TRABAJO DE FIN DE MÁSTER

Estudio y predicciones de calidad del aire en la ciudad de Madrid

MÁSTER EN INGENIERÍA Y CIENCIA DE DATOS
UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Estudiante: José Manuel Domínguez Martín

Tutor: Antonio Robles Gómez

JUNIO 2022

Resumen

Este trabajo fin de máster contiene toda la información del estudio realizado sobre los niveles de calidad del aire de la ciudad de Madrid y también la incidencia que puede tener el tráfico sobre los niveles de los distintos parámetros que determinan la calidad.

En una primera fase de trabajo se ha realizado un análisis previo de los conjuntos de datos que pone a disposición de los ciudadanos el ayuntamiento de Madrid. Una vez realizado el análisis exploratorio de los datos, se realiza un tratamiento sobre ellos para adecuarlos a las necesidades de los siguientes pasos que se realizan en el trabajo.

En la siguiente fase se realiza un trabajo de clasificación sobre los datos ya limpios y preparados en la fase anterior. Para ello se realizan estudios estadísticos sobre los datos, y se adecúan a las necesidades de los algoritmos de clasificación para trabajar de una manera óptima y obtener resultados mejores. Los algoritmos de clasificación utilizados son: Naive Bayes, KNN y Bosques Aleatorios.

Para finalizar, se han vuelto a analizar los datos y realizar los ajustes necesarios para su representación gráfica. Para ello se ha utilizado ATOTI y con ella se han implementado varios tipos de gráficos relevantes, donde se puede ver cómo se comportan los datos para sacar conclusiones de los mismos. Con varias de estas gráficas se ha preparado un cuadro de mandos para poder publicar los datos de una manera conjunta.

Palabras clave

Calidad del aire, tráfico, aprendizaje automático, clasificación, visualización de datos.

Abstract

This master's thesis contains all the information from the study carried out on air quality levels in the city of Madrid and, also, the impact that traffic can have on the levels of the different parameters that determine quality.

In a first phase of work, a preliminary analysis of the data sets made available to citizens by the Madrid City Council has been carried out. Once the exploratory analysis of the data has been carried out, a treatment will be carried out on them to adapt them to the needs of the following steps that are carried out in the work.

In the next phase, it is about carrying out a classification work on the data already cleaned and prepared in the previous phase. For this, statistical studies will be carried out on the data, adapting to the needs of the classification algorithms to work optimally and obtain better results. The classification algorithms used are next: Naive Bayes, KNN and Random Forests.

To finish, the data has been re-analyzed and the necessary adjustments have been made for its graphical representation, for this ATOTI has been used and with it several types of relevant graphs have been implemented, where you can see how the data behaves to draw conclusions from the same. With several of these graphs, a dashboard has been prepared to be able to publish the data jointly.

Keywords

Air quality, traffic, machine learning, classification, data visualization.

Índice de contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

Dentro de este capítulo se va a incluir la información sobre los antecedentes y puesta en contexto y también se compone de los subapartados de objetivos y motivación. En los apartados comentados se desarrollará la información que guarda relación con este Trabajo Fin de Máster (TFM).

1.1. Antecedentes y puesta en contexto

1.1.1. Trabajo origen

El origen de este TFM tiene como antecedente una práctica realizada en dicho Máster, para la asignatura Visualización de Datos. Partiendo de la base de la misma y de los conjuntos de datos que se analizaron en su momento, se amplía el ámbito de este trabajo, teniendo en cuenta un intervalo de tiempo horario y ampliando los datos del tráfico que en el anterior trabajo solamente se tenían en cuenta parámetros referentes a Calle30. Esto conlleva tener que realizar de nuevo el preprocesamiento de los datos, también se realizarán tareas de clasificación de Machine Learning (ML) para intentar predecir parámetros de calidad del aire teniendo en cuenta los datos del tráfico, para finalizar se mostrarán los datos en un cuadro de mando utilizando herramientas de código abierto y que permitan conectarse fácilmente a tecnologías Big Data.

El trabajo original puede consultarse en la url: https://tabsoft.co/391byAt, como se puede ver en la Figura *1* una vez en la web se podrá consultar distinta información relativa a la calidad del aire de la ciudad de Madrid. Se incluyen gráficos temporales en los que se observa la evolución de la contaminación dependiendo de la época de año, por trimestres o meses, también se muestra la información con respecto al día se la semana o del mes, incluso se puede ver un mapa con la posición de las estaciones de medida de la contaminación y se mostrará con distintos colores para indicar los niveles altos o bajos de los contaminantes que se miden.

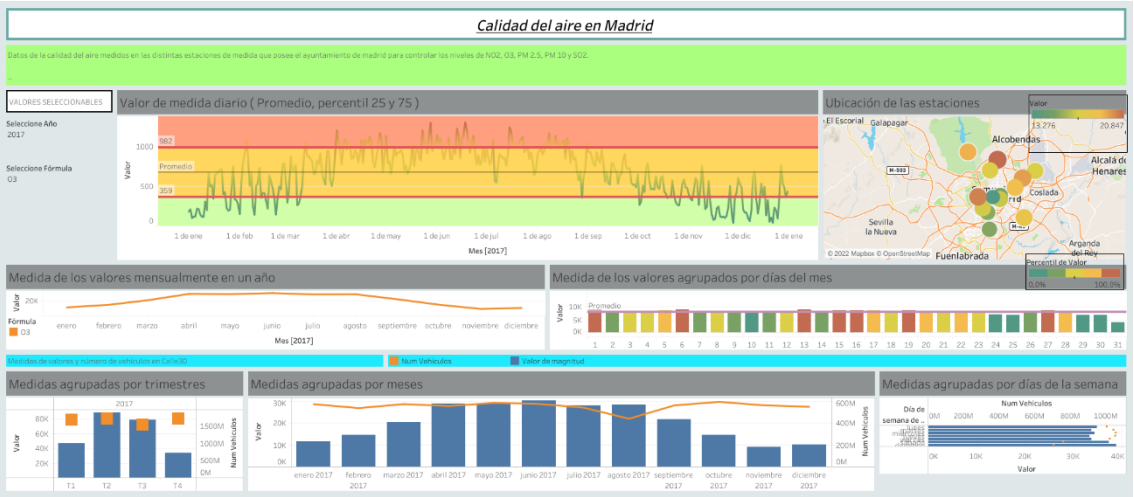

Figura 1.- Cuadro de mando origen de este TFM.

