarray.

Las principales características de NumPy<sup>4</sup> son las siguientes:

- Optimizado y de alto rendimiento.
- Interoperabilidad: Soporta un amplia gama de *hardware* y plataformas de computación.

<sup>1</sup><https://www.python.org/>

<sup>2</sup><https://pandas.pydata.org/>

<sup>3</sup><https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

<sup>4</sup><https://numpy.org/>

- Facilidad de uso: Ofrece un sintaxis de alto nivel con el objetivo de ser accesible a desarrolladores con y sin experiencia.
- Compatibilidad con *arrays* N-dimensionales: Soporta indexado y vectorización de forma rápida y versátil.

## Matplotlib

Matplotlib<sup>5</sup> es una librería que se centra en la creación de visualizaciones estáticas, animadas o interactivas en Python.

Esta librería ha sido indispensable en la realización de este proyecto, ya que ha sido utilizado para la visualización de los datos obtenidos y la extracción de información incluso en fases tempranas del estudio.

## Seaborn

La librería Seaborn<sup>6</sup> se centra en la visualización de datos y se basa en la anteriormente mencionada Matplotlib. Proporciona una interfaz de alto nivel capaz de generar gráficos estadísticos atractivos a la par de informativos.

## SciPy

SciPy<sup>7</sup> es una librería de Python usada para resolver problemas científicos y matemáticos. Es una de las librerías más utilizadas en análisis numéricos y matemáticos debido a su capacidad de operar con *arrays* de datos en tareas de ordenación, indexado, etc.

En este trabajo ha sido utilizado para transformar los conjuntos de datos y calcular nuevos campos a partir de los ya existentes. Por otro lado, ha sido indispensable para calcular la correlación entre ambos *datasets* y por lo tanto en la obtención de los resultados finales.

## Folium

Folium<sup>8</sup> se trata de otra librería de visualización de datos para Python, sin embargo, la principal cualidad que la diferencia del resto es su capacidad de visualizar datos geográficos. Está basada en la librería de JavaScript Leaflet.js<sup>9</sup> y su facilidad de uso, así como su potencia, han sido de gran ayuda para la realización del proyecto.

---

<sup>5</sup><https://matplotlib.org/>

<sup>6</sup><https://seaborn.pydata.org/>

<sup>7</sup><https://www.scipy.org/>

<sup>8</sup><https://python-visualization.github.io/folium/>

<sup>9</sup><https://leafletjs.com/>

### 3.2.2. Herramientas de desarrollo

#### Amazon S3

Amazon S3<sup>10</sup> (*Simple Storage Service*) es un servicio de almacenamiento en la nube ofrecido por la plataforma de computación en la nube *Amazon Web Services* (AWS), cuyas principales características son su escalabilidad, velocidad de respuesta y la facilidad de uso de su interfaz web.

Todos los conjuntos de datos utilizados en este proyecto, así como los resultados finales y parciales han sido almacenados en esta plataforma.

#### Amazon EC2

Este servicio de la plataforma AWS proporciona la capacidad de crear instancias de máquinas virtuales fácilmente configurables y escalables desde la comodidad de una interfaz web.

Para este trabajo, Amazon EC2<sup>11</sup> ha sido utilizado para conectarse a la base de datos de OpenSky y descargar el conjunto de datos.

#### Amazon EMR

Amazon EMR<sup>12</sup> (*Elastic MapReduce*) es una herramienta ofrecida por AWS centrada en el procesado y análisis de *big data*.

Sus principales características pasan por la gran compatibilidad que ofrece con otras herramientas, su alto rendimiento a un precio reducido y la disponibilidad y escalabilidad que ofrece.

El procesado y análisis de los datos de este proyecto han sido ejecutados bajo esta herramienta.

#### Google Colab

Google Colab<sup>13</sup> es un entorno de Jupyter Notebook<sup>14</sup> ya configurado y completamente ejecutado en remoto. Permite editar y ejecutar código, guardar y compartir proyectos y tener acceso a potentes recursos desde un navegador web.

Para este proyecto, esta herramienta ha sido usada a la hora de realizar algunas visualizaciones de datos debido a su rápida puesta en marcha y la facilidad de instalación de librerías externas.

---

<sup>10</sup><https://aws.amazon.com/s3/>

<sup>11</sup><https://aws.amazon.com/ec2/>

<sup>12</sup><https://aws.amazon.com/emr/>

<sup>13</sup><https://colab.research.google.com>

<sup>14</sup><https://jupyter.org/>

### 3.2.3. Herramientas de documentación

#### **L<sup>A</sup>T<sub>E</sub>X**

L<sup>A</sup>T<sub>E</sub>X<sup>15</sup> es un sistema de composición de textos orientado a la creación de documentos con una alta calidad tipográfica como podrían ser artículos científicos u otros textos con, por ejemplo, expresiones matemáticas.

Se trata de un software libre basado en un conjunto de macros de T<sub>E</sub>X, que a su vez se trata de un lenguaje escrito por Leslie Lamport en 1984.

#### **Overleaf**

Overleaf<sup>16</sup> es un editor en línea de documentos L<sup>A</sup>T<sub>E</sub>X muy intuitivo y fácil de usar. No requiere de instalación y ofrece diversas características como colaboración en tiempo real, control de versiones o el uso de plantillas L<sup>A</sup>T<sub>E</sub>X.

### 3.2.4. Otras técnicas y herramientas

#### **PuTTY**

PuTTY<sup>17</sup> es un emulador de terminal de software para Windows y Linux que proporciona una interfaz de usuario de texto a ordenadores remotos bajo diversos protocolos como SSH<sup>18</sup> o Telnet<sup>19</sup>.

#### **JSON**

JSON<sup>20</sup> (*JavaScript Object Notation*) es un formato de texto utilizado para el intercambio de datos que desde el año 2019 es considerado como un formato independiente de lenguaje.

La principal característica de este lenguaje y la razón por la que se ha utilizado en este proyecto es la facilidad de extraer los datos que lo conforman con todo tipo de lenguajes y en concreto librerías como Pandas.

---

<sup>15</sup><https://es.wikipedia.org/wiki/LaTeX>

<sup>16</sup><https://es.overleaf.com/>

<sup>17</sup><https://www.putty.org/>

<sup>18</sup>[https://es.wikipedia.org/wiki/Secure\\_Shell](https://es.wikipedia.org/wiki/Secure_Shell)

<sup>19</sup><https://es.wikipedia.org/wiki/Telnet>

<sup>20</sup><https://es.wikipedia.org/wiki/JSON>

## CSV

CSV<sup>21</sup> (*Comma-separated values*) es un formato de fichero de texto que usa una coma para separar sus valores en el que cada línea se corresponde a un registro formado por una serie de campos separados por comas.

Al igual que el formato JSON ya mencionado, este tipo de archivos semi-estructurados han sido utilizados para almacenar los datos usados en este trabajo.

---

<sup>21</sup>[https://es.wikipedia.org/wiki/Valores\\_separados\\_por\\_comas](https://es.wikipedia.org/wiki/Valores_separados_por_comas)

## 4. Conceptos teóricos

Este apartado recoge los aspectos técnicos más importantes con el objetivo de aportar un contexto y abordar las claves para entender los temas tratados en el proyecto. Dado que los principales tópicos son la contaminación del aire y las rutas aéreas comerciales, se tratarán de abordar por separado, tanto conceptos teóricos básicos, como la generación y obtención de datos sobre ambos campos.

### 4.1. Contaminación del aire

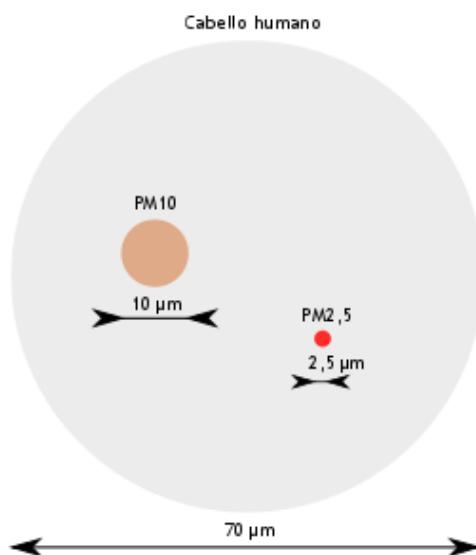
#### 4.1.1. Contaminantes

Los principales contaminantes que se pueden encontrar en el aire son los siguientes:

- **Contaminación por partículas:** Se trata de una mezcla de partículas suspendidas en el aire, también conocida como materia particulada, compuesta por una serie de componentes entre los que se pueden encontrar ácidos, amoníaco, metales, partículas de polvo o alérgenos. El tamaño de estas partículas es variable y se clasifican en función del diámetro que toman. Las dos principales clasificaciones de estas partículas son las siguientes:
  - Partículas finas o  $PM_{2.5}$ <sup>22</sup>: Son aquellas con un diámetro menor a 2,5 micras. Estas partículas únicamente pueden detectarse con un microscopio electrónico. Las principales fuentes que generan estas partículas con la combustión de varios elementos tales como los motores de vehículos, plantas energéticas o incendios forestales.
  - Partículas de polvo grueso o  $PM_{10}$ <sup>23</sup>: Se trata de partículas de entre 2,5 y 10 micras de diámetro que a menudo son generadas en actividades industriales y agrícolas.
- **Ozono al nivel del suelo o *ground-level ozone*:** Es un gas corrosivo e incoloro formado sobre la superficie de la Tierra con efectos significativos sobre la salud además de sobre la vegetación. Se produce con la reacción de otros contaminantes como los óxidos de nitrógeno y algunos componentes orgánicos.
- **Monóxido de carbono:** Este gas incoloro e inodoro es producto de la combustión de combustibles basados en hidrocarburos en vehículos, industrias e incendios forestales. El monóxido de carbono tiene un impacto significativo en la salud siendo los mas susceptibles, los niños, ancianos y personas con problemas de corazón.
- **Óxidos de azufre:** El dióxido de azufre o  $SO_2$  pertenece a la familia de los gases de óxidos de azufre ( $SO_x$ ) y está formado por el azufre que se encuentra en materiales como el carbón, el petróleo o metales pesados durante los procesos de refinado y combustión. Este componente se disuelve en vapor de agua una vez en el aire y forma ácidos que interaccionan con otros gases y partículas para formar partículas llamadas sulfatos que pueden llegar a ser dañinas para las personas y el medio ambiente.

<sup>22</sup><https://es.wikipedia.org/wiki/PM2.5>

<sup>23</sup><https://es.wikipedia.org/wiki/PM10>



*Figura 1.* Comparación del tamaño de  $PM_{10}$  y  $PM_{2.5}$  con un cabello humano  
 Imagen extraída de <https://es.wikipedia.org/wiki/PM10>

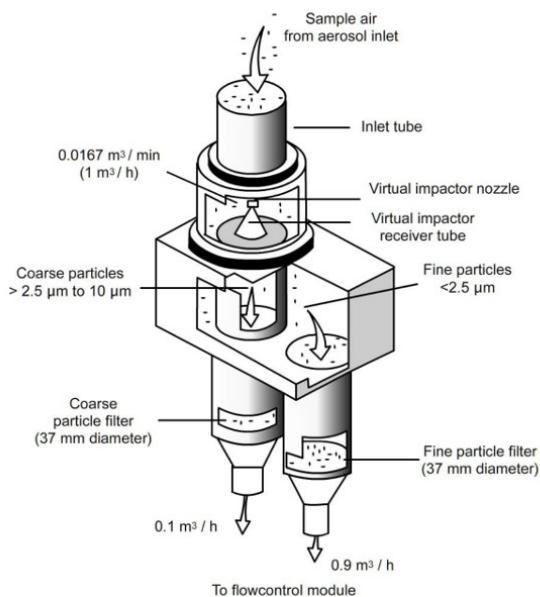
- **Óxidos de nitrógeno:** Incluyen el óxido de nitrógeno ( $NO$ ) y el dióxido de nitrógeno ( $NO_2$ ). Estos gases se forman con la liberación del nitrógeno que se encuentra en varios combustibles así como en diversos procesos de combustión.

Se disuelve en vapor de agua y forma ácidos que al entrar en contacto con otros gases y partículas en el aire, forma nitratos y otros productos dañinos para las personas y el entorno.

#### 4.1.2. Monitorización del aire

La contaminación del aire se monitoriza las 24 horas del día en estaciones a lo largo de todo el mundo. Existen diversas formas de tomar muestras del aire y comprobar su composición y el nivel de cada uno de sus componentes. A continuación describen los principales métodos de monitorización del aire:

- **Monitorización pasiva:** Este tipo de estaciones no necesitan de energía para funcionar ya que se basan en la absorción de contaminantes específicos en tubos de difusión que más tarde han de ser analizados en una laboratorio con el objetivo de medir cuánta contaminación se ha detectado.
- **Muestreo activo (semiautomático):** El funcionamiento de estas estaciones se basa en el filtrado de una muestra de aire durante un determinado periodo de tiempo, por ejemplo una vez cada 24 horas. Al igual que en el anterior tipo, estas muestras han de ser analizadas en un laboratorio para detectar el nivel de contaminación que han recogido.
- **Monitorización automática:** Estas estaciones son las más utilizadas debido al modo en que recogen y envían sus resultados. El aire pasa por un analizador que reconoce los gases



*Figura 2.* Diagrama del funcionamiento de un filtro de partículas PM2.5 y PM10.

Imagen extraída de <https://aqicn.org/products/monitoring-stations/>

elegidos y calcula su concentración en el momento. Este tipo de sensores trabajan las 24 horas del día y los datos recogidos se envían instantáneamente a los centros de datos.

- **Sensores fotoquímicos y ópticos:** La principal característica de estas estaciones es que son portátiles. Son capaces de monitorizar de forma continua una serie de contaminantes para más tarde descargar los datos recogido en un ordenador o servidor de forma manual.
- **Monitorización a larga distancia:** Este tipo de estaciones basan su funcionamiento en la detección de contaminación entre una fuente de luz y un detector ubicados a gran distancia entre sí. Estas estaciones reportan sus resultados en tiempo real.



*Figura 3.* Estación de monitorización de contaminación del aire.

Imagen extraída de <https://aqicn.org/products/monitoring-stations/>

## 4.2. Seguimiento de vuelos

### 4.2.1. ADS-B

El sistema ADS-B, en inglés *Automatic Dependent Surveillance Broadcast*, es un equipamiento electrónico que los aviones llevan equipados con el objetivo de reportar en tiempo real información del mismo, como podría ser la localización o la velocidad (Wikipedia, 2020).

Estos datos son utilizados por los controladores aéreos, así como por otros aviones para tener constancia de la localización del resto de aviones sin que sea necesario hacer uso de un radar.

Este sistema consiste en que las aeronaves con un emisor ADS-B determinen su posición con un GPS y la transmitan en intervalos rápidos junto con códigos identificativos y otro tipo de datos. Las estaciones dedicadas reciben estas señales y devuelven la información a los controladores aéreos para así tener un seguimiento detallado y preciso de los aviones en circulación («How ADS-B works», s.f.).

Las principales características sobre la utilización de este sistema se resumen en los siguientes puntos, que además se corresponden a sus siglas en inglés:

- **Automatic:** Se trata de un sistema totalmente automático en el que el piloto no debe realizar ninguna acción.
- **Dependent:** El sistema depende de los datos precisos de la posición y velocidad que el sistema de navegación (GPS) le reporta.
- **Surveillance:** Proporciona estos datos, junto con códigos identificativos de cada aeronave a aquellos servicios o infraestructuras que requieren esta información.
- **Broadcast:** Estos datos están continuamente siendo transmitidos y monitorizados en estaciones dedicadas a lo largo de todo el mundo.

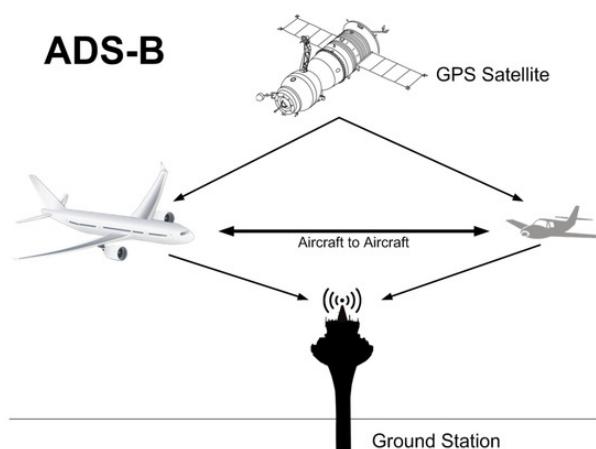


Figura 4. Diagrama del funcionamiento del sistema ADS-B.

Imagen extraída de <https://www.mdpi.com/1424-8220/17/1/188>

La estructura que siguen las transmisiones del sistema ADS-B se compone de los siguientes campos, entre otros:

- **Identificador del vuelo:** Número de vuelo único.
- **ICAO24:** Identificador único asociado a cada nave.
- **Posición:** Latitud y longitud.
- **Altitud:** Altitud barométrica y geométrica.
- **Tasa vertical:** Ratio de escalada o desescalada del avión.
- **Velocidad:** Velocidad a la que se mueve el avión.

Cabe destacar que la operatividad de las estaciones dedicadas depende de la altitud, distancia y posibles obstrucciones por el terreno. El rango máximo operativo de estas estaciones es de alrededor de 250 millas náuticas (450 kilómetros).

#### 4.3. Trabajos relacionados

Parte de la investigación llevada a cabo ha consistido en la búsqueda de otros trabajos y estudios relacionados con esta propuesta, con el objetivo de tomar ciertos conceptos como referencia o tener en cuenta sus resultados y metodologías a la hora de plantear la realización de este proyecto.

Uno de ellos, con objetivos similares a este trabajo es el artículo con el título *Impacts of aircraft emissions on the air quality near the ground* (Lee y col., 2013) cuyo tema principal es el estudio sobre el impacto de las emisiones de aeronaves sobre la calidad del aire. En este artículo se investiga, haciendo uso de múltiples modelos y simulaciones, los efectos de las emisiones en distintos niveles de la troposfera<sup>24</sup>.

Los resultados obtenidos en este estudio indican que las emisiones en altitudes de crucero de alrededor de 10 kilómetros son más responsables que las emisiones durante despegues y aterrizajes para la mayor parte del óxido de nitrógeno, ozono y aerosoles observados en mediciones hechas a nivel del suelo. Por otro lado, también indica que los resultados obtenidos varían notablemente en función de la estación del año en la que se encuentre debido al clima y las temperaturas.

Por otro lado, el artículo *Climate impact of air traffic emissions in dependency of the emission location and altitude* (Fichter, 2009) detalla los efectos del tráfico aéreo sobre el clima en función de la posición y altitud por medio de simulaciones de sus emisiones. Se han tomado como referencia los resultados que obtiene, ya que hacen alusión a que los efectos aumentan con la altitud en la que los distintos contaminantes son emitidos.

---

<sup>24</sup><https://es.wikipedia.org/wiki/Troposfera>

En cuanto a estudios sobre la contaminación del aire mas generales, encontramos un artículo titulado *The air of Europe: where are we going?* (Annesi-Maesano, 2017) centrado en el impacto que supone sobre la salud en entornos cuyos niveles se encuentran entre los límites establecidos por la Unión Europea.

## 5. Origen de los datos

Esta sección recoge las fuentes de datos utilizadas para la realización del proyecto. En primer lugar se detallarán las diversas opciones contempladas para el recavado de datos sobre la contaminación del aire, así como de las rutas aéreas. Y, por último, se expondrá el funcionamiento de la fuente de datos escogida junto con los aspectos más relevantes sobre la obtención de estos datos.