1.1.2. Situación actual

La calidad del aire es uno de los principales problemas en las grandes ciudades, durante los últimos años se están realizando estudios y propuestas para atajar el gran problema que existe actualmente. La Unión Europea está impulsando junto con los gobiernos de los Estados miembros unas normativas para crear Zonas de Bajas Emisiones (ZBE) en las ciudades con una población mayor de cincuenta mil habitantes, en estas zonas se realizarán restricciones al tráfico dependiendo de las emisiones de cada vehículo y de la calidad del aire en cada momento. Esta medida afecta hoy en día a 149 municipios españoles, por lo que durante los próximos años deben ir cumpliendo las nuevas leyes de movilidad sostenible a nivel europeo (Ley europea movilidad urbana, 2022) para evitar posibles multas.

1.1.3. Datos Abiertos

Este estudio no sería posible sin las fuentes de datos que desde hace unos años se van publicando por diversas administraciones y que pueden ser explotados libremente. Estos portales públicos ponen a disposición de los ciudadanos o de empresas conjuntos de datos que son recogidos en su mayor parte por sensores de manera automática y que se basan en los principios de transparencia, apertura de datos y la interacción entre administraciones y ciudadanos o empresas. Todo esto es gracias a que cada vez se intenta

tener ciudades más inteligentes (Smart Cities) y para conseguirlo es necesario tener cada vez un mayor número de sensores IoT que son los encargados de recolectar los datos que posteriormente serán compartidos en los portales de las distintas administraciones. En el caso particular de los datos utilizados para este estudio, se han utilizado fuentes de datos publicadas por el Excmo. Ayuntamiento de Madrid y que pueden ser utilizados libremente por cualquier persona. La ruta desde donde se pueden descargar los conjuntos de datos utilizados en este trabajo es (Datos Abiertos Madrid, 2022), en esta web se pueden obtener distintos tipos de datos y en varios formatos. Como ejemplo de otros tipos de datos, podemos obtener datos sobre la disponibilidad de los aparcamientos, datos meteorológicos, datos COVID, datos sobre accidentes de tráfico, etc. Todos estos datos sirven para ofrecer al ciudadano la posibilidad de explotar los datos para conocer con más detalle la información de la ciudad o para consultar algún tipo de información que considere relevante y que hasta hace unos años no estaba disponible y producía cierta desconfianza sobre la veracidad de alguna información trasladada por los dirigentes y que nos teníamos que creer sin poder contrastarlo en ningún lugar.

En la fecha que se realizó la redacción de este documento, en el portal de datos abiertos de la ciudad de Madrid se encontraban 530 fuentes de datos y es un número que está creciendo de manera exponencial en los últimos años. Todo esto se puede consultar en un catálogo de datos (Catálogo Datos Madrid, 2022) donde se muestra información sobre cada conjunto de datos y en el que se puede obtener la URL para la descarga, los tipos de ficheros en los que se puede descargar la información, el número de descargas que se han realizado, la periodicidad en la que se actualizan los datos, la fecha de la última actualización, etc.

1.2. Objetivos

La realización de este trabajo tiene una serie de objetivos principales y que tienen como origen varias de las asignaturas cursadas durante todo el Máster. Desde un principio se ha intentado aplicar gran parte del conocimiento obtenido durante el estudio.

Podemos definir dos objetivos principales, el primer objetivo es intentar ver la relación que existe entre los distintos parámetros del tráfico y la incidencia que tiene en los datos que intervienen en la calidad del aire de Madrid y utilizar distintos modelos

predictivos para poder predecir los momentos en los que la calidad de aire es mejor o peor dependiendo de las características del tráfico. El segundo es realizar una representación gráfica de los datos de tráfico y de medidas de calidad del aire para poder ver si realmente tienen una incidencia directa.

Una vez marcados los objetivos principales, vamos a indicar los distintos objetivos secundarios en los que nos vamos a basar para poder dar respuesta a los principales.

- Búsqueda y descarga de fuentes de datos como origen del trabajo.
- Análisis de los distintos conjuntos de datos con los que se va a trabajar.
- Preprocesamiento de los conjuntos de datos utilizando técnicas apropiadas para el tratamiento de grandes de conjuntos de datos (utilizando tecnologías BigData).
- Análisis de los conjuntos de datos una vez procesados antes de utilizarlos en los modelos predictivos y creación de las distintas clases.
- Entrenar distintos modelos de clasificación de inteligencia artificial.
- Analizar los resultados de los modelos de clasificación empleados.
- Representación gráfica de los conjuntos de datos, tanto de calidad del aire como del tráfico.

1.3. Motivación

La principal motivación para la realización de este trabajo es intentar aportar información que pudiera ser considerada de ayuda para tomar las medidas necesarias en referencia al tráfico y la incidencia del mismo en la calidad del aire de las grandes ciudades. En este estudio se ha tomado como referencia la ciudad de Madrid por varios motivos, comenzando por ser la ciudad más grande de España y tener uno de los peores datos de calidad de aire dentro de las ciudades europeas, también se ha tenido en cuenta la facilidad del acceso al gran volumen de datos que se genera continuamente, además hay que tener en cuenta que es una de las principales ciudades españolas con problemas con la fluidez del tránsito de vehículos.

Para poder dar solución a los problemas, uno de los factores fundamentales es realizar un estudio de los mismos, y poder determinar cuáles son los principales motivos causantes de dichos problemas. Por ello es una base para poder comprender la influencia

del tráfico en la calidad del aire es analizar los datos, realizar predicciones y realizar su representación gráfica y poder obtener toda la información que nos ofrecen estas herramientas.

1.4. Plan de proyecto

Teniendo en cuenta la carga de horas que se deben dedicar a la realización de este trabajo, se ha realizado una planificación de las tareas a realizar y de la carga de horas que supone cada una de ellas (ver Tabla *1*).

TAREAS	HORAS
• Realizar plan de proyecto.	30
• Analizar los datos y estudiar los preprocesados necesarios	30
• Preprocesar los datos	45
• Analizar, diseñar e implementar modelos predictivos	90
• Representación gráfica de los resultados de los modelos	20
• Comparar resultados de los modelos utilizados	12
• Representación gráfica de los datos origen	30
• Análisis de las representaciones gráficas	13
• Documentar las distintas tareas realizadas en el TFM	30
TOTAL	300

Tabla 1.- Planificación de horas en tareas a realizar en el TFM.