### 5.1. Fuentes de datos sobre la contaminación del aire

Existen diversas páginas web con datos sobre este tópico ofrecidas al público para la realización de estudios de investigación o visualización en tiempo real. La mayoría de alternativas ofrecen un servicio abierto y gratuito aunque con algunas restricciones, a continuación se detallan algunas de las opciones contempladas:

#### 5.1.1. OpenAQ

OpenAQ<sup>25</sup> es una plataforma *open-source* sin ánimo de lucro con el objetivo de agregar, estandarizar y publicar datos sobre la calidad del aire de todo el mundo. Este proyecto permite no sólo visualizar de forma sencilla los datos recogidos, sino además construir nuevas herramientas o realizar estudios de investigación.

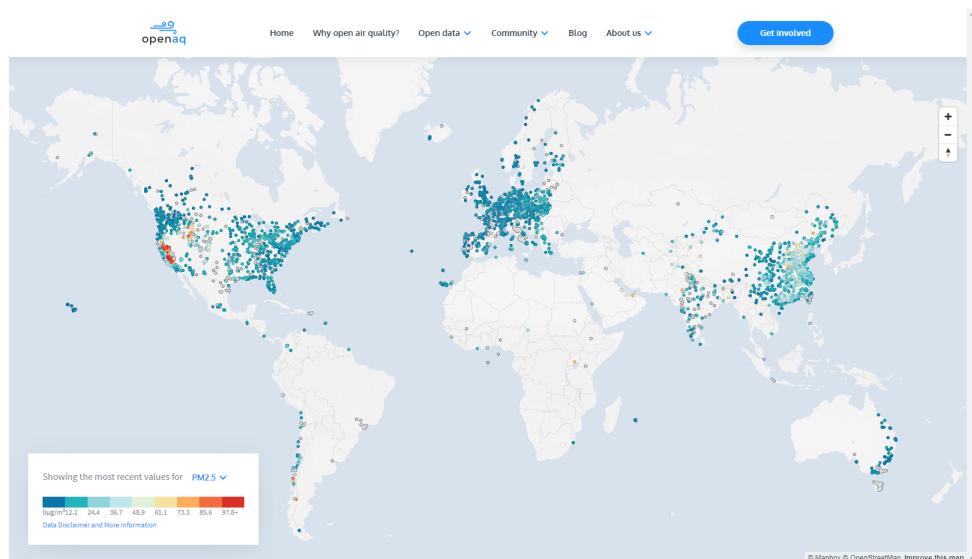


Figura 5. Visualización de datos de OpenAQ en un mapa. «OpenAQ», 2020

Por el momento, OpenAQ recoge datos de 93 países diferentes gracias a plataformas *open data* de gobiernos y otras investigaciones. El archivo almacena datos accesibles de los últimos 2 años con información acerca de la concentración de contaminantes como *PM2.5*, *PM10*, ozono, dióxidos de azufre, dióxido de nitrógeno o monóxido de carbono.

<sup>25</sup><https://openaq.org/>

El acceso a los datos puede realizarse por medio de una API pública o descargando el *dataset* completo de un país determinado para un rango de fechas previamente delimitadas. La API tiene un límite de 2000 peticiones cada 5 minutos y únicamente devuelve datos de los últimos 90 días. Sin embargo, existe un repositorio<sup>26</sup> en Amazon S3 con los datos históricos desde abril de 2018.

### 5.1.2. AQICN



Figura 6. Visualización de datos de AQICN en un mapa. «Real-time Air Quality Index», 2020

Este proyecto también conocido como *World Air Quality Index* es un proyecto sin ánimo de lucro fundado en Pekín, China en 2007. Su misión es promover la sensibilización sobre la contaminación del aire y proporcionar información unificada sobre la calidad del aire alrededor de todo el mundo.

Actualmente, esta plataforma proporciona información sobre la calidad del aire de más de 130 países con datos provenientes de más de 30.000 estaciones situadas en 2.000 ciudades. Estos datos son accesibles desde dos páginas web: [aqicn.org](https://aqicn.org/)<sup>27</sup> y [waqi.info](https://waqi.info/)<sup>28</sup>.

El acceso a los datos ofrecidos por esta plataforma cuenta con filtros para fechas y localizaciones, pudiendo acceder a los datos históricos de países, ciudades o incluso estaciones concretas. Por otro lado, cuenta con una API con la que acceder a datos en tiempo real. El acceso a esta herramienta está restringido a usuarios con un *token* que puede ser solicitado en la misma página.

La API cuenta con tres servicios diferenciados, el primero, llamado *Map tile API* tiene el objetivo de servir datos directamente sobre mapas interactivos ya sean de Google, Bing u OpenStreetMaps. *Widget API* puede ser usado para integrar *wIDGETS* en cualquier página web con información en tiempo real de diferentes índices de calidad del aire. Por último, la API principal proporciona datos estructurados en ficheros JSON con información en tiempo real de más de 11.000 estaciones y 1.000 ciudades con información sobre la localización, concentración, condiciones meteorológicas e incluso predicciones de la calidad del aire para los próximos días.

<sup>26</sup><https://openaq-data.s3.amazonaws.com/index.html>

<sup>27</sup><https://aqicn.org/>

<sup>28</sup><https://waqi.info/>

### 5.1.3. European Environment Agency

La EEA (*European Environment Agency*)<sup>29</sup> es una agencia de la Unión Europea con la tarea de proporcionar información sólida e independiente sobre el medio ambiente. El principal objetivo de esta plataforma es apoyar el desarrollo sostenible ayudando a lograr una mejora significativa en este entorno mediante el acceso libre a información específica y fiable a otras agencias, gobiernos y público general.

Esta agencia está asociada con Eionet<sup>30</sup> (*European environment information and observation network*) y a todos los países que cooperan en esta red. Gracias a esto, los datos con los que cuenta la EEA cuentan con la validación de los países de los que provienen.

La información que recoge esta asociación está disponible a través de la web de la EEA en forma no sólo de *datasets* sino también de mapas interactivos, informes o gráficos de diversos tópicos relacionados con el medio ambiente de Europa.

En relación a los datos requeridos para este proyecto, la agencia cuenta con una amplia base de datos de acceso libre con múltiples filtros sobre los países, ciudades, años o contaminantes sobre los que se requiere descargar los datos. El resultado de esta descarga es un archivo CSV compuesto por 17 campos entre los que se encuentran el código identificativo de la estación que ha hecho la medición, el contaminante, la concentración medida, o la fecha de la medición entre otros.

**Download form**  
 The form below will help you to build the request URL to get the list of files to download matching your criteria.  
 Before executing the URL it is possible to refine the request, e.g. by adding a specific station or leaving a parameter blank.  
 Note: Country, City and Pollutant are interlinked and changing the country will cause the others to change.

Country	AD Andorra
City name	
Pollutant	CO
Year from	2020
Year to	2020
Source	E1a
Output type	HTML
Update date	
Time coverage	Year

Update request URL  
  
 Download

Figura 7. Página de descarga del *dataset* de la web de EEA.

<sup>29</sup><https://www.eea.europa.eu/>

<sup>30</sup><https://www.eea.europa.eu/about-us/countries-and-eionet>

## 5.2. Obtención de los datos sobre la contaminación del aire

La fuente de datos elegida sobre las anteriormente expuestas, ha sido la EEA. La principal razón por la que se ha elegido ha sido por el número de estaciones en España con las que opera y la facilidad de acceso a estos datos. El sistema de filtrado y posterior descarga de los datos, así como la estructura que siguen los ficheros descargados, hacen de esta opción las más indicada para la obtención de los datos necesarios para completar el estudio de este proyecto.

El *dataset* descargado se compone de 3.073 archivos CSV que contienen los datos previamente seleccionados con los siguientes filtros:

- **País:** España
- **Ciudad:** Todas las ciudades
- **Contaminante:** Todos los contaminantes
- **Año:** 2020

La estructura de cada uno de estos archivos se compone de los siguientes campos:

Tabla 1  
*Estructura del dataset sobre la contaminación del aire*

Campo	Tipo	Descripción
<i>Countrycode</i>	String	Código identificativo del país de origen.
<i>Namespace</i>	String	Nombre único proporcionado por el país de origen.
<i>AirQualityNetwork</i>	String	Identificador de la red a la que pertenece.
<i>AirQualityStation</i>	String	Identificador local de la estación.
<i>AirQualityStationEoICode</i>	String	Identificador <i>AirBase</i> de la estación.
<i>Samplingpoint</i>	String	Identificador local del punto de muestreo.
<i>SamplingProcess</i>	String	Identificador local del proceso de muestreo.
<i>Sample</i>	String	Identificador local de la muestra.
<i>AirPollutant</i>	String	Abreviatura del contaminante medido.
<i>AirPollutantCode</i>	String	Referencia (URL) a la definición del contaminante medido.
<i>AveragingTime</i>	String	Define el tiempo en el que se ha hecho la medición (hora, día, etc).
<i>Concentration</i>	String	Valor de la concentración del contaminante medido.
<i>UnitOfMeasurement</i>	String	Unidad de medida de la concentración de la muestra.
<i>DateTimeBegin</i>	Datetime	Hora de inicio de la medición (yyy-mm-dd hh:mm:ss)
<i>DateTimeEnd</i>	Datetime	Hora de finalización de la medición (yyy-mm-dd hh:mm:ss)
<i>Validity</i>	Integer	Indicador de validez de la medición.
<i>Verification</i>	Integer	Indicador de verificación de la medición.

Tabla que describe los campos que componen la estructura de los archivos CSV descargados desde la base de datos de la EEA referente a la contaminación del aire medida por las Estaciones de España en el año 2020.

### 5.3. Fuentes de datos sobre vuelos

Como en el apartado anterior, esta sección recoge algunos de los repositorios de datos relacionados con vuelos comerciales que han sido contemplados para la realización del estudio.

#### 5.3.1. OpenFlights

OpenFlights<sup>31</sup> es un proyecto *open-source* que permite buscar y filtrar vuelos de todo el mundo además de calcular estadísticas de forma automática y compartir los resultados. Se trata de un proyecto con una gran cantidad de *datasets*, los más importantes recogen datos históricos sobre aeropuertos, aerolíneas, rutas, aviones y países.

<sup>31</sup><https://openflights.org/data.html>



Figura 8. Visualización del *dataset* de rutas del proyecto Openflights.

Sin embargo, el principal inconveniente de este proyecto es que no dispone de datos actualizados. Las últimas actualizaciones para algunos de estos conjuntos de datos son del año 2018.

### 5.3.2. Flightradar24

Flightradar24<sup>32</sup> es un servicio de seguimiento global sobre vuelos que proporciona datos en tiempo real sobre miles de vuelos alrededor de todo el mundo cada día. El servicio que ofrece está disponible no sólo desde su página web, sino desde aplicaciones móviles para Android e iOS.

El proyecto nace en 2006 en Suecia como una simple red de receptores ADS-B en el norte de Europa. En 2009 la red se hace pública y se abre a que cualquier persona pueda añadir su receptor ADS-B a la red para subir los datos recogidos y aumentar la cobertura de ésta.

Flightradar24 cuenta con la red de receptores ADS-B más extensa del mundo con más de 20.000 conectados y es capaz de hacer un seguimiento de más de 180.000 vuelos al día. Una de sus características principales es la capacidad de filtrar y hacer búsquedas directamente sobre su base de datos.

Además de este servicio gratuito, Flightradar24 cuenta con diversos planes de suscripción de pago en los que ofrece datos históricos, conjuntos de datos más detallados, y un sistema de alertas entre otras características. Por otro lado, también cuenta con un servicio de venta de datos para negocios en el que ofrecen servicios personalizados, independientes de los planes de suscripción mencionados y con precio variable en función del volumen de datos obtenidos.

---

<sup>32</sup><https://www.flightradar24.com/>



Figura 9. Visualización los datos de Flightradar24 sobre Europa.

### 5.3.3. The OpenSky Network

Por último, encontramos OpenSky Network<sup>33</sup> como una asociación sin ánimo de lucro con base en Suiza y con el objetivo de mejorar la seguridad, fiabilidad y eficiencia del uso del espacio aéreo proporcionando acceso público a datos sobre el tráfico aéreo global. Esta asociación consiste en una red de sensores conectados entre sí e instalados por usuarios de todo el mundo, industrias especializadas y asociaciones académicas y gubernamentales.

Todos los datos extraídos de esta red son almacenados en bases de datos usadas por investigadores de diversas áreas para analizar y mejorar las tecnologías y procesos relacionados con el tráfico aéreo.

OpenSky Network nació en 2012 como un proyecto entre universidades suizas, alemanas y británicas hasta que en 2015 se fundó la asociación garantizando un desarrollo continuo de la red hasta convertirse en una de las más extensas con receptores a lo largo de todo el mundo.

En cuanto al acceso a los datos que ofrece, OpenSky cuenta con una API de libre acceso, además de una base de datos históricos accesible desde un servidor Impala<sup>34</sup>. Cabe destacar que para acceder a este servidor, antes es necesario llenar un formulario en el que se explica el motivo y el uso que se dará a estos datos.

---

<sup>33</sup><https://opensky-network.org/>

<sup>34</sup><https://impala.apache.org/>

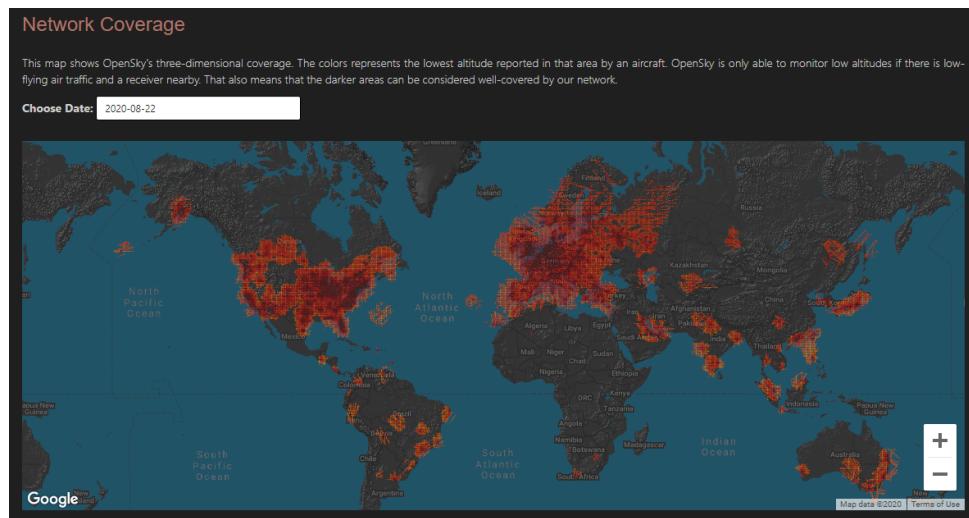


Figura 10. Visualización del tráfico aéreo a fecha 20-01-2010 por OpenSky Network.

#### 5.4. Obtención de los datos sobre el tráfico aéreo

La fuente de datos elegida para la parte de tráfico aéreo ha sido finalmente OpenSky Network. El hecho de que disponga de una base de datos históricos tan detallada y accesible hace de esta opción la más indicada para la obtención de los datos necesarios para completar el estudio.

Para acceder a la base de datos, tal y como se ha mencionado en el apartado anterior, es necesario llenar un formulario y obtener permisos de la propia web. Una vez se obtienen los permisos, simplemente hay que conectarse por medio de SSH a la dirección del servidor Impala y acceder a los datos deseados. En próximos apartados se detallará el proceso de filtrado y descarga de los datos.

La base de datos cuenta con 12 tablas diferentes, sin embargo, la información que se precisa para este estudio se encuentra en la tabla *state\_vectors\_data4*.

```
[hadoop-2:21000] > show tables;
Query: show tables
+-----+
| name           |
+-----+
| acas_data4    |
| allcall_replies_data4 |
| identification_data4 |
| operational_status_data4 |
| position_data4 |
| rollcall_replies_data4 |
| sensor_visibility |
| sensor_visibility_data3 |
| state_vectors  |
| state_vectors_data3 |
| state_vectors_data4 |
| velocity_data4  |
+-----+
Fetched 12 row(s) in 0.04s
```

Figura 11. Tablas disponibles en la base de datos de OpenSky Network.

```
[hadoop-2:21000] > describe state_vectors_data4;
Query: describe state_vectors_data4
+-----+-----+-----+
| name | type | comment |
+-----+-----+-----+
| time | int | Inferred from Parquet file. |
| icao24 | string | Inferred from Parquet file. |
| lat | double | Inferred from Parquet file. |
| lon | double | Inferred from Parquet file. |
| velocity | double | Inferred from Parquet file. |
| heading | double | Inferred from Parquet file. |
| vertrate | double | Inferred from Parquet file. |
| callsign | string | Inferred from Parquet file. |
| onground | boolean | Inferred from Parquet file. |
| alert | boolean | Inferred from Parquet file. |
| spi | boolean | Inferred from Parquet file. |
| squawk | string | Inferred from Parquet file. |
| baroaltitude | double | Inferred from Parquet file. |
| geoaltitude | double | Inferred from Parquet file. |
| lastposupdate | double | Inferred from Parquet file. |
| lastcontact | double | Inferred from Parquet file. |
| serials | array<int> | Inferred from Parquet file. |
| hour | int | Inferred from Parquet file. |
+-----+-----+-----+
Fetched 18 row(s) in 0.05s
```

Figura 12. Estructura de la tabla *state\_vectors\_data4*.

A continuación se detalla la estructura que sigue esta base de datos:

Tabla 2  
*Estructura del dataset sobre el tráfico aéreo*

Campo	Tipo	Descripción
<i>time</i>	Integer	<i>Timestamp</i> en formato UNIX <sup>35</sup> de estado de vector.
<i>icao24</i>	String	Identificador de 24 bits del transpondedor de la aeronave.
<i>lat</i>	Double	Coordenada de latitud expresada en decimales WGS84 <sup>36</sup> .
<i>lon</i>	Double	Coordenada de latitud expresada en decimales WGS84.
<i>velocity</i>	Double	Velocidad de la aeronave en metros por segundo.
<i>heading</i>	Double	Dirección de movimiento de la aeronave.
<i>vertrate</i>	Double	Velocidad vertical de la aeronave en metros por segundo.
<i>callsign</i>	String	Identificador del vuelo emitido desde la aeronave.
<i>onground</i>	Boolean	Indica si la aeronave está en la tierra ( <i>true</i> ), o en el aire ( <i>false</i> ).
<i>alert</i>	Boolean	Indicador especial utilizado por los controladores de tráfico aéreo.
<i>spi</i>	Boolean	Indicador especial utilizado por los controladores de tráfico aéreo.
<i>squawk</i>	String	Identificador de 4 dígitos utilizado por los controladores de tráfico aéreo para emergencias.
<i>baroaltitude</i>	Double	Altitud de la aeronave medida por un barómetro.
<i>geoaltitude</i>	Double	Altitud de la aeronave medida por un sensor GNSS <sup>37</sup> .
<i>lastposupdate</i>	Double	<i>Timestamp</i> que indica la última actualización de la posición de la aeronave
<i>lastcontact</i>	Double	<i>Timestamp</i> que indica la última actualización sobre la aeronave.
<i>hour</i>	Integer	<i>Timestamp</i> que marca el inicio de la hora a la que pertenece.

Tabla que describe los campos que componen la tabla *state\_vectors\_data4* de la base de datos de OpenSky Network.