Una vez que se han estimado la carga de trabajo para cada una de las tareas, se ha realizado una planificación temporal para cuadrar el trabajo dentro de los tiempos marcados por el calendario del cuatrimestre. En él se puede ver la fecha inicio y fin de cada una de las tareas que se han planificado y que solamente serán realizadas por un estudiante (ver Tabla *2*).

TAREAS	Comienzo	Fin
Trabajo Fin de Máster	15/11/2021	16/05/2022
• Realizar plan de proyecto.	15/11/2021	23/11/2021
• Analizar los datos y estudiar los preprocesados necesarios	23/11/2021	01/12/2021
• Preprocesar los datos	01/12/2021	14/12/2021
• Analizar, diseñar e implementar modelos predictivos	14/12/2021	10/01/2022
• Representación gráfica de los resultados de los modelos	10/01/2022	17/01/2022
• Comparar resultados de los modelos utilizados	17/01/2022	21/01/2022
• Representación gráfica de los datos origen	21/01/2022	31/01/2022
• Análisis de las representaciones gráficas	31/01/2022	04/02/2022
• Documentar las distintas tareas realizadas en el TFM	04/02/2022	16/05/2022

Tabla 2.- Fechas de inicio y fin de las tareas.

También se muestra el diagrama de Gantt donde se pueden observar estas fechas de manera gráfica (ver Figura 2).

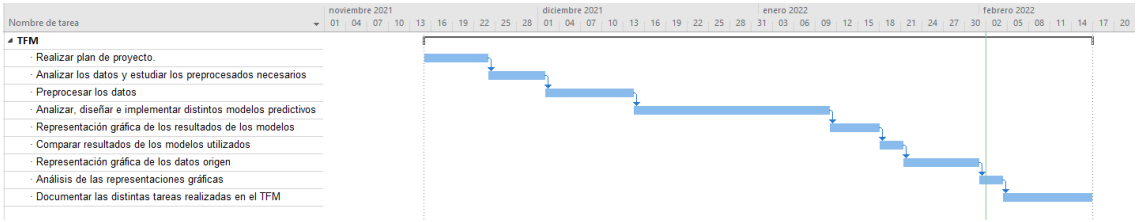

Figura 2.- Diagrama de Gantt – Planificación.

1.4.1. Recursos y costes

A continuación, se van a detallar los recursos utilizados (ver Tabla *3*), tanto de personal como de recursos hardware y software, para el desarrollo del trabajo.

Comenzamos indicando los recursos de personal utilizados. Ha sido necesario contar con el trabajo de un responsable de proyecto para realizar todas las tareas de gestión del proyecto, se ha necesitado a un analista, también se ha contado con el trabajo de un ingeniero de datos para poder realizar de una manera más eficiente todos los trabajos relacionados con la limpieza y el tratamiento de los datos y finalmente ha sido imprescindible contar con un desarrollador para implementar todo lo necesario en código Python.

Recurso	Horas	Coste/hora(euros)	Total(euros)
Ingeniero de Datos	83	50	4150
Analista	68	40	2720
Desarrollador	94	30	2820
Responsable Proyecto	55	55	3025
TOTAL	300		**12715**

Tabla 3.- Dedicación y coste de los recursos de personal utilizados.

Además de contar el personal indicado, ha sido necesario utilizar recursos hardware y software para poder completar el trabajo (ver Tabla *4*). Para ello se ha utilizado un ordenador de sobre mesa y sus periféricos, también ha sido necesario contar con una licencia de Word para realizar toda la documentación.

Componente	Coste(euros)
Ordenador + periféricos (PcCom_Bronze AMD Ryzen 5 3600)	967
Licencia Word	149
TOTAL	**1116**

Tabla 4.- Costes de elementos hardware y software.

El total del coste del proyecto teniendo en cuenta el total de los recursos utilizados (ver Tabla *5*).

Recursos	Coste(euros)
Recursos de personal	12715
Recursos de equipos	1116
TOTAL	**13831**

Tabla 5.- Total de costes del trabajo

1.5. Estructura de la memoria

Esta memoria está estructurada de la siguiente manera:

1. Introducción: Es el primer capítulo donde se realiza una puesta en contexto y se indican la motivación y los objetivos de dicho trabajo, también se incluye información sobre la planificación indicando los tiempos para cada

una de las tareas y también se incluye información de los costes asociados al proyecto.

2. Estado del arte: En este capítulo se incluye información sobre alguna de las leyes que regulan los parámetros de la calidad del aire, tanto a nivel europeo como estatal. Se hace referencia y se explican diversos estudios previos relacionados con la calidad del aire y la influencia del tráfico en estos parámetros. Incluye información de las distintas medidas que se tienen en cuenta para medir la calidad del aire. Dentro de este apartado también se explica el protocolo de para mejorar la calidad del aire propuesto por el ayuntamiento de Madrid, así como la distinta clasificación de los vehículos asignándolos una etiqueta según la expulsión de gases contaminantes.
3. Tecnologías empleadas: Se explica cada una de las tecnologías empleadas en el trabajo entrando en detalle del motivo por el que han sido elegidas comenzando por el tratamiento de los datos inicialmente, que se realiza con Apache Spark utilizando como lenguaje Python, continuando con todo lo que se refiere al aprendizaje automático, que todo esto se ha realizado con Python y la librería Scikit-learn y finalmente se comenta ATOTI que ha sido la herramienta utilizada para la representación gráfica y la realización de un cuadro de mandos para poder ver los distintos datos obtenidos del estudio, también se incluye un apartado donde se indican los equipos utilizados para realizar el trabajo.
4. Propuesta: Detallado de los trabajos realizados tanto a nivel del preprocesamiento de datos, donde se puede ver con detalle los distintos trabajos de tratamiento inicial, limpieza de datos y adaptación para poder trabajar de una manera adecuada en los puntos siguiente. Una vez que ya se tienen los datos preparados continuamos explicando toda la parte de aplicación de los modelos analizando cada uno de ellos en detalle y se finaliza el capítulo con lo relativo a la visualización de los datos, explicando las distintas gráficas utilizadas.
5. Evaluación y resultados: Se evalúan los resultados obtenidos en las distintas fases del proyecto, principalmente se hará una comparación más en detalle en lo que se refiere a los resultados obtenidos por los distintos algoritmos que se han utilizado de aprendizaje automático indicando las

virtudes y las carencias de cada uno de ellos teniendo en cuenta estos conjuntos de datos.

6. Conclusiones: Se presentan las conclusiones obtenidas durante la realización de este trabajo y se indican posibles trabajos futuros en los que se puede tomar como referencia parte del trabajo realizado.