<sup>35</sup>[https://es.wikipedia.org/wiki/Tiempo\\_Unix](https://es.wikipedia.org/wiki/Tiempo_Unix)

<sup>36</sup><https://gisgeography.com/wgs84-world-geodetic-system/>

<sup>37</sup><https://en.wikipedia.org/wiki/GNSS>

## 6. Obtención de los datos

El proceso de extracción de los datos es diferente para cada uno de los campos, a continuación se describe el proceso seguido.

### 6.1. Extracción de datos sobre contaminación del aire

Cómo ya se ha mencionado en el apartado 5.2, el conjunto de datos sobre la contaminación del aire se ha descargado desde la *European Environment Agency*.

Para descargar los datos, la web permite hacer una petición hacia una URL con 9 campos. Cada uno de estos campos actúa como un filtro y el resultado de ejecutar esta URL es una lista de direcciones HTTP que apuntan hacia ficheros en formatos CSV accesibles de forma pública.

Los posibles campos de la URL se muestran a continuación:

- **CountryCode**: Código de país en formato ISO
- **CityName**: Nombre de la ciudad
- **Pollutant**: Nombre del contaminante
- **Year\_from**: Año de inicio
- **Year\_to**: Año de finalización
- **Station**: Identificador de la estación
- **Source**: Fuente de datos
- **Output**: Formato de salida del resultado (texto o HTML)

Cabe destacar, que no es necesario utilizar todos los campos a la hora de crear la URL, es posible dejar vacío un campo si no se desea aplicar un filtro. Por lo tanto, la URL utilizada en este caso sería la siguiente:

`https://fme.discomap.eea.europa.eu/fmedatastreaming/AirQualityDownload/AQData_Extract.fmw?  
CountryCode=ES&CityName=&Pollutant=&Year_from=2020&Year_to=2020&Station=  
&Source=E2a&Output=TEXT`

```

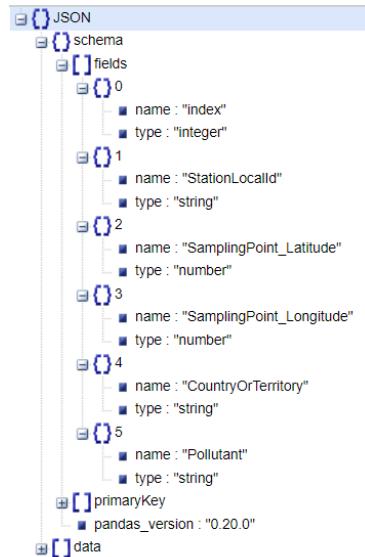
CountryCode=ES      // ES - Datos cuyo país se corresponde con España
&CityName=          // Vacío - Se aplica para todas las ciudades de España
&Pollutant=         // Vacío - Se aplica para todos los tipos de contaminantes
&Year_from=2020    // 2020 - Se aplica únicamente para el año 2020
&Year_to=2020      // 2020 - Se aplica únicamente para el año 2020
&Station=          // Vacío - Se aplica para todas las estaciones de España
&Source=E2a         // E2a - Se corresponde al conjunto de datos actualizados
&Output=TEXT        // TEXT - Modo de extracción de los conjuntos de datos
  
```

*Figura 13.* Descripción de los campos de la URL utilizados para filtrar los datos sobre la contaminación del aire.

En el momento de la realización de este proyecto, la URL devuelve un listado de 3.073 direcciones de archivos CSV, las cuales comparten la misma estructura vista en la tabla 1. Una vez descargados son unificados en un único fichero de 2,8 Gigabytes y almacenados en un *bucket* de Amazon S3 en la siguiente dirección:

[https://ui1-tfm-data.s3.amazonaws.com/air-data/spain\\_airQuality.csv](https://ui1-tfm-data.s3.amazonaws.com/air-data/spain_airQuality.csv)

Si nos fijamos en los campos que conforman la estructura de este conjunto de datos, vemos que ninguno de ellos hace referencia a la localización concreta de las mediciones. Debido a esto, se ha descargado un conjunto de datos sobre las estaciones de medición en el que entre otros campos, se encuentran el de las coordenadas de latitud y longitud.



*Figura 14.* Estructura JSON de los datos de las estaciones de calidad del aire.

Este archivo también se ha almacenado en Amazon S3 para su posterior uso. Se puede encontrar en el siguiente enlace:

<https://ui1-tfm-data.s3.amazonaws.com/air-data/stations.json>

## 6.2. Extracción de datos sobre tráfico aéreo

En la sección 5.4 hemos visto cómo la fuente de datos sobre el tráfico aéreo que hemos seleccionado es OpenSky Network. Tal y como se ha mencionado, para acceder a su base de datos históricos es necesario registrarse y obtener permisos para acceder al servidor Impala desde el cual hacer las peticiones SQL.

Una vez obtenido el acceso al servidor vía SSH, se han hecho pruebas para determinar las mejor forma de descargar la información y se han obtenido las siguientes conclusiones:

- Las constantes actualizaciones que reporta cada aeronave hacen que el número de entradas sea muy alto, por lo que se deberá filtrar el periodo de tiempo del cual descargar los datos. En este caso, se extraerán las entradas recibidas cada 30 segundos.
- Los campos *lat* y *lon* permiten filtrar las entradas por las coordenadas en las que se encuentra el avión, por lo tanto, se deberá limitar un área del cual extraer los datos. Para este proyecto, nos limitaremos a extraer los datos sobre los vuelos que pasan sobre la Península Ibérica.
- Se ha de decidir un rango de fechas para el cual extraer las entradas de la base de datos. En nuestro caso, se ha decidido utilizar los datos sobre el mes de enero de 2020.
- Por último, y tras aplicar las limitaciones mencionadas, el rendimiento del filtrado y extracción de los datos es realmente pobre. Sin embargo, en la documentación<sup>38</sup> de OpenSky se encuentra una manera de acelerar el proceso. Consiste en utilizar el campo *hour* para indicar al servidor «dónde buscar».

Las marcas de tiempo vienen dado en formato UNIX, por lo tanto la forma de calcular la entrada por la que filtrar la búsqueda y aumentar el rendimiento es de la siguiente manera:

1. Convertir la fecha y hora deseada a formato UNIX:

La fecha 3 de diciembre de 2016 10:15AM en formato UNIX es 1480760100 (para hacer esta conversión se ha utilizado la herramienta web unixtimestamp<sup>39</sup>).

2. Calculamos la «hora» en la que comienza este intervalo usando la marca de tiempo que hemos calculado:

$$1480760100 - (1480760100 \% 3600) = 1480759200$$

3. Utilizar el campo *hour* con la marca de tiempo que hemos calculado, para filtrar las entradas que lo contienen y reducir el tiempo de búsqueda considerablemente.

Para hacer la conexión al servidor y almacenar los datos descargados, se ha decidido utilizar una máquina virtual de Amazon EC2 bajo el sistema operativo de Linux y la distribución Ubuntu Server en su versión 18.04 LTS. En cuanto al hardware de la máquina virtual, se ha elegido una instancia de tipo *t2.micro* ya que no se precisa de una gran capacidad de computación además de ser de uso gratuito.

<sup>38</sup><https://opensky-network.org/data/impala>

<sup>39</sup><https://www.unixtimestamp.com/>

▼ AMI Details

	Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0bcc094591f354be2
Free tier eligible	Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical ( <a href="http://www.ubuntu.com/cloud/services">http://www.ubuntu.com/cloud/services</a> ).
Root Device Type: ebs	Virtualization type: hvm

▼ Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

▼ Security Groups

Security Group ID	Name	Description
sg-0880c3522c1ad4f99	launch-wizard-1	launch-wizard-1 created 2020-08-26T14:15:46.439+02:00

All selected security groups inbound rules

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ
SSH	TCP	22	0.0.0.0/0	

► Instance Details

▼ Storage

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput (MB/s) ⓘ	Delete on Termination ⓘ	Encrypted ⓘ
Root	/dev/sda1	snap-091c9b89d2082ce92	8	gp2	100 / 3000	N/A	Yes	Not Encrypted

Figura 15. Resumen de la configuración de la instancia EC2 utilizada.

Como podemos ver en la figura 15, la máquina cuenta con un procesador mononúcleo y 1 Gigabyte de memoria RAM junto con un disco duro de estado sólido de 8 Gigabytes de almacenamiento. La conexión se ha realizado mediante el protocolo SSH usando el programa PuTTY.

Con el objetivo de acortar las sesiones de descarga de los datos, se ha dividido el dataset en 30 partes (días 1 al 30 de enero). Por lo tanto se ha de generar una *query* diferente para cada una de las búsquedas.

A continuación se muestra la *query* utilizada para la descarga de los datos pertenecientes al día 1 de enero de 2020 junto con la explicación de cada uno de sus campos:

```
SELECT * FROM state_vectors_data4
WHERE lat > 35.512 AND lat < 44.512
AND lon > -10.415 AND lon < 5.054
AND time>=1577833200 AND time<=1577919600
AND hour>=1577833200 AND hour<=1577919600
AND time%30=0
\\ Indica la tabla sobre la que se aplican los filtros
\\ Indica las latitudes entre las que debe encontrarse el avión
\\ Indica las longitudes entre las que debe encontrarse el avión
\\ Indica el rango de fechas en formato UNIX
\\ Indica la hora a la que pertenecen las entradas para mejorar el rendimiento
\\ Indica que únicamente deben seleccionarse las entradas cada 30 segundos
```

Figura 16. Explicación de la *query* utilizada para extraer información desde la base de datos de OpenSky.

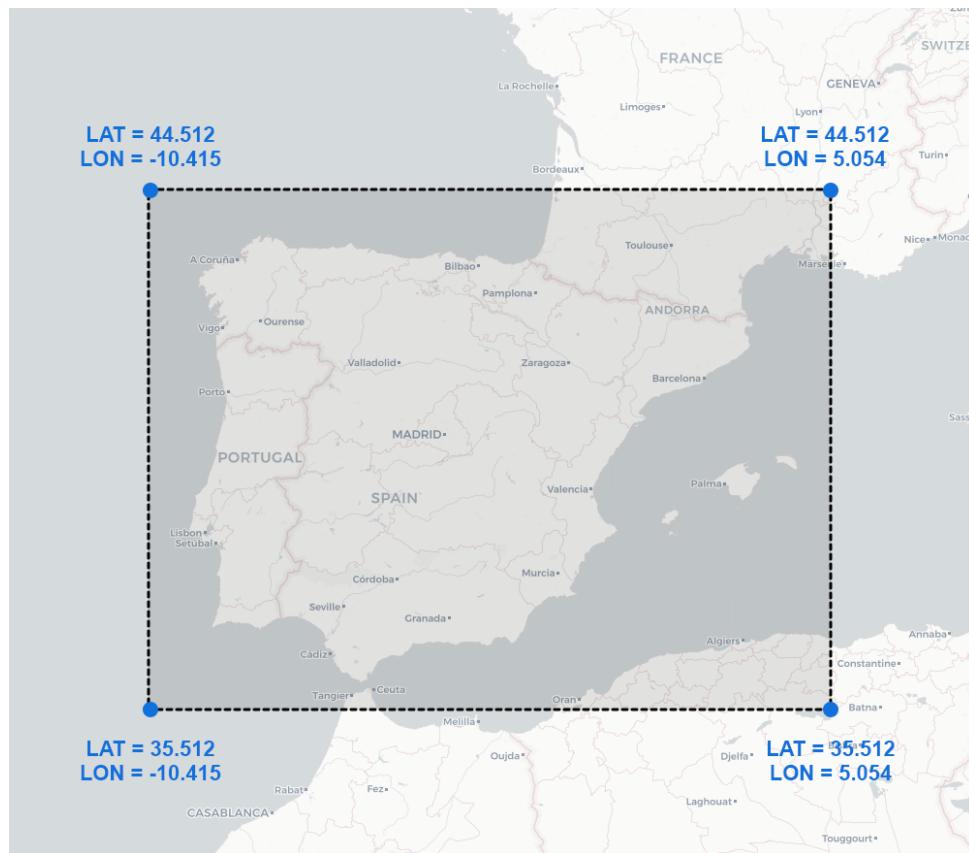


Figura 17. Mapa que muestra el área sobre el que se extraen los datos del tráfico aéreo.

Tal y como se puede ver en la figura 17, el área para el cual se han descargado los datos sobre el tráfico aéreo no coincide únicamente con la Península Ibérica, por lo tanto hay grandes zonas para las que no existirán datos sobre la contaminación del aire, por ejemplo Portugal y parte Francia.

Por último, es necesario añadir que el almacenamiento de los datos extraídos con este método se ha realizado a través de Amazon S3 desde la propia máquina virtual con el comando `aws s3 cp` indicando la ruta del fichero a almacenar y la dirección remota del *bucket S3* en el que se desea guardar.

El resultado de la extracción de datos son 30 archivos en formato CSV con un tamaño de aproximadamente 100 Megabytes cada uno y con la estructura detallada en la tabla 2. Se puede encontrar en el siguiente enlace, cambiando el número final por el del día deseado:

[https://ui1-tfm-data.s3.amazonaws.com/opensky-data/enero\\_1.csv](https://ui1-tfm-data.s3.amazonaws.com/opensky-data/enero_1.csv)

## 7. Análisis y procesado de los datos

### 7.1. Infraestructura empleada

Tal y como se ha mencionado en el apartado 3.2.2, la mayoría del procesado y análisis de los datos ha sido ejecutado sobre la plataforma de Amazon Web Services. Ya hemos descrito los usos de Amazon EC2 para la puesta en marcha de una máquina virtual con la que acceder a la base de datos de OpenSky, y Amazon S3 para el almacenamiento de los conjuntos de datos.

En esta sección, seguimos en la plataforma de AWS y utilizaremos el servicio Amazon EMR como entorno sobre el que ejecutar el código necesario para analizar y procesar los datos.

#### 7.1.1. Configuración

Antes de mostrar cómo ha sido configurado este entorno, es necesario entender el funcionamiento de Amazon EMR:

Se trata de una herramienta pensada para el procesado y análisis de grandes cantidades de datos usando herramientas como Spark, Hive, HBase, junto con otros servicios de AWS como Amazon S3 para el almacenamiento y Amazon EC2 para la capacidad computacional. El uso conjunto de estas herramientas hace posible en análisis y procesado de datos a gran escala más barata y eficiente que los sistemas tradicionales basados en clústers *on-premise*.

Uno de los principales casos de uso de Amazon EMR es el proceso de extracción, transformación y carga de datos (ETL), que es precisamente en lo que se basa esta parte del proyecto.

En cuanto al funcionamiento de este servicio, hay que destacar que cada clúster está formado por múltiples nodos o instancias directamente gestionadas por EMR y cuyas funciones varían dependiendo del tipo de nodo. A continuación se describen los distintos tipos de nodo junto sus funciones:

- **Nodo *master*:** Se trata de el nodo principal del clúster y es el encargado de ejecutar los principales componentes de las aplicaciones distribuidas. Entre sus funciones, también se encuentra la monitorización de tareas, así como el funcionamiento general de todo el clúster. Cabe destacar que un clúster EMR puede estar únicamente formado por un nodo *master*.
- **Nodo *core*:** Estos nodos están gestionados por el nodo *master* y sus principales tareas se centran en la coordinación del almacenamiento de datos como parte del sistema de archivos distribuidos de Hadoop (HDFS) y de la ejecución en paralelo de las diferentes tareas necesarias para el correcto funcionamiento de las aplicaciones instaladas.

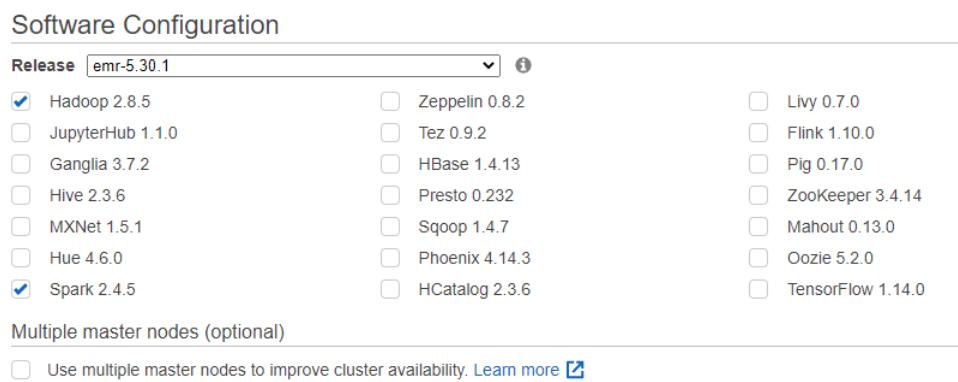
A diferencia del nodo *master*, en un mismo clúster EMR puede haber más de un nodo *core*.

- Nodo *task*: Este tipo de nodo es opcional. Se pueden usar como capacidad de procesamiento extra para la realización de ciertas tareas, aunque cuentan con ciertas limitaciones como la incapacidad de almacenar datos en HDFS.

En cuanto a la configuración usada para este proyecto, a continuación se describe junto con imágenes de su puesta en marcha:

El primer apartado de la configuración del clúster se basa en la selección del *software* a instalar. En nuestro caso, seleccionamos Hadoop para hacer uso del sistema de almacenamiento distribuido entre los diferentes nodos, y Spark para utilizar PySpark como entorno de ejecución.

Asimismo, se puede ver una opción en la que se especifica que únicamente habrá un nodo de tipo *master* en el clúster.



**Software Configuration**

Release emr-5.30.1

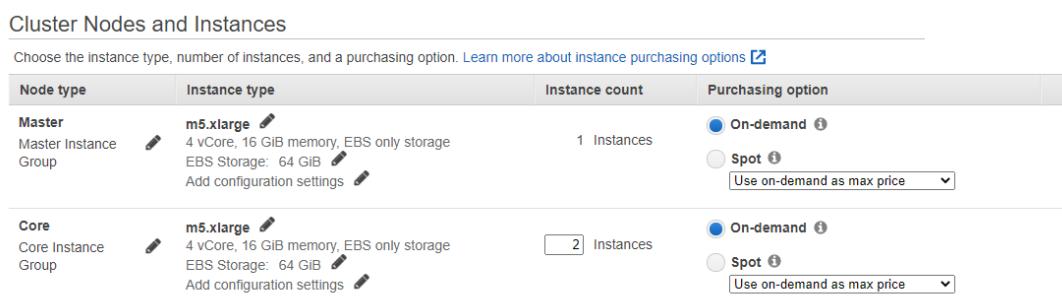
<input checked="" type="checkbox"/> Hadoop 2.8.5	<input type="checkbox"/> Zeppelin 0.8.2	<input type="checkbox"/> Livy 0.7.0
<input type="checkbox"/> JupyterHub 1.1.0	<input type="checkbox"/> Tez 0.9.2	<input type="checkbox"/> Flink 1.10.0
<input type="checkbox"/> Ganglia 3.7.2	<input type="checkbox"/> HBase 1.4.13	<input type="checkbox"/> Pig 0.17.0
<input type="checkbox"/> Hive 2.3.6	<input type="checkbox"/> Presto 0.232	<input type="checkbox"/> ZooKeeper 3.4.14
<input type="checkbox"/> MXNet 1.5.1	<input type="checkbox"/> Sqoop 1.4.7	<input type="checkbox"/> Mahout 0.13.0
<input type="checkbox"/> Hue 4.6.0	<input type="checkbox"/> Phoenix 4.14.3	<input type="checkbox"/> Oozie 5.2.0
<input checked="" type="checkbox"/> Spark 2.4.5	<input type="checkbox"/> HCatalog 2.3.6	<input type="checkbox"/> TensorFlow 1.14.0

Multiple master nodes (optional)

Use multiple master nodes to improve cluster availability. [Learn more](#)

Figura 18. Apartado de configuración de *software* del clúster EMR utilizado.

El siguiente apartado en la configuración se centra en el *hardware* que utilizarán los nodos de los que se compone el clúster utilizado. Tal y como se ha mencionado, cada clúster puede tener un número variable de nodos. En este caso, se ha optado por la utilización de dos nodos *Core* a parte del nodo *Master* siendo los 3 de tipo m5.xlarge, que como podemos ver en la imagen siguiente, están compuestos por procesadores de 4 núcleos, 16 Gigabytes de memoria RAM y 64 Gigabytes de almacenamiento.



**Cluster Nodes and Instances**

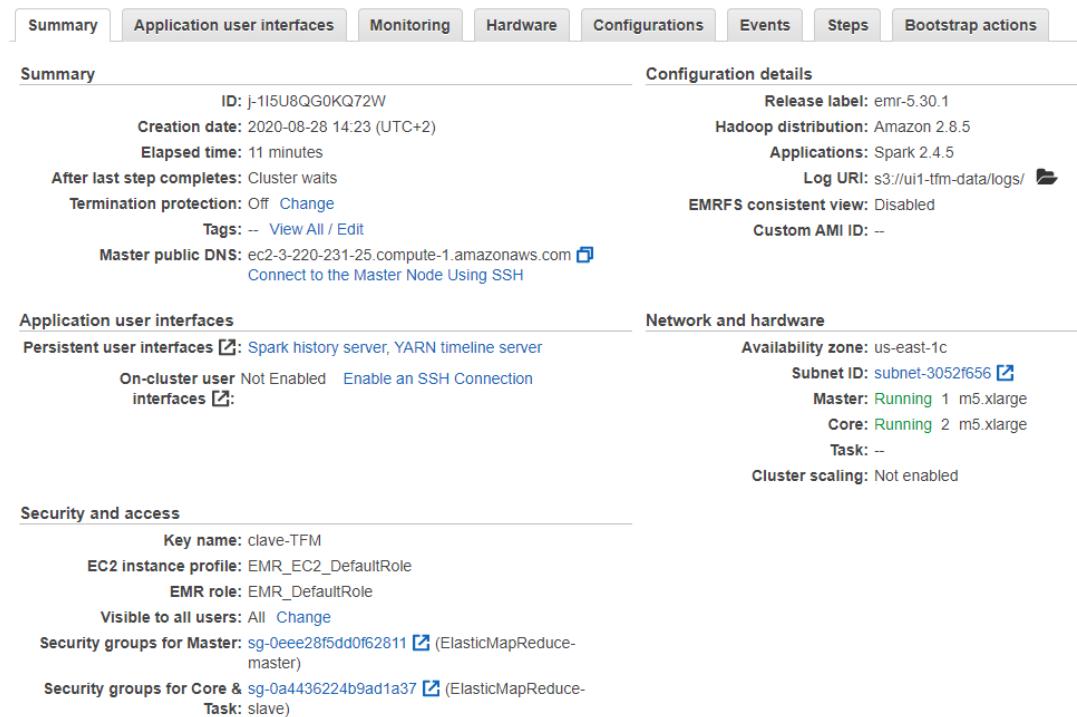
Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Node type	Instance type	Instance count	Purchasing option
Master	m5.xlarge	1 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot <input type="checkbox"/> Use on-demand as max price
Master Instance Group	4 vCore, 16 GiB memory, EBS only storage EBS Storage: 64 GiB Add configuration settings		
Core	m5.xlarge	2 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot <input type="checkbox"/> Use on-demand as max price
Core Instance Group	4 vCore, 16 GiB memory, EBS only storage EBS Storage: 64 GiB Add configuration settings		

Figura 19. Apartado de configuración de *hardware* del clúster EMR utilizado.

A continuación vemos una imagen que muestra el resumen de la configuración del clúster EMR utilizado para la realización del proyecto:

Cluster: TFM-EMR-CLUSTER Waiting Cluster ready after last step completed.



**Summary**

- ID: J-1I5U8QG0KQ72W
- Creation date: 2020-08-28 14:23 (UTC+2)
- Elapsed time: 11 minutes
- After last step completes: Cluster waits
- Termination protection: Off [Change](#)
- Tags: -- [View All / Edit](#)
- Master public DNS: ec2-3-220-231-25.compute-1.amazonaws.com [Connect to the Master Node Using SSH](#)

**Application user interfaces**

- Persistent user interfaces: [Spark history server, YARN timeline server](#)
- On-cluster user interfaces: Not Enabled [Enable an SSH Connection](#)

**Configuration details**

- Release label: emr-5.30.1
- Hadoop distribution: Amazon 2.8.5
- Applications: Spark 2.4.5
- Log URI: s3://ui1-tfm-data/logs/ [File](#)
- EMRFS consistent view: Disabled
- Custom AMI ID: --

**Network and hardware**

- Availability zone: us-east-1c
- Subnet ID: subnet-3052f656 [Edit](#)
- Master: [Running](#) 1 m5.xlarge
- Core: [Running](#) 2 m5.xlarge
- Task: --
- Cluster scaling: Not enabled

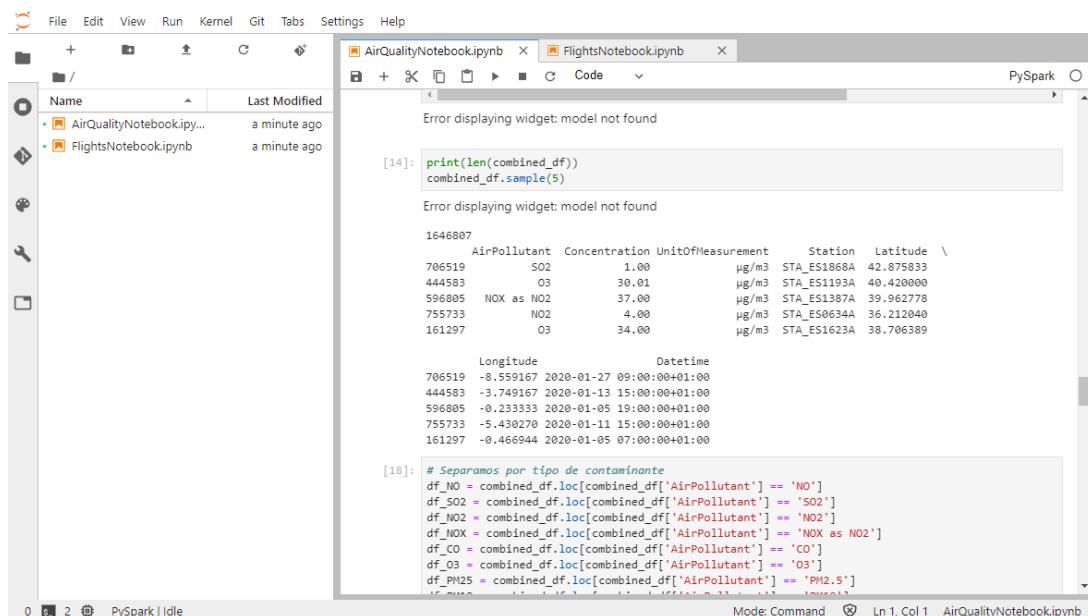
**Security and access**

- Key name: clave-TFM
- EC2 instance profile: EMR\_EC2\_DefaultRole
- EMR role: EMR\_DefaultRole
- Visible to all users: All [Change](#)
- Security groups for Master: sg-0eee28f5dd0f62811 [Edit](#) (ElasticMapReduce-master)
- Security groups for Core & Task: sg-0a4436224b9ad1a37 [Edit](#) (ElasticMapReduce-Task: slave)

Figura 20. Resumen de la configuración del clúster EMR.

Por último, para ejecutar el código Python con el que se analizan y procesan los datos extraídos, utilizaremos la interfaz web que ofrece JupyterLab<sup>40</sup> para la ejecución de *notebooks*.

Dispondremos de dos *notebooks* independientes, en el primero se tratará el conjunto de datos sobre la contaminación del aire, mientras que en el segundo se tratará el *dataset* del tráfico aéreo.



```

[14]: print(len(combined_df))
combined_df.sample(5)

1646807
   AirPollutant  Concentration UnitOfMeasurement      Station  Latitude \
706519          SO2       1.00        ug/m3 STA_ES1868A  42.875833
444583           O3       30.01        ug/m3 STA_ES1193A 40.420000
596805         NOX as NO2      37.00        ug/m3 STA_ES1387A 39.962778
755733           NO2       4.00        ug/m3 STA_ES0634A 36.212040
161297           O3       34.00        ug/m3 STA_ES1623A 38.706389

   Longitude          Datetime
706519 -8.559167 2020-01-27 09:00:00+01:00
444583 -3.749167 2020-01-13 15:00:00+01:00
596805 -0.233333 2020-01-05 19:00:00+01:00
755733 -5.430270 2020-01-11 15:00:00+01:00
161297 -0.466944 2020-01-05 07:00:00+01:00

[18]: # Separamos por tipo de contaminante
df_NO = combined_df.loc[combined_df['AirPollutant'] == 'NO']
df_SO2 = combined_df.loc[combined_df['AirPollutant'] == 'SO2']
df_NO2 = combined_df.loc[combined_df['AirPollutant'] == 'NO2']
df_NOX = combined_df.loc[combined_df['AirPollutant'] == 'NOX as NO2']
df_CO = combined_df.loc[combined_df['AirPollutant'] == 'CO']
df_O3 = combined_df.loc[combined_df['AirPollutant'] == 'O3']
df_PM25 = combined_df.loc[combined_df['AirPollutant'] == 'PM2.5']

```

Figura 21. Entorno de JupyterLab con los *notebooks* usados en el proyecto.

<sup>40</sup><https://jupyterlab.readthedocs.io/en/stable/>

## 7.2. Análisis exploratorio

Llega el momento de entender los datos que hemos descargado y decidir que transformaciones llevar a cabo sobre ellos para poder relacionar los distintos campos entre sí. Este apartado se divide en dos secciones, primero trataremos el conjunto de datos sobre la contaminación del aire, y después el de tráfico aéreo.

### 7.2.1. Contaminación del aire

Comenzamos con el conjunto de datos sobre las estaciones de monitorización mencionado en la sección 6.1 y que contiene información acerca de la localización o el contaminante que mide cada una. Se trata de un archivo JSON con 2.452 entradas de 5 campos cada una.

Lo primero que se ha hecho ha sido convertirlo en un objeto *Dataframe* con ayuda de la librería pandas cuyo contenido podemos ver a continuación:

	StationLocalId	SamplingPoint_Latitude	SamplingPoint_Longitude	CountryOrTerritory	Pollutant
0	GIB_Station_GB0050A	36.133317	-5.353175	Gibraltar	o-Xylene (air)
1	GIB_Station_GB0051A	36.112824	-5.350188	Gibraltar	Particulate matter < 10 µm (aerosol)
2	GIB_Station_GB0951A	36.128068	-5.351246	Gibraltar	Nitrogen oxides (air)
3	HR_DOC_TYPE_D_STA_RH0101	45.800339	15.974072	Croatia	Indeno_123cd_pyrene in PM10 (aerosol)
4	HR_DOC_TYPE_D_STA_RH0103	45.764947	16.006469	Croatia	Indeno_123cd_pyrene in PM10 (aerosol)

Figura 22. Contenido del *dataframe* con las estaciones de contaminación del aire.

Sin embargo, para poder analizar estos datos e intentar extraer información de ellos, lo primero que se ha hecho ha sido identificar aquellas entradas que nos son útiles y desechar el resto. Como ya se ha dicho en apartados anteriores, el estudio se realiza sobre el espacio aéreo que se encuentra sobre España. Por lo tanto, se han filtrado aquellas entradas cuyo campo *CountryOrTerritory* se corresponda con *Spain* y después de esto, se han identificado las estaciones que se encuentran en los límites marcados por las latitudes y longitudes expuestas anteriormente.

Una vez identificadas las estaciones correspondientes y dado que este conjunto de datos únicamente es necesario para aportar información sobre la localización de las estaciones, nos deshacemos de los campos *CountryOrTerritory* y *Pollutant*.

Después de renombrar los campos restantes para mayor comodidad, el *dataframe* resultante cuenta con 493 entradas que se corresponden con las estaciones españolas de la Península Ibérica (e Islas Baleares) cuyo contenido vemos a continuación:

	Station	Latitude	Longitude
0	STA_ES0001R	39.54694	-4.35056
1	STA_ES0005R	42.72056	-8.92361
2	STA_ES0006R	39.87528	4.31639
3	STA_ES0007R	37.23722	-3.53417
4	STA_ES0008R	43.43917	-4.85000

Figura 23. Contenido del *dataframe* final con las estaciones de contaminación del aire españolas.

La siguiente tabla muestra la estructura del conjunto de datos final junto con los tipos de datos que contiene y una breve descripción de cada uno:

Tabla 3  
*Estructura del dataset final de las estaciones de medición de la contaminación del aire en España*

Campo	Tipo	Descripción
<i>Station</i>	String	Código identificativo de la estación
<i>Latitude</i>	Float	Coordenada de latitud
<i>Longitude</i>	Float	Coordenada de longitud

Tabla que muestra la estructura del *dataset* final de las estaciones de medición de la contaminación del aire después de filtrar las entradas válidas para el estudio e identificar los campos con importancia para futuros pasos.

Para finalizar con este conjunto de datos, se ha hecho una visualización de cada una de las estaciones sobre un mapa geográfico de la Península Ibérica. Para ello se ha utilizado la librería Folium<sup>41</sup> con la que se ha señalado la localización de las estaciones con un marcador sobre el mapa tal y como se puede ver en la siguiente figura:

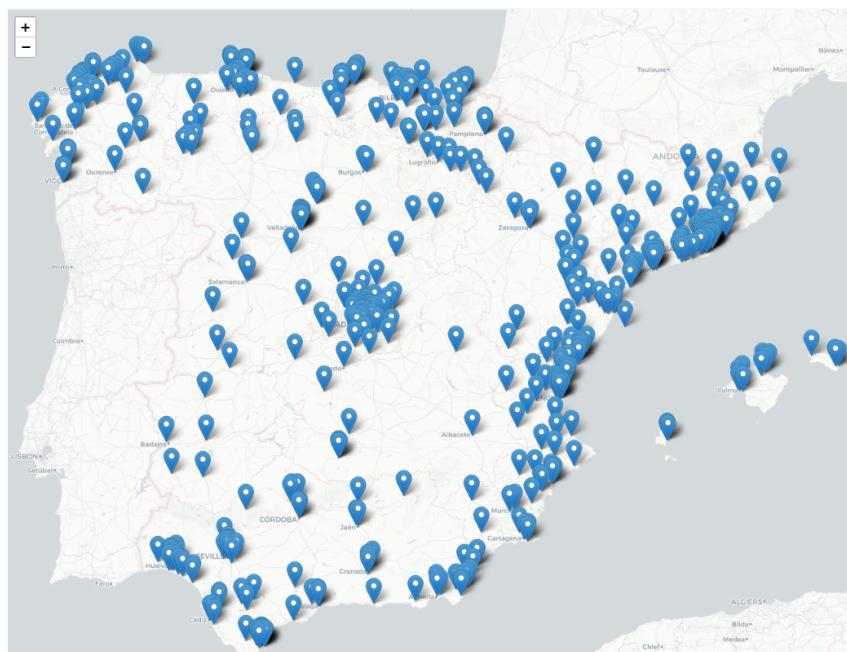


Figura 24. Mapa que muestra la localización de las estaciones seleccionadas para la realización del estudio.

Por otro lado, tenemos el conjunto de datos con las mediciones de los contaminantes. De la misma forma que con el anterior, se ha utilizado pandas para convertirlo en un objeto *Dataframe* que inicialmente cuenta con 12.850.337 entradas y 17 campos diferentes.

<sup>41</sup><https://python-visualization.github.io/folium/>

CountryCode	Namespace	AirqualityNetwork	AirqualityStation	AirqualityStationEoiCode	SamplingPoint	SamplingProcess	Sample	AirPollutant	AirPollutantCode	AveragingTime	Concentration	UnitOfMeasurement	DatetimeBegin	DatetimeEnd	Validity	Verification	
0	ES	ES_BDCA_AQD	NET_ES214A	STA_ES1633A	ES1633A	SP_30030007_7_B	SPP_30030007_7_B_1	SAM_30030007_7_B	NO	http://dd.eionet.europa.eu/vocabulary/aq/pollutants	hour	36.0	µg/m³	2020-01-01 00:00:00+01:00	2020-01-01 01:00:00+01:00	1	3
1	ES	ES_BDCA_AQD	NET_ES214A	STA_ES1633A	ES1633A	SP_30030007_7_B	SPP_30030007_7_B_1	SAM_30030007_7_B	NO	http://dd.eionet.europa.eu/vocabulary/aq/pollutants	hour	68.0	µg/m³	2020-01-01 01:00:00+01:00	2020-01-01 02:00:00+01:00	1	3
2	ES	ES_BDCA_AQD	NET_ES214A	STA_ES1633A	ES1633A	SP_30030007_7_B	SPP_30030007_7_B_1	SAM_30030007_7_B	NO	http://dd.eionet.europa.eu/vocabulary/aq/pollutants	hour	58.0	µg/m³	2020-01-01 01:00:00+01:00	2020-01-01 03:00:00+01:00	1	3
3	ES	ES_BDCA_AQD	NET_ES214A	STA_ES1633A	ES1633A	SP_30030007_7_B	SPP_30030007_7_B_1	SAM_30030007_7_B	NO	http://dd.eionet.europa.eu/vocabulary/aq/pollutants	hour	47.0	µg/m³	2020-01-01 03:00:00+01:00	2020-01-01 04:00:00+01:00	1	3
4	ES	ES_BDCA_AQD	NET_ES214A	STA_ES1633A	ES1633A	SP_30030007_7_B	SPP_30030007_7_B_1	SAM_30030007_7_B	NO	http://dd.eionet.europa.eu/vocabulary/aq/pollutants	hour	42.0	µg/m³	2020-01-01 04:00:00+01:00	2020-01-01 05:00:00+01:00	1	3

Figura 25. Contenido del *dataframe* con mediciones sobre la contaminación del aire.

Lo primero que encontramos al analizar estos datos es que contiene una gran cantidad de valores nulos en la columna *Concentration* que se corresponde con el valor medido. Por lo tanto, empezamos por deshacernos de dichas entradas completas.

Debido a la naturaleza del estudio que se pretende realizar, hay campos que no aportan datos relevantes, por los que quedan desechados para los siguientes pasos. Estos campos son los siguientes: *CountryCode*, *Namespace*, *AirQualityNetwork*, *AirQualityStationEoiCode*, *Sampling point*, *SamplingProcess*, *Sample*, *AirPollutantCode*, *AveragingTime*, *DateEnd*, *Validity* y *Verification*.

Por último, se han filtrado las entradas en función de la fecha para quedarnos con aquellas que se corresponden con mediciones hechas durante el mes de enero.

Después de estas modificaciones, el *dataframe* resultante cuenta con 1.956.447 entradas y 5 campos, los cuales podemos ver a continuación:

	Station	AirPollutant	Concentration	UnitOfMeasurement	Datetime
0	STA_ES1633A	NO	36.0	µg/m³	2020-01-01 00:00:00+01:00
1	STA_ES1633A	NO	68.0	µg/m³	2020-01-01 01:00:00+01:00
2	STA_ES1633A	NO	58.0	µg/m³	2020-01-01 02:00:00+01:00
3	STA_ES1633A	NO	47.0	µg/m³	2020-01-01 03:00:00+01:00
4	STA_ES1633A	NO	42.0	µg/m³	2020-01-01 04:00:00+01:00

Figura 26. Contenido del *dataframe* con mediciones sobre la contaminación del aire después del filtrado.