2. Estado del arte

Dentro de este capítulo se intenta recoger información sobre estudios previos que dan origen a las bases que se han tenido en cuenta para realizar este trabajo.

2.1. Normativas

Como se ha comentado anteriormente, existen una serie de Directivas que regulan lo relativo a las mediciones de la calidad del aire y las medidas a tomar por los municipios afectados. La Directiva vigente es:

DIRECTIVA 2011/850/UE

Decisión de ejecución de la comisión de 12 de diciembre de 2011 por la que se establecen disposiciones para las directivas 2004/107/CE y 2008/50/CE del parlamento europeo y del consejo en relación con el intercambio recíproco de información y la notificación sobre la calidad del aire ambiente (2011/850/UE) (Europea, 2011).

A nivel estatal existen Reales Decretos creados en base a la Directiva Europea. El Real Decreto más reciente y que realiza modificaciones sobre los anteriores es:

Real Decreto 39/2017, de 27 de enero, por el que se modifica el Real Decreto 102/2011, de 28 de enero, relativo a la mejora de la calidad del aire (Estado, 2017).

2.2. Contaminantes

Los principales contaminantes que se tienen en cuenta para medir la calidad del aire son determinados gases y también partículas en suspensión.

Los gases a medir y que influyen en la calidad del aire son:

- **Nitrógeno**: en este apartado hacemos referencia a los gases que son formados por partículas de nitrógeno. Las principales fuentes de generación de estos gases son

debido al uso de combustibles fósiles que se usan para la movilidad en vehículos de combustión y en las calefacciones.

- **Azufre**: son los gases que contienen azufre y que son generados principalmente por el uso de gasoil tanto para la movilidad como para la generación de energía, también lo generan algunos tipos de calefacciones domésticas. El compuesto que vamos a estudiar en este trabajo es el Dióxido de Azufre ($S0_2$).
- **Ozono**: es un gas que normalmente se encuentra en muy bajas concentraciones, pero que en las ciudades suele formarse debido a la reacción química de los Óxidos de Nitrógeno y los Compuestos Orgánicos Volátiles (COV).
- **Óxidos de carbono**: estos gases se producen principalmente por los vehículos que producen Dióxido de Carbono (CO_2) y Monóxido de Carbono (CO) si se produce la combustión incompleta de combustibles orgánicos.

Las partículas que se tienen en cuenta para las medidas de calidad del aire son:

- **PM 10**: se refiere a las partículas que tienen un diámetro igual o inferior a 10 micras. Las principales fuentes que generan este tipo de partículas son las actividades industriales y los vehículos a combustión.
- **PM 2.5**: se refiere a las partículas que tienen un diámetro igual o inferior a 2.5 micras. Las principales fuentes que generan este tipo de partículas son las actividades industriales y los vehículos a combustión.

Tanto las PM10 como las PM2.5 pueden resultar de las reacciones químicas que suelen estar provocadas por gases como Dióxido de Azufre (SO_2), Óxidos de Nitrógeno (NO_x) y los compuestos orgánicos volátiles (COV).

Teniendo una mala calidad del aire en las ciudades se desencadenan una serie de perjuicios que afectan al clima, la salud, reduce la calidad de vida de los ciudadanos, perjudica a la vegetación de la zona, genera un deterioro de la imagen de la ciudad y crea un impacto económico negativo (ver Figura 3).

Figura 3.- Principales consecuencias de la elevada contaminación.

2.3. Efectos sobre la salud

Si hablamos sobre los efectos en la salud, las micropartículas y el ozono troposférico son los contaminantes que producen mayores problemas, la continua exposición a los mismos puede ocasionar consecuencias que van desde leves efectos en el sistema respiratorio a mortalidad debido a estas causas. Como se ha comentado anteriormente, el ozono no se emite directamente por la acción humana, es derivado de la reacción de los compuestos orgánicos volátiles (COV) y los óxidos de nitrógeno (NOx) en presencia de luz solar y los NO_X están fuertemente relacionados con la mano del hombre. Las micropartículas pueden emitirse directamente a la atmósfera (partículas primarias) o también pueden ser originadas (partículas secundarias) a partir de gases como los óxidos de nitrógeno (NOx) y el dióxido de azufre (SO2).

2.4. Evolución de la calidad del aire

Debido a las medidas que se están llevando a cabo por la Unión Europea, se puede ver una mejoría notable en la reducción de los contaminantes en las últimas décadas (Europea, 2011).

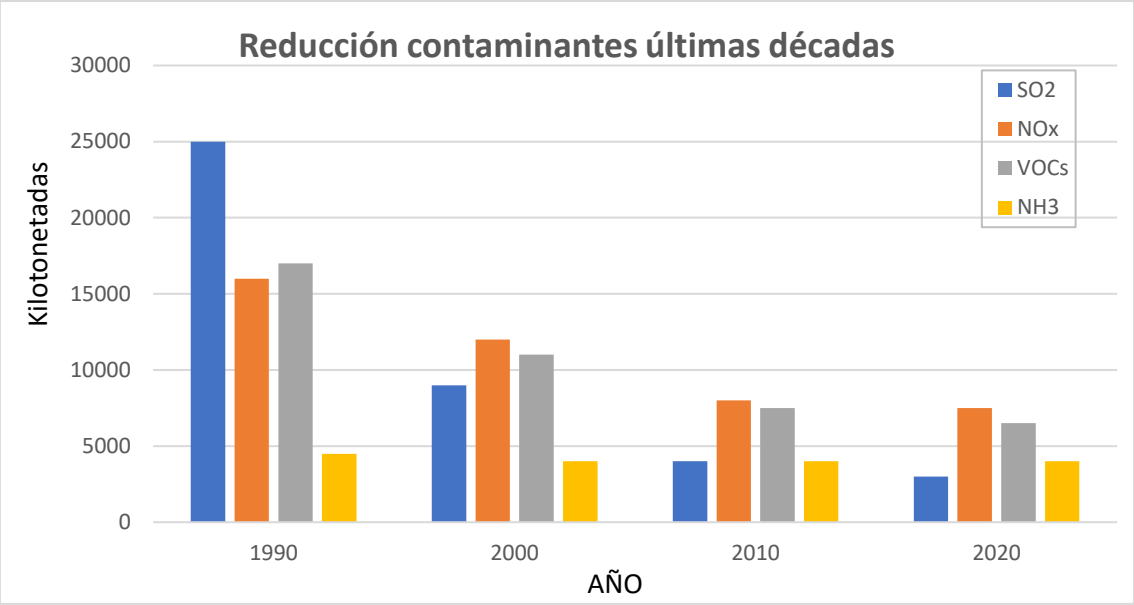

Figura 4.- *Evolución de los principales contaminantes en las grandes ciudades europeas*.

Esta evolución descendente es muy notable comparada con el 1990 (ver Figura 4), pero en los últimos años el descenso no es tan grande, por lo que es necesario seguir trabajando en este ámbito para continuar mejorando los niveles de calidad del aire. Para intentar buscar una solución al problema, se realizan estudios (Ec.europa.eu transporte, 2022) (Ec europa, 2022) para comprender mejor el problema y de esta manera poder definir una serie de medidas eficientes.

2.4.1. Restricciones Madrid

Para poder conseguir el objetivo de la reducción de los contaminantes en Madrid y de este modo poder tener una buena calidad del aire, lo que conlleva mejorar la salud de todas las personas que viven en la ciudad, se ha definido un plan llamado Madrid 360 que se compone de una serie de medidas, tanto para reducir el tráfico como para favorecer que el tráfico que exista sea con vehículos menos contaminantes (Madrid E. A., madrid.es, 2021). Dependiendo de un etiquetado realizado por la Dirección General de Tráfico (DGT), cada vehículo está catalogado con una pegatina que indica el nivel de contaminación que emite dicho vehículo (ver Figura 5).

Figura 5.- Catalogación de vehículos según la DGT (Tráfico, 2022)

Además de las pegatinas mostradas en la Figura 5, existe un determinado número de vehículo que por su antigüedad y por sus características no disponen de ninguna pegatina.

Existe un protocolo de contaminación que tiene en cuenta los niveles de NO_2 en la ciudad que contempla 5 escenarios distintos y que dependiendo de ellos limita el libre movimiento por la ciudad de los vehículos con determinadas etiquetas (Madrid E. A., 2021).

Escenario 1: El escenario 1 del protocolo de actuación para episodios de contaminación por NO_2 de la ciudad de Madrid se activa el día inmediatamente posterior a cuando las concentraciones de ese contaminante superan durante un día el nivel de preaviso, es decir, cuando en dos estaciones cualesquiera de una misma zona se superan los 180 µg/m3 durante dos horas consecutivas de forma simultánea o en tres estaciones cualesquiera de la red de vigilancia se superan los 180 µg/m3 durante tres horas consecutivas de forma simultánea.

Escenario 2: El escenario 2 del protocolo de actuación para episodios de contaminación por NO_2 de la ciudad de Madrid se activa el día inmediatamente posterior a cuando:

- Dos días consecutivos se supera el nivel de preaviso (en dos estaciones cualesquiera de una misma zona exceden los 180 µg/m3 durante dos horas consecutivas de forma simultánea o en tres estaciones cualesquiera de la red de vigilancia se superan los 180 µg/m3 durante tres horas consecutivas de forma simultánea).

- Un día el nivel de aviso (cuando dos estaciones cualesquiera de una misma zona superan, simultáneamente, 200 μg/m3 durante dos horas consecutivas, o tres estaciones cualesquiera de la red de vigilancia superan, simultáneamente, 200 μg/m3 durante tres horas consecutivas).

Escenario 3: El escenario 3 del protocolo de actuación para episodios de contaminación por NO_2 de la ciudad de Madrid se activa el día inmediatamente posterior a cuando:

- Tres días consecutivos se supera el nivel de preaviso (dos estaciones cualesquiera de una misma zona exceden los 180 μg/m3 durante dos horas consecutivas de forma simultánea o tres estaciones cualesquiera de la red de vigilancia superan los 180 μg/m3 durante tres horas consecutivas de forma simultánea).
- Dos días consecutivos el nivel de aviso (cuando dos estaciones cualesquiera de una misma zona superan, simultáneamente, 200 μg/m3 durante dos horas consecutivas, o tres estaciones cualesquiera de la red de vigilancia superan, simultáneamente, 200 μg/m3 durante tres horas consecutivas).

Escenario 4: El escenario 4 del protocolo de actuación para episodios de contaminación por NO_2 de la ciudad de Madrid se activa el día inmediatamente posterior a cuando durante cuatro días consecutivos se supera el nivel de aviso (dos estaciones cualesquiera de una misma zona superan, simultáneamente, 200 μg/m3 durante dos horas consecutivas, o tres estaciones cualesquiera de la red de vigilancia superan, simultáneamente, 200 μg/m3 durante tres horas consecutivas).

Escenario Alerta: El escenario de alerta del protocolo de actuación para episodios de contaminación por NO_2 de la ciudad de Madrid se activa el día inmediatamente posterior a cuando las concentraciones de ese contaminante superan el nivel de alerta (tres estaciones cualesquiera de una misma zona - o dos si se trata de la zona 4 - superan los 400 μg/m3 durante tres horas consecutivas de forma simultánea).

A parte de estos escenarios que restringen el tránsito de determinados vehículos cuando se dan determinadas condiciones, también existen actualmente 2 zonas que tienen limitaciones independientemente de los niveles de calidad del aire (ver Figura 6).