Una vez tenemos los dos *dataframes* con las entradas y campos deseados, se han fusionado tomando el campo *Station* como punto en común con el objetivo de dotar a cada una de las entradas con las coordenadas de latitud y longitud de su correspondiente estación. El *dataframe* resultante tiene la siguiente estructura:

	AirPollutant	Concentration	UnitOfMeasurement	Station	Latitude	Longitude	Datetime
0	NO	36.0	µg/m³	STA_ES1633A	37.993611	-1.144722	2020-01-01 00:00:00+01:00
1	NO	68.0	µg/m³	STA_ES1633A	37.993611	-1.144722	2020-01-01 01:00:00+01:00
2	NO	58.0	µg/m³	STA_ES1633A	37.993611	-1.144722	2020-01-01 02:00:00+01:00
3	NO	47.0	µg/m³	STA_ES1633A	37.993611	-1.144722	2020-01-01 03:00:00+01:00
4	NO	42.0	µg/m³	STA_ES1633A	37.993611	-1.144722	2020-01-01 04:00:00+01:00

Figura 27. Contenido del *dataframe* sobre la contaminación del aire e información de su posición.

La siguiente tabla muestra la estructura del *dataset* final después de la fusión con el anterior conjunto de datos:

Tabla 4  
*Estructura del dataset final sobre la contaminación del aire*

Campo	Tipo	Descripción
<i>AirPollutant</i>	String	Nombre identificativo del contaminante medido
<i>Concentration</i>	Float	Concentración medida para este tipo de contaminante
<i>UnitOfMeasurement</i>	String	Unidad de medida para este tipo de contaminante
<i>Station</i>	String	Código identificativo de la estación
<i>Latitude</i>	Float	Coordenada de latitud
<i>Longitude</i>	Float	Coordenada de longitud
<i>Datetime</i>	Datetime	Fecha y hora de la medición

Tabla que muestra la estructura del *dataset* final sobre la contaminación del aire y la localización de las estaciones de medición para cada una de las entradas después de filtrar las entradas válidas para el estudio e identificar los campos con importancia para futuros pasos.

Una vez que contamos con el conjunto de datos limpios, se ha comenzado por analizar la distribución de los datos correspondientes al tipo de contaminante medido. En la figura 28 podemos observar como los contaminantes que cuentan con más mediciones son los óxidos de nitrógeno (NO, NO2 y NOX as NO2) junto con el Ozono (O3) y el dióxido de azufre (SO2), y que los contaminantes con menos entradas en el *dataset* son el Benceno (C6H6) y las partículas finas (PM2.5).

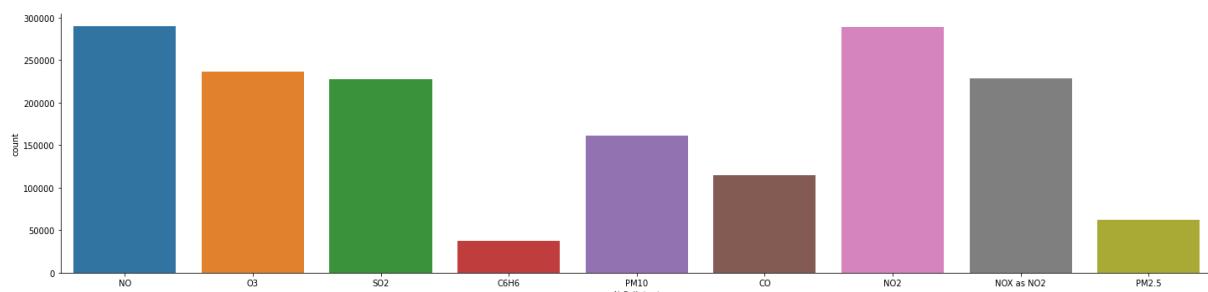


Figura 28. Gráfico de barras que muestra el número de entradas para cada tipo de contaminante.

Por otro lado, con el fin de visualizar la distribución de los valores medido para cada contaminante, se han generado histogramas para cada uno de ellos (ver figura 29).

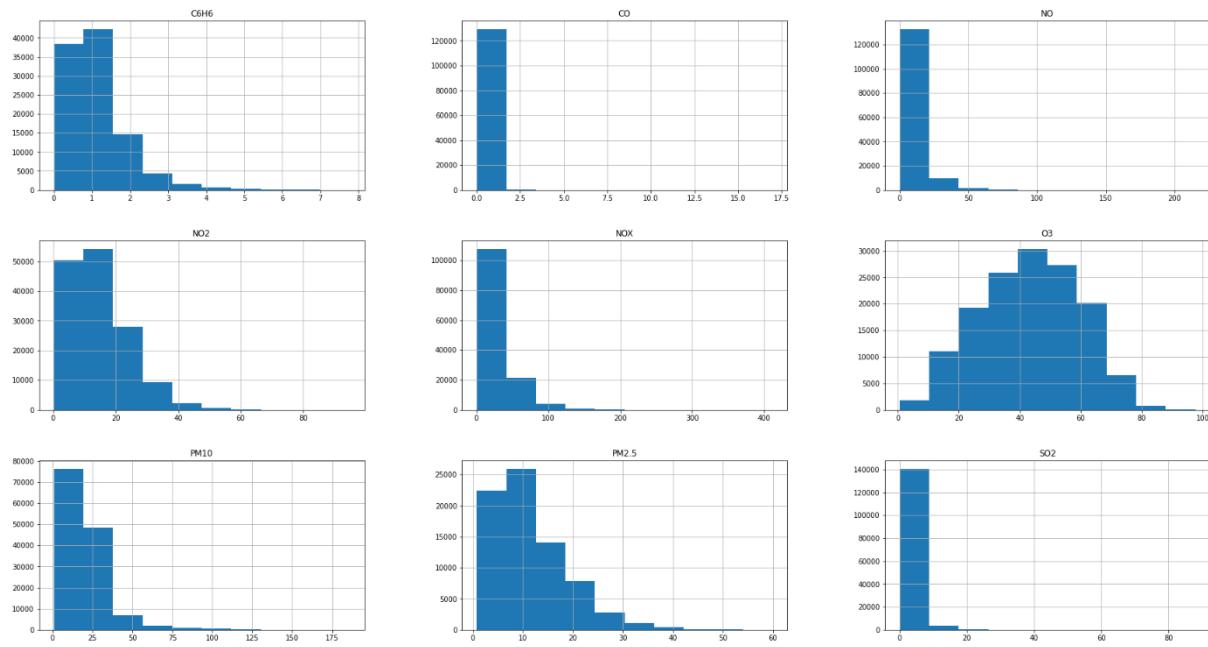


Figura 29. Conjunto de histogramas que muestran la distribución de los valores medidos para cada uno de los contaminantes.

La figura 29 muestra la distribución de los valores medidos para todos los contaminantes. Lo primero que salta a la vista es que únicamente para el contaminante O<sub>3</sub> (Ozono) es reconocible una distribución normal de sus datos, mientras que el resto muestra una gran asimetría hacia la izquierda. Esto hace pensar que puedan existir una cantidad notable de *outliers* y entradas nulas. Sin embargo, es necesario destacar que se tratan de datos brutos que apenas han sido procesados y que, por lo tanto, en apartados siguientes se podrá ver cómo estos mismos histogramas cambian a medida que se procesan los datos.

Por otro lado, en las figuras 30 y 31 encontramos una representación gráfica de dos de los contaminantes observados. Teniendo en cuenta el rango de valores que puede tomar la concentración de los contaminantes, los marcadores situados en función de la latitud y longitud toman un color más amarillento o más azulado. Por lo tanto, los lugares donde hay marcadores con un color más amarillento son aquellos donde se han medido valores más altos. En ambos casos, Madrid y Barcelona cuentan con colores que se corresponden a valores altos, pero también zonas como la costa de Asturias o la costa Valenciana. El hecho de que para diferentes contaminantes con distintos rangos de valores existan similitudes, indica que tanto los datos como la visualización de los mismos son correctos.

Otro punto a tener en cuenta es que la diferencia de marcadores es notable, lo cual coincide con lo visto en el diagrama de la figura 28.

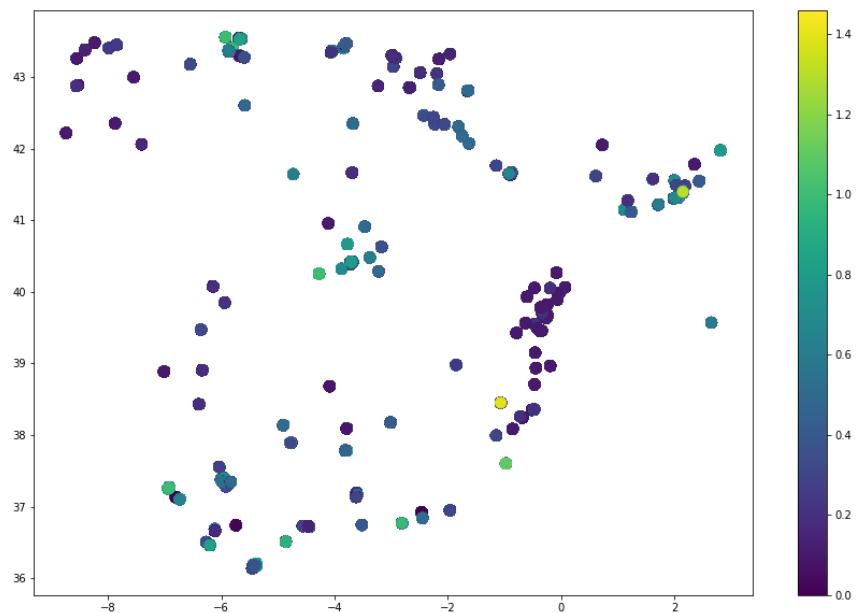


Figura 30. Mapa en el que se muestra la posición de las observaciones de monóxido de carbono junto con su concentración.

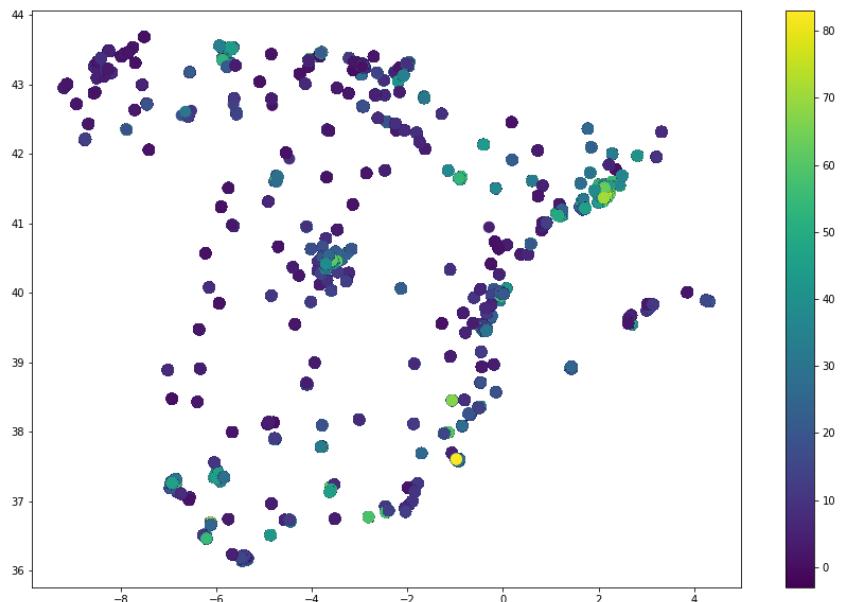


Figura 31. Mapa en el que se muestra la posición de las observaciones de dióxido de nitrógeno junto con su concentración.

Las principales conclusiones que sacamos en relación a este conjunto de datos en un estado aún temprano, es que antes de poder ser utilizados requieren de cierto procesamiento. En primer lugar se deberán identificar y eliminar los valores atípicos y agrupar las entradas en función del día en que se han medido y de su posición geográfica.

Este último punto, también hace pensar en que existen grandes zonas en las que no hay ninguna estación de medición. Por lo tanto, otro punto sobre el que trabajar será la interpolación de datos situados en esas zonas a partir de las mediciones tomadas en estaciones existentes cercanas.

### 7.2.2. Tráfico aéreo

Para este segundo conjunto de datos, el proceso de carga y análisis sigue una estructura similar al visto anteriormente. Comenzamos por cargar cada uno de los 30 archivos CSV mencionados en la sección 6.2 con alrededor de 600.000 entradas cada uno y 17 campos distintos.

En primer lugar, se han de convertir estos datos a un objeto de tipo *Dataframe* para mayor comodidad a la hora de tratarlos. En la siguiente figura se puede apreciar la estructura:

	time	icao24	lat	lon	velocity	heading	vertrate	callsign	onground	alert	spi	squawk	baroaltitude	geoaltitude	lastposupdate	lastcontact	hour
244857	1577887500	Qa001b	44.298365	3.662199	254.784331	162.369920	0.32512	DAH1121	False	False	False	7674.0	11277.60	11468.10	1.577887e+09	1.577887e+09	1577887200
226012	1577890410	4ca27a	40.459946	0.875916	228.913707	226.730189	12.02944	RYR3VD	False	False	False	3271.0	9448.80	9761.22	1.577890e+09	1.577890e+09	1577887200
450327	1577859780	490036	39.549105	-8.867966	NaN	NaN	NaN	NaN	True	False	False	NaN	NaN	NaN	1.577860e+09	1.577860e+09	1577858400
505100	1577902440	020045	39.211183	-4.527039	227.862109	213.582436	0.00000	RAM851W	False	False	False	2121.0	11879.58	12222.48	1.577902e+09	1.577902e+09	1577901600
381573	1577871090	345643	38.356842	-0.314221	197.945323	67.700860	11.37920	VLG40NN	False	False	False	5504.0	5471.16	5814.06	1.577871e+09	1.577871e+09	1577869200

Figura 32. Contenido del *dataframe* con los datos sobre tráfico aéreo.

De la misma forma que en los conjuntos de datos sobre la contaminación del aire, en este caso hay una gran cantidad de campos que no otorgan ningún valor al estudio que se pretende realizar. Por lo tanto, se han eliminado las columnas correspondientes a los campos *time*, *heading*, *vertrate*, *callsign*, *onground*, *alert*, *spi*, *squawk*, *baroaltitude*, *lastposupdate*, *lastcontact* y *hour*.

Cabe destacar, que al tener cada *dataset* separado por días, el campo que contiene la fecha y hora no otorga ningún valor extra y por esa razón queda desecharido.

En cuanto a los campos resultantes, se han renombrado para mayor legibilidad quedando como resultado un *dataframe* con la siguiente estructura:

	ICA024	Latitude	Longitude	Velocity	Altitude
256698	4ca78d	41.368286	-4.872986	218.120616	11620.50
97099	344156	40.646481	3.566312	197.083087	6073.14
175529	440176	42.465231	-6.630313	223.637038	11612.88
429330	34164c	39.224310	-9.488342	253.545749	10789.92
446438	3c496d	41.379501	3.803894	229.380882	10439.40

Figura 33. Contenido del *dataframe* con los datos sobre tráfico aéreo una vez preprocesados.

En la siguiente tabla se muestra la estructura del *dataset* sobre el tráfico aéreo listo para ser tratado y analizado:

Tabla 5  
*Estructura del dataset final sobre el tráfico aéreo*

Campo	Tipo	Descripción
ICAO24	String	Código identificativo de la aeronave
<i>Latitude</i>	Float	Coordenada de latitud
<i>Longitude</i>	Float	Coordenada de longitud
<i>Velocity</i>	Float	Velocidad de la aeronave
<i>Altitude</i>	Float	Altitud de la aeronave

Tabla que muestra la estructura del *dataset* final sobre el tráfico aéreo con datos sobre la posición, altitud y velocidad de la aeronave en cada momento. Cada conjunto de datos está identificado por el día al que pertenece.

Por último, y a modo de análisis del conjunto de datos que acabamos de tratar, se han generado una serie de gráficos para ayudar a entender los datos con los que contamos.

En primer lugar, tenemos un conjunto de los histogramas que representan la distribución de los datos sobre la altitud, latitud, longitud y velocidad.

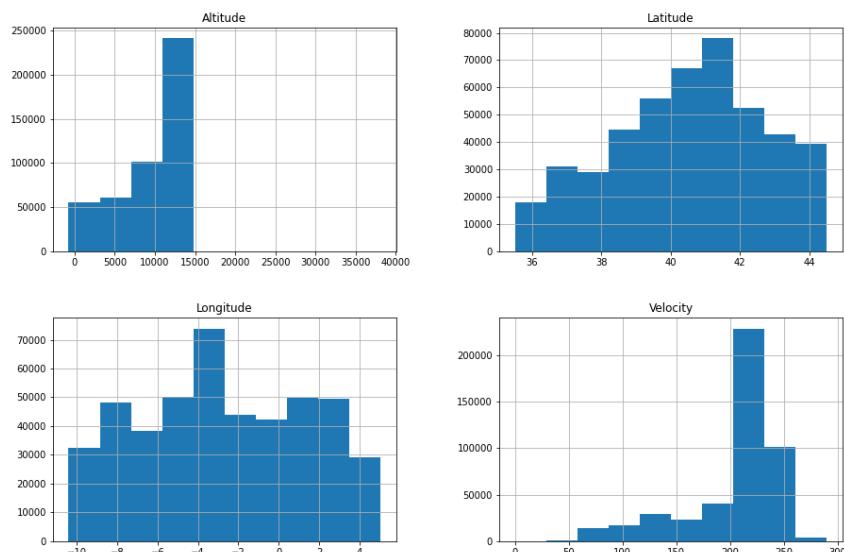


Figura 34. Contenido del *dataframe* con los datos sobre tráfico aéreo una vez preprocesados.

Como cabe esperar en cuanto a la altitud y velocidad, su distribución es muy asimétrica con una orientación hacia valores más altos ya que la mayor parte del tiempo, los aviones se encuentran en una velocidad y altitud determinadas de crucero que no varían apenas con el paso del tiempo. Los valores más bajos de estas gráficas se corresponden a despegues y aterrizajes mayoritariamente.

Sin embargo, la distribución de los valores de latitud y longitud, si que siguen una distribución normal con algunos picos correspondientes a la localización de ciudades con grandes aeropuertos como podrían ser Madrid (40.4, -3.7), Barcelona (41.3, 2.1) o Bilbao (43.1, -2.9).