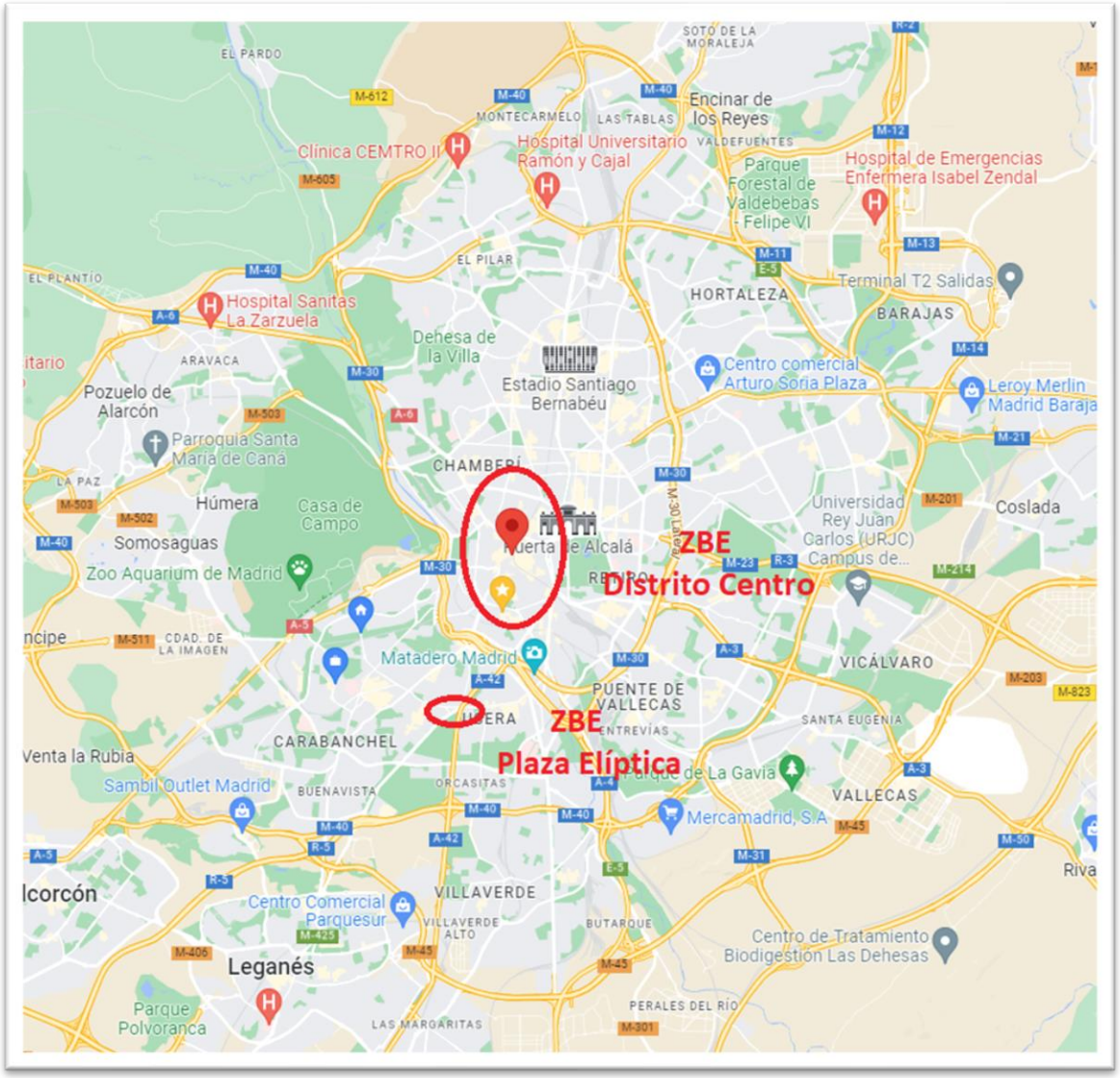

Figura 6.- Zonas de bajas emisiones en Madrid.

Los vehículos con etiqueta 0 no tienen ningún tipo de restricción para poder moverse libremente por la ciudad normalmente, ni en ninguno de los escenarios de contaminación.

Los vehículos con etiqueta ECO pueden circular sin restricciones en todos los escenarios y aparcar en zona SER salvo en escenario de alerta en el que se puede circular, pero no estacionar.

Los vehículos con etiqueta B y C tienen limitado el acceso a la zona de bajas emisiones del centro, excepto si van a estacionar en un aparcamiento de uso público. También tienen restricciones de circulación en la Calle30 y acceso al interior de la misma, dependiendo de los distintos escenarios.

Los coches sin etiqueta tendrán prohibido circular en todo el casco urbano, independientemente de si el propietario está o no empadronado en Madrid a partir del 1 de enero de 2025.

3. Tecnologías empleadas

Dentro de este capítulo se van a indicar las tecnologías empleadas para realizar cada una de las tareas principales.

3.1. Arquitectura

En este apartado se va a mostrar la arquitectura global y el flujo de datos y de los programas, desde la descarga de los conjuntos de datos hasta la representación gráfica de los mismos pasando por la ejecución de los datos por los modelos de clasificación.

Todo el proceso comienza con la descarga de las fuentes de datos abiertos, una vez descargados el proceso continúa tratando los datos de con Spark SQL obteniendo conjuntos de datos formateados tanto para el tratamiento de los datos con algoritmos de aprendizaje automático, como para la representación gráfica de los datos. Para la parte de aprendizaje automático, se utiliza la librería Scikit-learn y para la representación gráfica se utiliza ATOTI (ver Figura *7*).

Figura 7.- Arquitectura del TFM.

3.1.1. Herramientas

Para realizar el trabajo se ha utilizado equipo Hardware y Software. El Hardware utilizado es un ordenador personal, sin necesidad de ser específico para el trabajo realizado, pero para procesar más rápidamente lo datos se podría considerar un equipo más potente o incluso se podría tener en cuenta las distintas alternativas que se ofrecen en la nube y de esta forma se podría ajustar las necesidades de procesamiento del equipo al volumen del conjunto de datos, que va a seguir creciendo anualmente, y esto daría la posibilidad de ir modificando los parámetros dependiendo de las necesidades en cada momento.

- Equipo sobremesa:
    - Procesador: AMD Ryzen 5 3600
    - Memoria: 16 GB DDR4
    - Tarjeta gráfica: GeForce GTX 1650
    - Disco Duro: SSD 240GB
    - SSOO: Ubuntu 20.04

En lo que se refiere a Software, se han utilizado diversas herramientas que se han estudiado durante el Máster y que se ha aplicado el conocimiento adquirido para utilizarlo en este contexto específico:

- Software:
    - **Anaconda Navigator:** Interfaz gráfico de usuario que permite lanzar aplicaciones y administrar los paquetes, canales y entorno Conda sin la necesidad de utilizar línea de comandos. Dentro de este se han lanzado Jupyter Notebooks y JupyterLab.
    - **Spark 3.2:** Software de computación en clúster. Utilizada para el tratamiento de los datos.
    - **Python 3.7:** Lenguaje de alto nivel de programación interpretado.
    - **Atoti 0.6:** Software para representación gráfica y construcción de cuadros de mando.
    - **Word 2016:** Procesador de texto. El uso de este programa ha sido principalmente para realizar la documentación del trabajo.
    - **Notepad ++:** Editor de texto.

- **MS Project:** Software para realizar planificaciones de tareas, recursos y costes de un proyecto.
- **MS Excel:** Hoja de cálculo.

3.2. Preparación de los datos

Se ha comenzado revisando el volumen de los datos y era necesario utilizar alguna de las tecnologías específicas para el tratamiento de grandes volúmenes, se han tenido en cuenta las estudiadas en el curso y finalmente se ha utilizado Apache Spark, más concretamente Spark SQL.