Para finalizar, se ha generado un mapa de calor sobre el mapa de la Península Ibérica con los datos sobre el tráfico aéreo correspondientes al día 1 de enero:

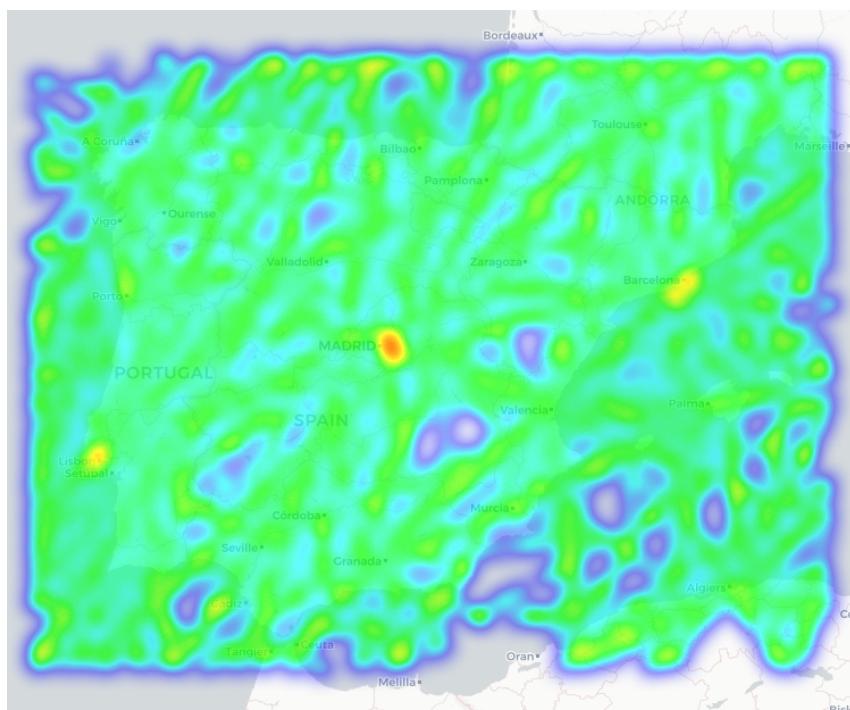


Figura 35. Mapa de calor que muestra el tráfico aéreo sobre la Península Ibérica el día 1 de enero de 2020.

El color va variando en función de la cantidad de entradas concentradas en una misma zona. Por esa razón, Madrid, Barcelona y Lisboa están claramente diferenciados del resto debido a sus grandes aeropuertos y la cantidad de vuelos que salen o acaban en ellos. También es destacable la posibilidad de identificar las rutas que han seguido los aviones durante ese determinado día.

### 7.3. Transformación de los datos

Una vez cargados y analizados los datos, llega el momento de pasar a la segunda fase del denominado proceso ETL (*Extract, Transform, Load*). Como en las anteriores secciones, en primer lugar se describirá el proceso llevado a cabo para los datos sobre la contaminación del aire, para más tarde seguir con el tráfico aéreo.

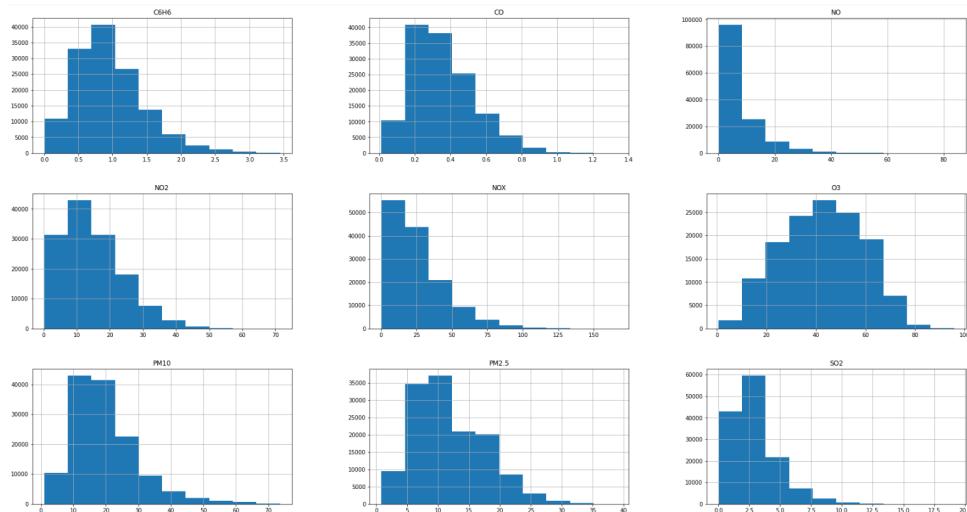
El principal objetivo de las transformaciones que se llevarán a cabo, es el de dotar de cierta estructura a los datos, así como generar nuevos campos a partir de los ya disponibles, o incluso generar datos ficticios con los que facilitar la posible relación entre conjuntos de datos.

#### 7.3.1. Contaminación del aire

Partimos de los datos previamente filtrados y preprocesados contenidos en un único *dataframe* y con la estructura vista en la tabla 4.

La primera transformación llevada a cabo será la eliminación de valores atípicos o *outliers*. Esto son valores que se encuentran a una distancia anormal de otros valores observados en una muestra de la población total. El método seguido para la identificación de estos valores ha sido utilizando la desviación típica<sup>42</sup>.

Tras eliminar los *outliers*, podemos observar en los siguientes histogramas, que la distribución de los valores de los contaminantes ha cambiado con respecto a los vistos en la figura 29:



*Figura 36.* Histogramas con los datos sobre contaminación del aire después de eliminar los valores atípicos encontrados.

Se puede observar como la distribución ha variado en la mayoría de los casos, interpretando así que las entradas eliminadas eran realmente atípicas y por lo tanto contraproducentes a la hora de extraer información veraz de los datos.

Por lo general, sobre los histogramas podemos ver una tendencia hacia los valores más bajos para todos los contaminantes exceptuando el Ozono, cuya curva no ha variado con respecto a los gráficos vistos anteriormente. Esta distribución asimétrica positiva<sup>43</sup> se debe a que la variación en los niveles de estos contaminantes no varía mucho a lo largo del tiempo.

Una vez que contamos con los valores correctos sobre la concentración de los contaminantes, la siguiente transformación se centra en agrupar las observaciones en función del día del mes y calculando la media de cada contaminante para cada jornada. De esta forma, obtenemos los datos por día tal y como sucede en el conjunto de datos sobre el tráfico aéreo. A la hora de relacionar los dos *datasets*, los campos comunes deben contener datos equiparables para así poder identificar correctamente los valores a correlacionar.

Por último, la última transformación llevada a cabo y también la más importante, se trata de la interpolación de valores de cada contaminante en zonas para las que no existen datos reales. Este proceso consiste en identificar las posiciones para las que no existen entradas y calcular un dato ficticio a partir de los valores reales observados alrededor de esta posición.

Para lograr esto, se ha comenzado por dividir el área seleccionada con ayuda de una cuadrícula

<sup>42</sup>[https://es.wikipedia.org/wiki/Desviacion\\_tipica](https://es.wikipedia.org/wiki/Desviacion_tipica)

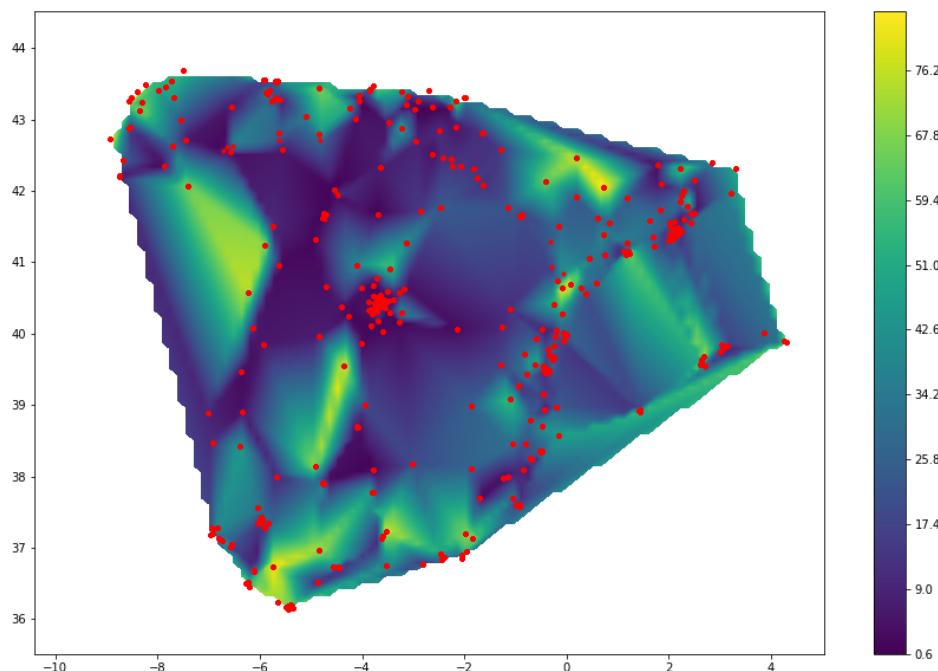
<sup>43</sup>[https://es.wikipedia.org/wiki/Asimetria\\_estadistica](https://es.wikipedia.org/wiki/Asimetria_estadistica)

con 100 divisiones por cada lado, o lo que es lo mismo, 10.000 celdas diferentes. Después, se han tomado los valores del contaminante para el cual deseamos obtener los nuevos datos, y se ha asociado el valor real a cada una de las celdas sobre las que éstos se encuentran.

Por último, se ha utilizado una función llamada *griddata*<sup>44</sup> de la librería *SciPy* para calcular los valores de aquellas celdas para las cuales no se había asociado ninguno. Una vez tenemos datos asociados a cada una de las celdas, añadimos al *dataframe* la posición de cada una como forma de identificar la localización de ahora en adelante.

Este proceso se ha realizado para cada uno de los contaminantes y para cada uno de los 30 días que componen el conjunto de datos.

En la siguiente figura se puede ver como en las zonas donde no hay observaciones reales (zonas sin puntos de color rojo), las entradas que se corresponden a esas posiciones toman un valor similar al de las posiciones con marcadores rojos en función de la distancia a la que se encuentren.



*Figura 37.* Ejemplo del resultado de la interpolación de valores de Ozono para un día concreto sobre el mapa de la Península Ibérica.

Esta transformación de los datos da solución a un problema que realmente imposibilitaba la correlación entre los dos conjuntos de datos ya que en el caso del tráfico aéreo, la distribución geográfica de los datos obtenidos ocupa la mayor parte del área previamente delimitada mientras que en este caso existían una cantidad limitada de zonas en las que se encontraban la mayoría de observaciones.

Para finalizar, se ha confeccionado un *dataframe* a partir de la unión de cada uno de los anteriores una vez aplicadas las transformaciones mencionadas para cada contaminante. El *dataframe* resultante cuenta con 132.193 entradas y está compuesto por 11 campos.

---

<sup>44</sup><https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.griddata.html>

Debido a la unión de cada uno de los *dataframes* de los diferentes contaminantes, es posible que haya entradas con valores nulos. Para estos casos, la primera decisión que se ha tomado es eliminar aquellas filas en las que los valores de los contaminantes son nulos para todos ellos. A parte de esto, para entradas cuyos valores nulos únicamente se corresponden a algunos contaminantes, se ha calculado una media utilizando los valores de las filas anteriores y siguientes siempre que haya sido posible.

Para finalizar, si aún existiesen valores nulos en alguna de las columnas del *dataframe*, estos campos han sido intercambiados por la media de la columna completa. De esta forma, nos aseguramos que no exista ningún valor nulo en la totalidad del conjunto de datos.

	Day	Cell	Cell_x	Cell_y	NO	SO2	NO2	NOX	CO	O3	PM2.5	PM10	C6H6
0	1	632	32	7	2.862296	3.920368	22.617984	25.145702	0.454491	62.542806	11.756215	19.655081	0.000149
1	1	731	31	8	1.197245	7.252647	3.050641	3.969700	0.437532	72.342491	11.756215	19.655081	0.132303
2	1	732	32	8	1.918880	5.683545	4.426156	7.471403	0.371390	67.735192	11.756215	15.553296	0.665350
3	1	733	33	8	5.354259	3.560602	5.189210	11.025467	0.703385	51.998919	11.756215	15.322981	0.086330
4	1	734	34	8	3.164620	2.206142	6.450169	9.391278	0.470465	58.380236	11.756215	13.393692	0.080844

Figura 38. Dataframe final sobre la contaminación del aire después de aplicar todas las transformaciones requeridas.

A continuación se describen cada uno de los campos que contiene el *dataset* final que será usado para extraer los resultados del estudio:

Tabla 6

*Estructura del dataset sobre la contaminación del aire después de aplicar las transformaciones sobre sus datos*

Campo	Tipo	Descripción
<i>Day</i>	Integer	Día (número) al que se corresponde la observación
<i>Cell</i>	Integer	Número de celda que determina la posición de la medición observada
<i>NO</i>	Float	Concentración de monóxido de nitrógeno observada
<i>SO2</i>	Float	Concentración de dióxido de azufre observada
<i>NO2</i>	Float	Concentración de dióxido de nitrógeno observada
<i>NOX</i>	Float	Concentración de óxidos de nitrógeno observada
<i>CO</i>	Float	Concentración de monóxido de carbono observada
<i>O3</i>	Float	Concentración de ozono observada
<i>PM2.5</i>	Float	Concentración de materia particulada (<2.5 micras) observada
<i>PM10</i>	Float	Concentración de materia particulada (<10 micras) observada
<i>C6H6</i>	Float	Concentración de benceno observada

Tabla que muestra la estructura del *dataset* final sobre la contaminación del aire con datos agrupados por la posición en una cuadricula de 100x100 celdas y el día en el que han sido tomados. Cada contaminante supone un campo individual para facilitar su estudio en pasos posteriores.

### 7.3.2. Tráfico aéreo

En este caso, partimos de un *dataset* más sencillo con únicamente los 5 campos vistos en la tabla 5 y de la misma forma que para los datos sobre la contaminación del aire, la primera transformación llevada a cabo será la de identificar y eliminar los valores atípicos. En este caso los únicos campos que pueden tener *outliers* son la altitud y la velocidad, ya que la latitud y la longitud han sido previamente filtradas, y el identificador de los aviones es de tipo *String*.

El proceso para identificar los valores atípicos es el mismo que se ha descrito anteriormente, basándose en la desviación típica y la media de los valores para ambos campos.

En la siguiente imagen se muestran los histogramas de los valores totales de estos campos después de haber aplicado esta transformación:

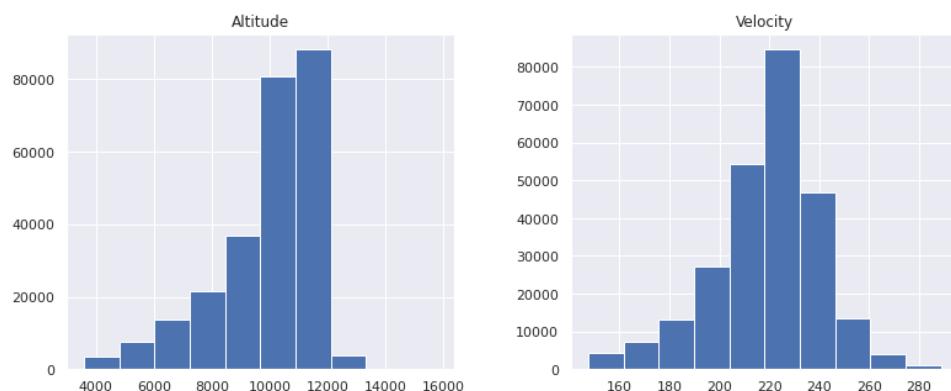


Figura 39. Histograma de los campos *Altitude* y *Velocity* después de haber eliminado sus valores atípicos.

Se puede observar como la distribución, que antes (ver figura 34) era mucho mas asimétrica para ambos casos, pasa a tener una curva más normal en el caso de la velocidad, y aunque para la altitud la cola sigue apuntando a valores mas bajos, es notable el cambio en su distribución.

La siguiente transformación que han sufrido los datos, tiene que ver con la decisión del interpolado de datos en el anterior apartado. Dado que para generar los datos ficticios se ha utilizado una cuadrícula, es razonable utilizar esta división del área para identificar de la misma manera la posición en ambos *datasets*. Por ello, se han utilizado la misma distribución de celdas que en el anterior apartado y se ha asignado a cada entrada, la celda a la que corresponde su posición.

De esta forma, es posible agrupar las entradas por celda y día de la misma manera en la que se ha hecho con el conjunto de datos sobre la calidad del aire. Gracias a esto, ahora es posible tratar e incluso fusionar ambos datasets tomando estos campos como equivalentes.

Es necesario añadir que, al agrupar las entradas por día y celda, se ha calculado la media de altitud y velocidad para cada nuevo grupo, así como la cantidad de vuelos únicos que han pasado por cada celda un determinado día.

	Day	Cell	Cell_x	Cell_y	Velocity	Altitude	ICAO24
0	1	1	1	1	233.235547	11410.053529	136
1	1	2	2	1	225.379147	11235.690000	6
2	1	3	3	1	229.229098	12593.955000	4
3	1	4	4	1	226.117806	12272.010000	4
4	1	5	5	1	218.804393	11948.948276	29

Figura 40. Dataframe final sobre el tráfico aéreo después de aplicar todas las transformaciones requeridas.

Así, al juntar todos los datos sobre cada día en un mismo *dataframe*, se han obtenido 264.709 entradas con 5 campos diferentes, siguiendo la siguiente estructura:

Tabla 7

*Estructura del dataset sobre el tráfico aéreo después de aplicar las transformaciones sobre sus datos*

Campo	Tipo	Descripción
<i>Day</i>	Integer	Día (número) al que se corresponde la observación
<i>Cell</i>	Integer	Número de celda que determina la posición de la medición observada
<i>Velocity</i>	Float	Media calculada de la velocidad para una determinada celda y día
<i>Altitude</i>	Float	Media calculada de la altitud para una determinada celda y día
<i>ICAO24</i>	Integer	Número de aeronaves diferentes para cada determinada celda y día

Tabla que muestra la estructura del *dataset* final sobre el tráfico aéreo con datos agrupados por su posición en una cuadrícula de 100x100 celdas y el día en el que han sido tomados.

## 8. Resultados

Una vez que los datos de ambos campos han sido analizados y preparados, llega el momento de cruzarlos entre sí y observar las relaciones que surgen entre sus campos. Partimos de dos conjuntos de datos cuyas entradas se identifican con campos equivalentes, estos son la celda que establece la posición en el mapa, y el día en el que los datos han sido tomados.

A partir de aquí es posible unir ambos *datasets* y comenzar a analizarlos de forma conjunta. A continuación se muestra la estructura del *dataframe* resultante de fusionar ambas tablas:

Tabla 8

*Estructura del dataset resultante de la unión entre los datos sobre contaminación del aire y tráfico aéreo*

Campo	Tipo	Descripción
<i>Day</i>	Integer	Día (número) al que se corresponde la observación
<i>Cell</i>	Integer	Número de celda que determina la posición de la medición observada
<i>NO</i>	Float	Concentración de monóxido de nitrógeno observada
<i>SO2</i>	Float	Concentración de dióxido de azufre observada
<i>NO2</i>	Float	Concentración de dióxido de nitrógeno observada
<i>NOX</i>	Float	Concentración de óxidos de nitrógeno observada
<i>CO</i>	Float	Concentración de monóxido de carbono observada
<i>O3</i>	Float	Concentración de ozono observada
<i>PM2.5</i>	Float	Concentración de materia particulada (<2.5 micras) observada
<i>PM10</i>	Float	Concentración de materia particulada (<10 micras) observada
<i>C6H6</i>	Float	Concentración de benceno observada
<i>Velocity</i>	Float	Media calculada de la velocidad para una determinada celda y día
<i>Altitude</i>	Float	Media calculada de la altitud para una determinada celda y día
<i>ICAO24</i>	Integer	Número de aeronaves diferentes para cada determinada celda y día

Tabla que muestra la estructura del *dataset* final resultante de unir los conjuntos de datos anteriores ya procesados agrupados por su posición en una cuadrícula de 100x100 celdas y el día en el que han sido tomados.

En primer lugar, se han analizado los diagramas de dispersión entre cada uno de los campos creados con ayuda de la función *pairplot*<sup>45</sup> de la librería *Seaborn* tal y como se pueden ver en la siguiente imagen:

<sup>45</sup><https://seaborn.pydata.org/generated/seaborn.pairplot.html>

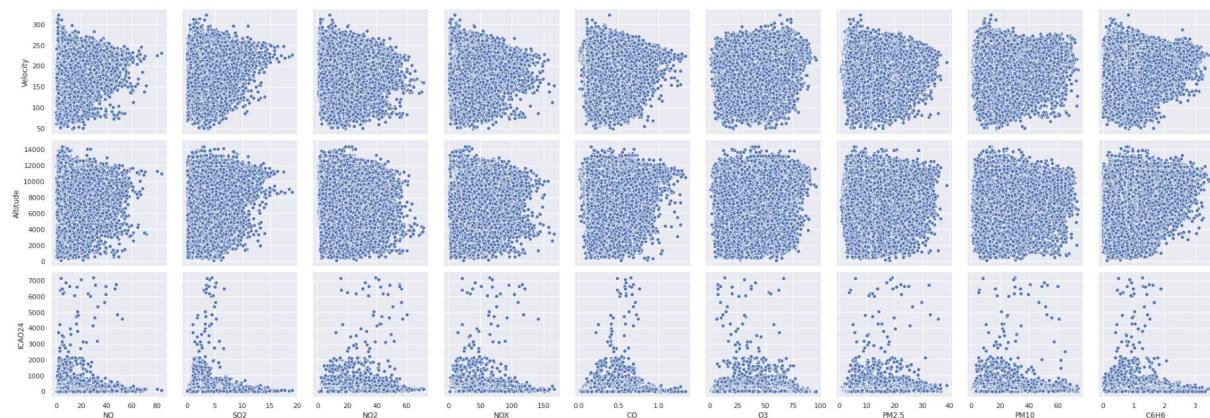


Figura 41. Conjunto de diagramas de dispersión entre los campos del *dataset* final.

Lo primero que salta a la vista es que los puntos de todos los gráficos están muy dispersos sobre todo el rango tanto del eje *X* (el valor del contaminante), como del eje *Y* (altitud, velocidad y número de aviones). Esto indica que posiblemente no exista relación directa, ya sea positiva o negativa, entre ninguno de los campos observados. Sin embargo, existe una manera de obtener un valor numérico que describa de manera matemática la relación, o ausencia de ésta, entre cada uno de los campos.

Esta técnica se basa en el cálculo de un coeficiente de correlación que describe de manera numérica como se comportan los valores de un campo respecto a los de otro. Si el coeficiente de correlación es igual o cercano a 1, esto indicaría que para cada incremento positivo en un campo, existe un incremento positivo de una proporción fija en el otro campo. Por otro lado, un coeficiente de correlación igual o cercano a -1 indicaría lo opuesto, para cada incremento positivo en un campo, existe un detrimento negativo que sigue una proporción fija sobre el otro campo.

En caso de que el coeficiente de correlación sea igual o cercano a 0, significaría que para cada incremento de los valores de un campo, no existe aumento ni detrimento en los valores del otro, y por lo tanto no hay una relación directa entre ambos campos.

Para calcular estos coeficientes con los datos de los que disponemos, se ha utilizado la función *corr*<sup>46</sup> de la librería pandas. Por otro lado, con el objetivo de mostrar de forma visual los resultados del cálculo de los coeficientes de correlación para cada uno de los pares de campos, se ha utilizado la función *heatmap*<sup>47</sup> de Seaborn para mostrar la matriz de correlación calculada de forma que su interpretación sea fácil e intuitiva:

<sup>46</sup><https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html>

<sup>47</sup><https://seaborn.pydata.org/generated/seaborn.heatmap.html>

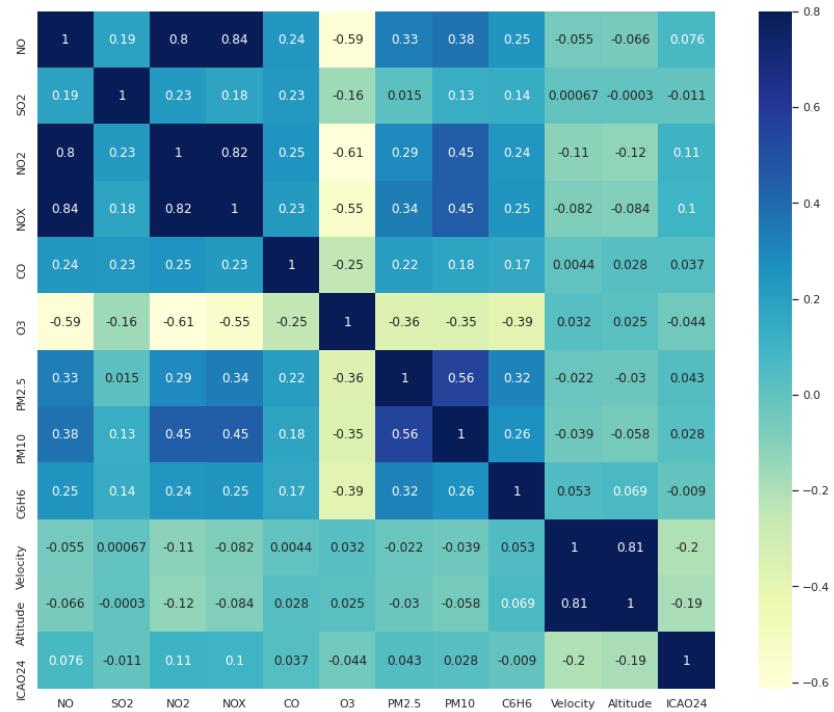


Figura 42. Matriz de correlación que muestra los coeficientes calculados para todos los pares de campos del dataset.

En esta imagen, nos debemos fijar en el cuadrante inferior izquierdo, el rectángulo formado por las celdas de color más claro. En cada una de estas celdas encontramos el coeficiente de correlación entre los campos situados en el eje X e Y.

Al igual que en el conjunto de diagramas de dispersión vistos en la figura 41, los resultados que obtenemos en este caso no indican una relación entre ninguno de los campos. Los coeficientes más altos que encontramos se corresponden a los campos de la familia de los óxidos de nitrógeno (NOX, NO y NO<sub>2</sub>) con todos los campos sobre el tráfico aéreo con un valor de alrededor de |0.11| y por otro lado algunos valores cercanos a |0.05| en compuestos como las materias particuladas o el benceno.

Dados estos resultados, tanto de los diagramas de dispersión como de la matriz de correlación, no ha sido posible probar, partiendo del conjunto de datos, que exista una relación directa entre la contaminación del aire y el tráfico aéreo.

Cabe destacar que este resultado no quiere decir que no exista dicha relación, únicamente que desde los datos con los que se han trabajado no se puede demostrar dicha premisa.

Entre las posibles causas de la incapacidad de encontrar una relación directa, tenemos la gran limitación que supone trabajar con un área relativamente pequeña así como un rango determinado de fechas con lo que ello supone. Por otro lado, los efectos de fuentes de contaminación externas, como podrían ser industrias, contaminación de otros vehículos, o incluso el clima y las temperaturas; pueden suponer un gran efecto sobre la muestra de datos recogidos y afectar negativamente a la búsqueda de relaciones entre los campos sobre el tráfico aéreo con los que se ha trabajado.

## 9. Conclusiones

En primer lugar y en relación a los principales requisitos y objetivos del proyecto, se han completado de forma satisfactoria ofreciendo un estudio imparcial sobre los dos campos propuestos y unos resultados concluyentes que en este caso niegan la premisa de que existe una relación directa entre la contaminación del aire y el tráfico aéreo en el área previamente determinada y durante un rango de fechas previamente delimitado.

El proceso completo de investigación, extracción, transformación y obtención de resultados ha supuesto todo un reto debido a varios factores. En primer lugar, los campos de estudio que trata el proyecto eran totalmente nuevos y ha requerido de gran esfuerzo el entendimiento y asimilación de los conceptos teóricos que los componen.

Por otro lado, la falta de experiencia en el uso de la plataforma de Amazon Web Services, ha supuesto una dificultad inicial que se ha ido minimizando con la ayuda de guías externas así como la ayuda del tutor del TFM.

En cuanto a conclusiones concretas sacadas a partir de los resultados obtenidos tras el procesamiento de los datos, encontramos una leve relación entre los óxidos de nitrógeno y el tráfico aéreo medido, que aunque por si sola no sea un resultado de peso a destacar, sugiere la posibilidad de que si se consiguiese aislar la procedencia de todos los contaminantes, la relación obtenida sería más clara y directa.

A parte de esto, tras la realización del proyecto se ha llegado a la conclusión de que algunas de las decisiones tomadas, como el interpolado de datos, han sido contraproducentes ya que al proporcionar datos para zonas en las que inicialmente no existían, han falseado las correlaciones calculadas a partir de éstos.

Asimismo, algunas deducciones a las que se han llegado tras completar los objetivos propuestos son la irregularidad del tráfico aéreo en el espacio aéreo de España, estando mucho más congestionada la zona central y el noreste de la península. Por otro lado, es muy notable la ausencia de estaciones de medición de la calidad del aire en grandes zonas de Extremadura, Castilla la Mancha y Aragón.

El hecho de no haber encontrado un resultado positivo al finalizar el estudio, no hace que el resto de componentes del proceso seguido hasta llegar a ese punto pierdan valor. Son las técnicas de *Big Data* aplicadas y los conocimientos adquiridos sobre bases de datos, computación en la nube y visualización y procesamiento de datos los que han hecho posible alcanzar los objetivos propuestos ofreciendo una plataforma con la que tratar ésta y futuras cuestiones relacionadas.

## 10. Líneas de trabajo futuras

A medida que se ha ido realizando el proyecto y tomando decisiones sobre el mismo, se han descartado algunas líneas de trabajo por falta de tiempo, recursos o de datos. En este apartado se detallan algunas ideas u opciones a tener en cuenta para ampliar el estudio en varias de las fases que lo componen, desde los datos utilizados a la utilización de nuevas técnicas para la obtención de resultados.

En primer lugar, con los mismos conjuntos de datos sería posible estudiar cómo cambian los resultados de este mismo estudio si limitasen ciertos campos como la altitud o velocidad de los aviones. En estudios relacionados (ver sección 4.3), hemos visto cómo se concluye con que los efectos observados sobre la calidad del aire en altitudes de crucero de entre 9 y 11 kilómetros son mayores que los observados a nivel de tierra en despegues y aterrizajes.

Otro punto a tener en cuenta son los valores interpolados para la contaminación del aire en zonas sin estaciones de medición. Existe la posibilidad de que estos valores introduzcan mediciones lejanas a las reales desembocando así en valores erróneos demasiado elevados. Por lo tanto, una posible mejora en versiones futuras del proyecto pasaría por modificar el interpolado de los nuevos datos teniendo esto en cuenta, o incluso hacer las correlaciones sin necesidad de interpolar nuevos datos y únicamente centrarse en zonas para las que existan valores reales sobre la contaminación del aire.

Por otro lado, y tal y como se ha mencionado en la explicación de los resultados obtenidos, es posible que tanto el área elegida para el estudio, como la periodicidad de los datos recogidos hayan podido ser limitantes a la hora de encontrar una relación clara entre los datos. En trabajos venideros sería razonable cambiar tanto el área geográfica elegida, como el tiempo entre observaciones para así tratar de obtener resultados de mayor calidad.

En cuanto a la incorporación de nuevos datos al estudio, existen infinidad de campos desde los que añadir complejidad al proyecto. Una clara opción pasa por tener en cuenta la contaminación causada por fuentes externas como industrias y otros vehículos terrestres. Más concretamente, se usaría esta información para identificar aquellos contaminantes que más se ven afectados por estas causas externas y obtener así conclusiones más claras sobre el efecto del tráfico aéreo.

Otras opciones similares a la anterior, podrían estar relacionadas con la adición de otros campos de información y ver la relación con los resultados obtenidos en este estudio. Por ejemplo ver los efectos del clima y temperatura, o la localización del estudio en diferentes zonas del mundo (cercano al ecuador o a los polos, hemisferio norte o sur, observaciones sobre el océano o sobre continentes, etc).

En cuanto a técnicas y tecnologías utilizadas, existen muchas opciones a tener en cuenta para futuros trabajos que tomen este como base. Una de ellas pasa por integrar la fuente de datos en el propio proyecto construyendo un sistema de monitorización de la calidad del aire con arduino y distintos sensores<sup>48</sup>, así como un receptor ADS-B para recibir las comunicaciones del tráfico aéreo cercano<sup>49</sup>.

---

<sup>48</sup><https://www.pubnub.com/blog/diy-air-quality-monitoring-system-with-realtime-readings-and-live-alerts/>

<sup>49</sup><https://www.flightradar24.com/build-your-own>

Por último, con el objetivo de tratar otros campos relacionados con el *Big Data*, sería posible abordar el proyecto desde perspectivas diferentes como la predicción de la calidad del aire en función del tráfico aéreo utilizando técnicas de aprendizaje automático.

## Referencias bibliográficas

- Agency, E. E. (2019). *Air quality in Europe — 2019 report* (inf. téc.). <https://www.eea.europa.eu/publications/air-quality-in-europe-2019>
- Agency, E. E. (2020). *European Union emission inventory report 1990-2018* (inf. téc.). <https://www.eea.europa.eu/publications/european-union-emission-inventory-report-1990-2018>
- Annesi-Maesano, I. (2017). The air of Europe: where are we going? *European Respiratory Review*, 26(146), <https://err.ersjournals.com/content/26/146/170024.full.pdf>. <https://doi.org/10.1183/16000617.0024-2017>
- Correlation Coefficient: Simple Definition, Formula, Easy Calculation Steps. (2020). <https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/>
- European Environment Agency. (2020). <https://www.eea.europa.eu/themes/air/air-pollution-sources>
- Fichter, C. (2009). *Climate impact of air traffic emissions in dependency of the emission location and altitude* (inf. téc.) [Thesis submitted for the degree of Doctor of Philosophy, Manchester Metropolitan University, Manchester, UK]. Thesis submitted for the degree of Doctor of Philosophy, Manchester Metropolitan University, Manchester, UK. <https://elib.dlr.de/61768/>
- Gilmour, J. B., Lui, A. W. & Briggs, D. C. (1986). Emr. Amazon. <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-master-core-task-nodes.html>
- How ADS-B works. (s.f.). Air Services Australia. <https://www.airservicesaustralia.com/projects/ads-b/how-ads-b-works/>
- How to Mount S3 bucket on EC2 Linux Instance. (2020). <https://cloudkul.com/blog/mounting-s3-bucket-linux-ec2-instance/>
- Hudgeon, D. & Nichol, R. (2020). Machine learning for business: using Amazon SageMaker and Jupyter. Manning Publications Co. <https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-preprocess-data-transform.html>
- Inspecting air pollution data from OpenAQ using Colab, Pandas, and BigQuery. (s.f.). <http://www.athoughtabroad.com/2019/02/09/inspecting-air-pollution-data-from-openaq-using-colab-pandas-and-bigquery>
- Lee, H., Olsen, S., Wuebbles, D. & Youn, D. (2013). Impacts of aircraft emissions on the air quality near the ground. *Atmospheric Chemistry & Physics Discussions*, 13. <https://doi.org/10.5194/acpd-13-689-2013>
- OpenAQ. (2020). <https://openaq.org/>
- Parra, S. (2020). El 3,5 % del calentamiento global es atribuible solamente al transporte aéreo. *Xataka Ciencia*. <https://www.xatakaciencia.com/medio-ambiente/3-5-calentamiento-global-attribuible-solamente-al-transporte-aereo>
- Real-time Air Quality Index. (2020). <https://aqicn.org/map/world/>
- Wikipedia. (2020). Sistema de Vigilancia Dependiente Automática — Wikipedia, La enciclopedia libre. [https://es.wikipedia.org/wiki/Sistema\\_de\\_Vigilancia\\_Dependiente\\_Autom%C3%A1tica](https://es.wikipedia.org/wiki/Sistema_de_Vigilancia_Dependiente_Autom%C3%A1tica)
- World Health Organization - Air pollution. (2020). <https://www.who.int/health-topics/air-pollution>

# Anexos

**Anexo I: Script Python para generar las *queries* necesarias para descargar los datos de OpenSky**

```
from datetime import datetime, timedelta

def gen_days():
    """
    Función que devuelve los días en formato timestamp UNIX
    """
    start_date = datetime(2020, 1, 1)
    end_date = datetime(2020, 1, 31)
    d = start_date
    dates = []
    while d < end_date:
        d_a = d
        d += timedelta(days=1)
        t1, t2 = gen_hours(d_a.timestamp(), d.timestamp())
        dates.append([round(d_a.timestamp()), round(d.timestamp()),
                      round(t1), round(t2)])
    return dates

def gen_hours(t1, t2):
    """
    Función que devuelve la hora a la que pertenece cada timestamp
    """
    t1 = int(t1)
    t2 = int(t2)
    h1 = t1 - (t1 % 3600)
    h2 = t2 - (t2 % 3600)
    return h1, h2

def gen_query(q):
    """
    Función que genera la query para cada día del mes junto con la hora
    a la que pertenece.
    """
    q1 = "SELECT * FROM state_vectors_data4 WHERE lat > 35.512 AND lat < 44.512
          AND lon > -10.415 AND lon < 5.054 AND time>="
    q2 = " AND time<="
    q3 = " AND hour>="
    q4 = " AND hour<="
    q5 = " AND time%30=0;"
    return q1 + str(q[0]) + q2 + str(q[1]) + q3 + str(q[2]) + q4 + str(q[3]) + q5
```

```

if __name__ == '__main__':
    ...
    Función principal que escribe por pantalla las queries necesarias para
    obtener los datos deseados de la base de datos e OpenSky
    ...
    d = gen_days()
    for i in range(len(d)):
        query = str(i+1) + " ENE --> " + gen_query(d[i])
        print(query)

```

## Anexo II: Código Python utilizado para la extracción, transformación y carga de datos sobre la contaminación aérea

```

import json
import numpy as np
import pandas as pd
import boto3

BUCKET_NAME = 'ui1-tfm-data'

# Credenciales de autentificación de AWS
s3 = boto3.client('s3', aws_access_key_id = "ASIAJAXIWLQYIZ5NYRRVC",
                  aws_secret_access_key = "+JIF096GBK6dMvIsy1QPCiXlwIB0m3RCnztNDsqG",
                  aws_session_token="FwoGZXIxYXdzEBsaDEUmtmKSyIojMq0X
1SLLAap50ogxmDB/IqREUsGtwZYe0+OD1d1IZLwXZ7fJB1s59bY
woOB8zzsBoTlmBnZ1EKQfQCwv/ywXzhiphqQXEkFEGru7RJWxYL
+NeytRaBjzhDMiKs2sv8r6iEFzpmupV6zbZwZ0GPe7TLailzK/V
xHX7bed0o6TvkkSbZKITbEnYpjKE0+ZXRJQ3Lzz8xe0RBVppHuY
I1EzNkh9RcffkjXE3Vo9SYa1LfkhLwqLLm+VhHzoV90lnNcjGR6
W35yqYLCQiBo19vs1HpJdKOuA2PoFMi3NDABviL0R8PebcCDnNk
PZqELkRxfhd0n5QfzM03Zjhgm3mCufjkpttSjCG3Y=")

# Descargamos los ficheros con los que vamos a trabajar desde el bucket
previamente creado en S3
s3.download_file(BUCKET_NAME, 'air-data/stations.json', '/tmp/stations.json')
s3.download_file(BUCKET_NAME, 'air-data/spain_airQuality.csv',
                 '/tmp/spain_airQuality.csv')

# Cargamos el archivo que contiene las estaciones
data = json.load(open('/tmp/stations.json'))

# Creamos el dataframe
df_stations = pd.DataFrame(data["data"])

# Filtramos las estaciones en España