3.2.1. Qué es Apache Spark

Podemos definir Apache Spark como un framework que realiza computación distribuida en clústeres, este framework ofrece una serie de acciones y transformaciones sobre colecciones de datos distribuidas a las que se denomina Resilient Distributed Dataset (RDDs). Una gran ventaja de Spark es que trabaja con los datos en memoria lo que hace que las operaciones realizadas sobre los mismos sean bastante rápidas, es tolerante a fallos, y optimiza la ejecución de los cálculos para utilizar solamente los necesarios, realiza la distribución de la mejor manera posible entre los nodos (Bill Chambers, 2018).

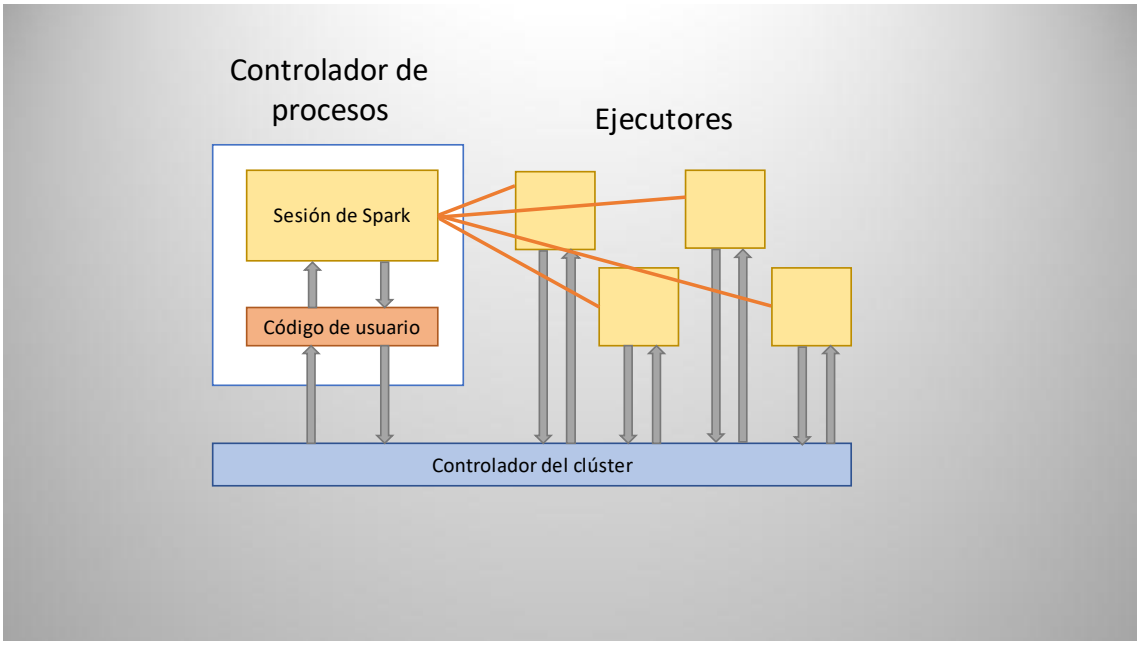

Figura 8.- Esquema del funcionamiento en clúster de Spark.

Spark utiliza un proceso para controlar las operaciones llamado SparkSession. La instancia de SparkSession es la forma en que Spark ejecuta manipulaciones definidas por el usuario en todo el clúster, por cada aplicación de Spark se debe crear una SparkSession (ver Figura *8*).

3.2.2. Spark SQL

Es la librería de Apache Spark para tratar datos estructurados de una forma eficiente y con un lenguaje similar a SQL, lo que suaviza bastante la curva de aprendizaje para un gran número de desarrolladores. Para trabajar con Spark SQL se cargan los datos en Dataframes y utiliza motores de búsqueda distribuidos. Los Dataframes son contenedores de datos distribuidos, pero tienen un formato similar a las tablas bidimensionales de las bases de datos relacionales, cuentan con una estructura de datos. Nos permite trabajar con tipos de datos anidados: mapa, matriz y estructura. Existe una API de Dataframes que proporciona una serie de operaciones relacionales a alto nivel.

Estos datos se pueden obtener desde muy diversas fuentes como ficheros *JSON*, ficheros *.csv*, *HIVE*, *RDBMS*, *Cassandra*, *etc*. (ver Figura *9*). Podríamos realizar un símil

con las bases de datos relacionales y diríamos que los Dataframes son equivalentes a una tabla en una base de datos relacional.

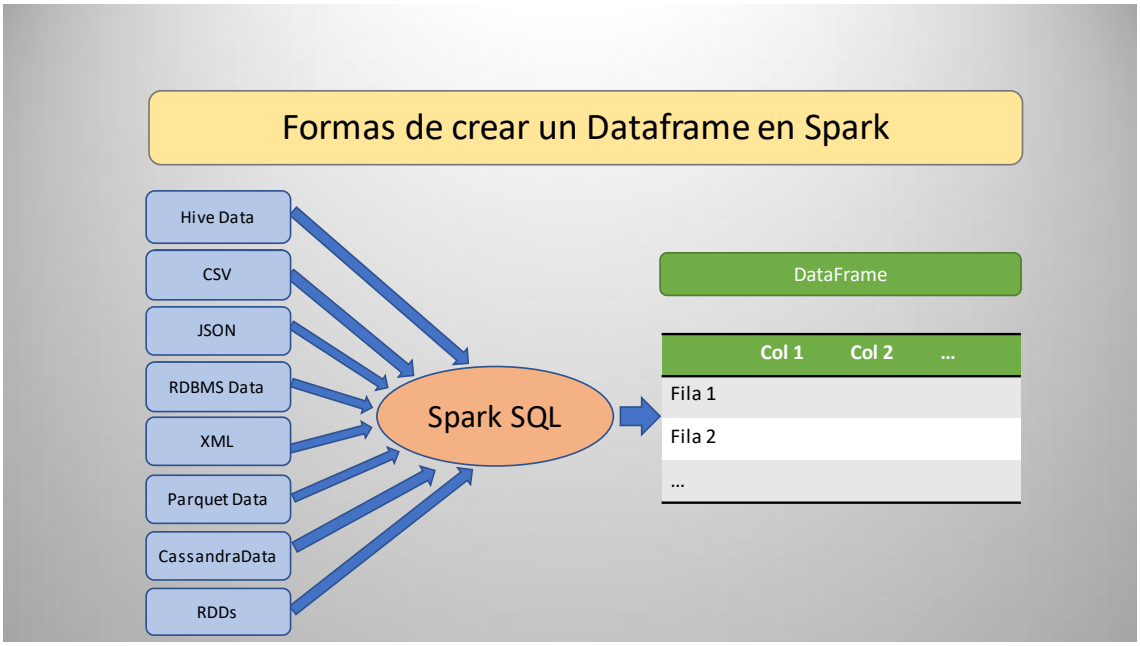

Figura 9.- Formas de crear un Dataframe en Spark.