```

```

df_stations_spain = df_stations.loc[df_stations['CountryOrTerritory'] == 'Spain']

# Eliminamos las estaciones de fuera de la península (Islas Canarias)
# lat > 35.512 AND lat < 44.512 AND lon > -10.415 AND lon < 5.054
df_stations_spain = df_stations_spain[(
    df_stations_spain.SamplingPoint_Latitude > 35.512)
    & (df_stations_spain.SamplingPoint_Latitude < 44.512)
    & (df_stations_spain.SamplingPoint_Longitude > -10.415)
    & (df_stations_spain.SamplingPoint_Longitude < 5.054)]

# Seleccionamos los campos relevantes
df_stations_spain = df_stations_spain[['StationLocalId', 'SamplingPoint_Latitude',
                                         'SamplingPoint_Longitude']]

# Renombramos los campos
df_stations_spain.columns = ['Station', 'Latitude', 'Longitude']

# Reseteamos el índice del dataframe
df_stations_spain = df_stations_spain.reset_index(drop=True)

# Cargamos el archivo y creamos el dataframe con las observaciones de
# los contaminantes
df = pd.read_csv('/tmp/spain_airQuality.csv')

# Seleccionamos los campos relevantes
df_measurements = df[['AirQualityStation', 'AirPollutant', 'Concentration',
                      'UnitOfMeasurement', 'DatetimeBegin']]

# Renombramos los campos
df_measurements.columns = ['Station', 'AirPollutant', 'Concentration',
                           'UnitOfMeasurement', 'Datetime']

# Eliminamos las entradas cuyas mediciones son nulas
df_measurements = df_measurements[df_measurements['Concentration'].notna()]

# Convertimos el campo de la fecha a formato Datetime
df_measurements['Datetime'] = pd.to_datetime(df_measurements['Datetime'])

# Seleccionamos las entradas que se corresponden al mes de enero
df_measurements = df_measurements.loc[df_measurements['Datetime'] < '2020-2-1']

# Reseteamos el índice del dataframe
df_measurements = df_measurements.reset_index(drop=True)

# join con el dataframe de las estaciones para obtener su localización
combined_df = df_measurements.merge(df_stations_spain, left_on='Station',
                                     right_on='Station')

```

```

right_on='Station')

combined_df = combined_df[['AirPollutant', 'Concentration', 'UnitOfMeasurement',
                           'Station', 'Latitude', 'Longitude', 'Datetime']]

def get_grid(lon_steps, lat_steps, n):
    """
    Función que genera un diccionario con la posición de las celdas resultantes
    de dividir el área en nxn celdas
    """
    grid_dict = {}

    lat_stride = lat_steps[1] - lat_steps[0]
    lon_stride = lon_steps[1] - lon_steps[0]

    count = 0
    for lat in lat_steps[:-1]:
        for lon in lon_steps[:-1]:
            count = count + 1
            # Define dimensions of box in grid
            upper_left = [lon, lat + lat_stride]
            upper_right = [lon + lon_stride, lat + lat_stride]
            lower_right = [lon + lon_stride, lat]
            lower_left = [lon, lat]
            grid_dict[count] = [upper_left[0], upper_left[1],
                                lower_right[0], lower_right[1]]
    return grid_dict

N_DIVISIONS = 100 # Número de divisiones horizontales y verticales
x_steps = np.linspace(-10.415, 5.054, N_DIVISIONS + 1) # Longitude
y_steps = np.linspace(35.512, 44.512, N_DIVISIONS + 1) # Latitude

# Diccionario que contiene las coordenadas de cada celda
grid_dict = get_grid(x_steps, y_steps, N_DIVISIONS)

def remove_outliers(df):
    """
    Identifica y elimina los outliers del campo 'Concentration'
    """
    return df[((df.Concentration - df.Concentration.mean()) /
               df.Concentration.std()).abs() < 3]

def group_by_day_station(df):

```

```

"""
Para cada día del mes, se obtiene la media de las mediciones en cada estación
"""

return df[['Datetime', 'Concentration', 'Station', 'Longitude', 'Latitude']]
    .groupby([df['Datetime'].dt.day, 'Station']).mean().reset_index()

from scipy.interpolate import griddata
def interpolate(df, lon_steps, lat_steps, n):
    """
    Crea una red con puntos cada 100Km y genera nuevos datos para estos puntos
    interpolando los datos ya conocidos
    """

    x = df["Longitude"].to_numpy()
    y = df["Latitude"].to_numpy()
    z = df["Concentration"].to_numpy()

    xi, yi = np.meshgrid(lon_steps, lat_steps)

    # interpolate
    zi = griddata((x,y),z,(xi,yi),method='linear')

    # Utilizamos los nuevos valores para crear un nuevo dataframe
    x_column = []
    for i in xi:
        for j in i:
            x_column.append(j)

    y_column = []
    for i in yi:
        for j in i:
            y_column.append(j)

    z_column = []
    for i in zi:
        for j in i:
            z_column.append(j)

    data = [x_column, y_column, z_column]
    columns = ['x', 'y', 'z']
    return pd.DataFrame(np.array(data).T, columns=columns)

def interpolateAll(df, n):
    """
    Tiene como resultado un dataframe en el que se guardan los datos generados
    cada día según el contaminante
    """

    interpolated_pollutant_df = pd.DataFrame()

```

```

for n_day in range(1,31):
    day_df = df.loc[df['Datetime'] == n_day]
    interpolated_day_df = interpolate(day_df, x_steps, y_steps, n)
    interpolated_day_df['Day'] = n_day
    interpolated_pollutant_df = interpolated_pollutant_df
                                .append(interpolated_day_df)

return interpolated_pollutant_df

def locate_point(grid_x, grid_y, point_x, point_y):
    """
    Localiza la celda a la que pertenece un punto y devuelve sus indices
    """
    x_step = grid_x[1]-grid_x[0]
    y_step = grid_y[1]-grid_y[0]
    cell_x = ((point_x - grid_x[0])//x_step) + 1
    cell_y = ((point_y - grid_y[0])//y_step) + 1
    return cell_x, cell_y

def get_cell_num(cell_x, cell_y, n):
    """
    Devuelve el número de una celda dados sus indices X e Y
    """
    return (((cell_y - 1) * (n-1)) + cell_x)

def addCellToAll(df, n):
    """
    Función que itera sobre todas las entradas y añade la celda en la que
    se encuentra el vuelo en cada momento
    """
    df['Cell'] = ''
    for i, row in df.iterrows():
        point_x = row[0] # Longitude
        point_y = row[1] # Latitude
        cell_x, cell_y = locate_point(x_steps, y_steps, point_x, point_y)
        cell_num = get_cell_num(cell_x, cell_y, n+1)
        df.at[i,'Cell'] = int(cell_num)
    return df

# Separamos por tipo de contaminante
df_NO = combined_df.loc[combined_df['AirPollutant'] == 'NO']
df_SO2 = combined_df.loc[combined_df['AirPollutant'] == 'SO2']
df_NO2 = combined_df.loc[combined_df['AirPollutant'] == 'NO2']
df_NOX = combined_df.loc[combined_df['AirPollutant'] == 'NOX as NO2']
df_CO = combined_df.loc[combined_df['AirPollutant'] == 'CO']
df_O3 = combined_df.loc[combined_df['AirPollutant'] == 'O3']
df_PM25 = combined_df.loc[combined_df['AirPollutant'] == 'PM2.5']
df_PM10 = combined_df.loc[combined_df['AirPollutant'] == 'PM10']

```

```

df_C6H6 = combined_df.loc[combined_df['AirPollutant'] == 'C6H6']

# Identificamos y eliminamos outliers de cada nuevo dataframe
df_NO = remove_outliers(df_NO)
df_SO2 = remove_outliers(df_SO2)
df_NO2 = remove_outliers(df_NO2)
df_NOX = remove_outliers(df_NOX)
df_CO = remove_outliers(df_CO)
df_O3 = remove_outliers(df_O3)
df_PM25 = remove_outliers(df_PM25)
df_PM10 = remove_outliers(df_PM10)
df_C6H6 = remove_outliers(df_C6H6)

# Agrupamos cada contaminante por día y estación y hacemos la media
df_NO_by_day = group_by_day_station(df_NO)
df_SO2_by_day = group_by_day_station(df_SO2)
df_NO2_by_day = group_by_day_station(df_NO2)
df_NOX_by_day = group_by_day_station(df_NOX)
df_CO_by_day = group_by_day_station(df_CO)
df_O3_by_day = group_by_day_station(df_O3)
df_PM25_by_day = group_by_day_station(df_PM25)
df_PM10_by_day = group_by_day_station(df_PM10)
df_C6H6_by_day = group_by_day_station(df_C6H6)

# Utilizamos los datos actuales para obtener un valor interpolado
# de los anteriores para cada 100Km
interpolated_df_NO = interpolateAll(df_NO_by_day, N_DIVISIONS)
interpolated_df_SO2 = interpolateAll(df_SO2_by_day, N_DIVISIONS)
interpolated_df_NO2 = interpolateAll(df_NO2_by_day, N_DIVISIONS)
interpolated_df_NOX = interpolateAll(df_NOX_by_day, N_DIVISIONS)
interpolated_df_CO = interpolateAll(df_CO_by_day, N_DIVISIONS)
interpolated_df_O3 = interpolateAll(df_O3_by_day, N_DIVISIONS)
interpolated_df_PM25 = interpolateAll(df_PM25_by_day, N_DIVISIONS)
interpolated_df_PM10 = interpolateAll(df_PM10_by_day, N_DIVISIONS)
interpolated_df_C6H6 = interpolateAll(df_C6H6_by_day, N_DIVISIONS)

# Renombramos la columna 'Concentration' en función del contaminante
# antes de unificar todos los dataframes un uno solo
interpolated_df_NO.columns = ['Longitude', 'Latitude', 'NO', 'Day']
interpolated_df_SO2.columns = ['Longitude', 'Latitude', 'SO2', 'Day']
interpolated_df_NO2.columns = ['Longitude', 'Latitude', 'NO2', 'Day']
interpolated_df_NOX.columns = ['Longitude', 'Latitude', 'NOX', 'Day']
interpolated_df_CO.columns = ['Longitude', 'Latitude', 'CO', 'Day']
interpolated_df_O3.columns = ['Longitude', 'Latitude', 'O3', 'Day']
interpolated_df_PM25.columns = ['Longitude', 'Latitude', 'PM2.5', 'Day']
interpolated_df_PM10.columns = ['Longitude', 'Latitude', 'PM10', 'Day']
interpolated_df_C6H6.columns = ['Longitude', 'Latitude', 'C6H6', 'Day']

```

```

# Por último, hacemos un join de todos los contaminantes para tenerlos
# en un único dataframe
from functools import reduce
final_df = reduce(lambda x,y:
    pd.merge(x,y, on=['Longitude' , 'Latitude' , 'Day'], how='outer'),
    [interpolated_df_NO, interpolated_df_SO2, interpolated_df_N02,
     interpolated_df_NOX, interpolated_df_CO, interpolated_df_O3,
     interpolated_df_PM25, interpolated_df_PM10, interpolated_df_C6H6])

# Eliminamos las entradas cuyas mediciones son nulas para todos
# los contaminantes
final_df = final_df.dropna(thresh=9)

# Rellenamos los valores nulos que quedan con la media de las
# filas anteriores y las siguientes
final_df = final_df.where(final_df.notnull(),
    other=(final_df.fillna(method='ffill')
        + final_df.fillna(method='bfill'))/2)

# Los nulos que no se han podido calcular, se rellenan con la media
# de toda la columna
for i in final_df.columns[final_df.isnull().any(axis=0)]:
    final_df[i].fillna(final_df[i].mean(), inplace=True)

# Reseteamos el índice del dataframe
final_df = final_df.reset_index(drop=True)

# Asignamos una celda a cada entrada del dataframe
final_df = addCellToAll(final_df, N_DIVISIONS)

# Ordenamos las columnas para mejor legibilidad
final_df = final_df[['Day', 'Cell', 'NO', 'SO2', 'N02', 'NOX', 'CO',
    'O3', 'PM2.5', 'PM10', 'C6H6']]

# Guardamos el dataframe final como un archivo json y lo almacenamos en S3
final_df.to_json(r'/tmp/final_airQuality_dataset.json')
s3.upload_file('/tmp/final_airQuality_dataset.json', BUCKET_NAME,
    'air-data/final_airQuality_dataset.json')

```

### Anexo III: Código Python utilizado para la extracción, transformación y carga de datos sobre la contaminación aérea

```

import json
import numpy as np
import pandas as pd
import boto3

BUCKET_NAME = 'ui1-tfm-data'

# Credenciales de autentificación de AWS
s3 = boto3.client('s3', aws_access_key_id = "ASIAJAIWQLYIZ5NYRRVC",
                  aws_secret_access_key = "+JIF096GBK6dMvIsy1QPCiXlwIB0m3RCnztNDsqG",
                  aws_session_token="FwoGZXIxYXdzEBsaDEUmtmKSyIojMq0X
1SLLAap50ogxmDB/IqREUsGtwZYe0+0Dldl1ZLwXZ7fJBls59bY
woOB8zzsBoTlmBnZ1EKQfQCwv/ywXzhiphqQXEkFEGru7RJWxYL
+NeytRaBjzhDMiKs2sv8r6iEFzpmupV6zbZwZOGPe7TLailzK/V
xHX7bed0o6TvkkSbZKITbEnYpjKE0+ZXRJQ3LzZ8xe0RBVppHuY
I1EzNkh9RcffkjXE3Vo9SYa1LfklHwqLLm+VhHZoV90lnNcjGR6
W35yqYLCQiBo19vs1HpJdKOuA2PoFMi3NDABviLOR8PebcCDnNk
PZqElkRxfhd0n5QfzM03Zhgm3mCufjkpttSjCG3Y=")

# Descargamos los ficheros con los que vamos a trabajar desde el bucket
previamente creado en S3
for n in range(1, 31):
    file_name = 'enero_' + str(n) + '.csv'
    origin_file = 'opensky-data/' + file_name
    dest_file = '/tmp/' + file_name
    s3.download_file(BUCKET_NAME, origin_file, dest_file)

def preprocess_vuelos_df(df):
    # Seleccionamos los campos relevantes
    df = df[['icao24', 'lat', 'lon', 'velocity', 'geoaltitude']]

    # Renombramos los campos
    df.columns =[['ICAO24', 'Latitude', 'Longitude', 'Velocity', 'Altitude']]

    # Eliminamos las entradas cuyas mediciones son nulas
    df = df[df['Altitude'].notna()]

    # Reseteamos el índice del dataframe
    df = df.reset_index(drop=True)

    return df

```

```

def get_grid(lon_steps, lat_steps, n):
    """
    Función que genera un diccionario con la posición de las celdas resultantes
    de dividir el área en nxn celdas
    """
    grid_dict = {}

    lat_stride = lat_steps[1] - lat_steps[0]
    lon_stride = lon_steps[1] - lon_steps[0]

    count = 0
    for lat in lat_steps[:-1]:
        for lon in lon_steps[:-1]:
            count = count + 1
            # Define dimensions of box in grid
            upper_left = [lon, lat + lat_stride]
            upper_right = [lon + lon_stride, lat + lat_stride]
            lower_right = [lon + lon_stride, lat]
            lower_left = [lon, lat]
            grid_dict[count] = [upper_left[0], upper_left[1],
                                lower_right[0], lower_right[1]]
    return grid_dict

def remove_outliers(df):
    """
    Identifica y elimina los outliers de los campos 'Altitude' y 'Velocity'
    """
    df = df[((df.Altitude - df.Altitude.mean()) / df.Altitude.std()).abs() < 3]
    return df[((df.Velocity - df.Velocity.mean()) / df.Velocity.std()).abs() < 3]

def locate_point(grid_x, grid_y, point_x, point_y):
    """
    Localiza la celda a la que pertenece un punto y devuelve sus índices
    """
    x_step = grid_x[1]-grid_x[0]
    y_step = grid_y[1]-grid_y[0]
    cell_x = ((point_x - grid_x[0])//x_step) + 1
    cell_y = ((point_y - grid_y[0])//y_step) + 1
    return cell_x, cell_y

def get_cell_num(cell_x, cell_y, n):
    """
    Devuelve el número de una celda dados sus índices X e Y
    """
    return (((cell_y - 1) * (n-1)) + cell_x)

```

```

def addCellToAll(df, n):
    """
    Función que itera sobre todas las entradas y añade la celda
    en la que se encuentra el vuelo en cada momento
    """
    df['Cell'] = ''
    for i, row in df.iterrows():
        point_x = row[2] # Longitude
        point_y = row[1] # Latitude
        cell_x, cell_y = locate_point(x_steps, y_steps, point_x, point_y)
        cell_num = get_cell_num(cell_x, cell_y, n+1)
        df.at[i, 'Cell'] = int(cell_num)
    return df

def group_by_cell_icao(df):
    """
    Para cada celda e ICAO, se obtiene la media de la velocidad y altitud
    """
    icao_count_df = df.groupby('Cell')['ICAO24'].count().reset_index()
    cell_group_df = df[['Cell', 'Velocity', 'Altitude']].groupby(['Cell'])
        .mean().reset_index()
    df = cell_group_df.merge(icao_count_df, on='Cell')
    return df

N_DIVISIONS = 100 # Número de divisiones horizontales y verticales

# lat > 35.512 AND lat < 44.512 AND lon > -10.415 AND lon < 5.054
x_steps = np.linspace(-10.415, 5.054, N_DIVISIONS + 1) # Longitude
y_steps = np.linspace(35.512, 44.512, N_DIVISIONS + 1) # Latitude

# Diccionario que contiene las coordenadas de cada celda
grid_dict = get_grid(x_steps, y_steps, N_DIVISIONS)

final_vuelos_df = pd.DataFrame()
for n_day in range(1,31):
    dataset_name = '/tmp/enero_' + str(n_day) + '.csv'
    print('> Preparando dataset', dataset_name)

    # Cargamos el dataset en un dataframe
    dia_vuelo_df = pd.read_csv(dataset_name)

    # Preparamos el dataframe haciendo una limpieza de campos
    dia_vuelo_df = preprocess_vuelos_df(dia_vuelo_df)

    # Eliminamos las entradas con valores atípicos en
    # los campos Altitude y Velocity
    dia_vuelo_df = remove_outliers(dia_vuelo_df)

```

```
# Asignamos una celda a cada entrada del dataframe
dia_vuelo_df = addCellToAll(dia_vuelo_df, N_DIVISIONS)

# Agrupamos las entradas por celda e ICAO y hacemos la media de velocidad
# y altitud para cada grupo
dia_vuelo_df = group_by_cell_icao(dia_vuelo_df)

# Creamos una nueva columna en el dataframe que contiene el día (número)
# al que se corresponde
dia_vuelo_df['Day'] = n_day

# Almacenamos este dataframe junto con el del resto de días
final_vuelos_df = final_vuelos_df.append(dia_vuelo_df)

# Reseteamos el índice
final_vuelos_df = final_vuelos_df.reset_index(drop=True)

# Ordenamos las columnas para mejor legibilidad
final_vuelos_df = final_vuelos_df[['Day', 'Cell', 'Velocity',
                                     'Altitude', 'ICAO24']]

# Guardamos el dataframe final como un archivo json y lo almacenamos en S3
final_vuelos_df.to_json(r'/tmp/final_vuelos_df.json')
s3.upload_file('/tmp/final_vuelos_df.json',
               BUCKET_NAME, 'opensky-data/final_flights_dataset.json')
```