Una vez que se encuentran cargados los datos en el Dataframe de Spark, ya se puede comenzar a realizar las operaciones necesarias sobre los datos. Estas operaciones tienen la ventaja que son realizadas sobre datos cargados en memoria, lo que supone una mayor eficiencia y velocidad de tratamiento de grandes volúmenes de datos.

3.3. Modelos de clasificación

Cuando ya se tienen unos conjuntos de datos preprocesados, se puede comenzar con el trabajo de analizar los datos y prepararlo para el proceso de ML.

3.3.1. ¿Qué es el aprendizaje automático?

Todo comienza intentando simular con máquinas el comportamiento humano, ésta va aprendiendo durante su vida diversas habilidades y que aplica continuamente para

realizar determinadas acciones. Partiendo de esta base, podríamos decir que el aprendizaje automático es la parte de la inteligencia artificial que se especializa en la creación de sistemas que son capaces de realizar predicciones en base a unos conjuntos de datos previos.

3.3.1.1. Tipos de modelos de aprendizaje

Podemos distinguir 3 categorías principales (ver Figura *10*):

- Aprendizaje supervisado: este tipo de algoritmos necesitan un aprendizaje anterior que se basa en un etiquetado que se asocia a unos datos y que de esta forma le permiten realizar las predicciones.
- Aprendizaje no supervisado: estos algoritmos no cuentan con un conocimiento previo, directamente toman el conjunto de datos y sin ningún tipo de referencia intentan obtener patrones que siguen determinados subconjuntos de datos.
- Aprendizaje por refuerzo: estos algoritmos tienen como característica que van realizando el aprendizaje en base a la experiencia que va obteniendo en base a decisiones anteriores, teniendo en cuenta distintos casos y realiza un proceso de prueba y error y va recompensando cuando toma buenas decisiones.

Figura 10.- Tipos de modelos de aprendizaje automático.

3.3.2. Tipos de algoritmos utilizados

Existen distintos tipos de algoritmos de aprendizaje automático y para este trabajo hemos utilizado algunos de ellos. En los siguientes apartados vamos a describir algunas de las características de ellos y los motivos que han llevado al uso de estos determinados algoritmos.

3.3.2.1. Naive Bayes

- **Teorema de Bayes**

Existen distintos tipos de estos algoritmos que se basan en el teorema de Bayes y para este trabajo se han utilizado todas las distintas implementaciones que existen en la librería Scikit-learn en la v1.0.2.

Todos estos algoritmos se apoyan en el teorema de Bayes, que fue una proposición planteada por el matemático británico Thomas Bayes publicada en 1763. Este teorema describe la probabilidad condicional de un evento basándose en realizar un conocimiento previo de las condiciones que podrían estar relacionadas con dicho evento (ver Figura 11).

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Figura 11.- Teorema de Bayes.

- **Algoritmos Naive Bayes en Scikit-learn**

En la librería de aprendizaje automático de Scikit-learn existen un conjunto de algoritmos de aprendizaje supervisado que se basan en la aplicación del Teorema de Bayes con la suposición de independencia condicional entre cada una de las características dado un valor de la variable de clase. Si desarrollamos el Teorema de Bayes teniendo en cuenta varias características podemos decir que dada una variable de clase ***y*** con un vector de características $x_i \ldots x_n$ la fórmula a aplicar sería (ver Figura 12):

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Figura 12.- Naive Bayes en Scikit-learn.

Los distintos algoritmos de esta familia de Naive Bayes se diferencian principalmente por las suposiciones que se realizan respecto a la distribución de P ($x_i \mid y$).

A pesar de parecer algoritmos demasiados simplificados, los clasificadores de Naive Bayes han funcionado bastante bien en muchas de las situaciones del mundo real, sobre todo en las aplicaciones de clasificación de documentos y el filtrado de spam en correos electrónicos. Con un pequeño conjunto de datos de entrenamiento, ya son suficientes para poder estimar los parámetros necesarios para obtener buenas predicciones. Una de las virtudes de estos algoritmos es que, comparándolos con otros algoritmos para funciones similares, son realmente rápidos.

- **Gaussian NB en Scikit-learn**

La implementación gaussiana de los algoritmos de Naive Bayes para la clasificación, se basa en el Teorema de Bayes pero teniendo la particularidad que supone que la probabilidad de las características es gaussiana. Teniendo en cuenta esto, podemos ver que la fórmula a aplicar por este algoritmo es la siguiente (ver Figura 13):

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Figura 13.- Gaussian Naive Bayes en Scikit-learn.

- **Multinomial NB en Scikit-learn**

La implementación del algoritmo *Multinomial* de Naive Bayes para la clasificación es una implementación para datos distribuidos multinomialmente, es una de las variantes utilizada principalmente para la clasificación de textos. La distribución está parametrizada por vectores $\Theta_i = (\Theta_{y1}, \ldots, \Theta_{yn})$ para cada clase y, donde n es el número de características y Θ_{y1} es la probabilidad $P(x_i \mid y)$ de característica *i* que aparece en una muestra perteneciente a la clase *y*.

Los parámetros Θ_y se estima mediante una versión suavizada de máxima verosimilitud, es decir, conteo de frecuencia relativa (ver Figura *14*):

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Figura 14.- Multinomial Naive Bayes en Scikit-learn - máxima verosimilitud.

Donde $N_{yi} = \sum_{x \epsilon T} x_i$ es el número de veces que la característica *i* aparece en la muestra de la clase *y* en el conjunto de entrenamiento *T*, y $N_y = \sum_{i=1}^{n} N_{yi}$ es el recuento total de todas las características de la clase *y*.

- **Complement NB en Scikit-learn**

La implementación del algoritmo *Complement* de Naive Bayes para la clasificación es una adaptación del algoritmo Multinomial, que tiene un buen comportamiento cuando el conjunto de datos es desequilibrado. Entrando en detalle, este algoritmo utiliza estadísticas del complemento de cada clase para calcular los pesos del modelo. Los inventores de este método muestran empíricamente que las estimaciones de los parámetros son más estables que las del algoritmo Multinomial. Regularmente *Complement* supera a Multinomial en tareas de clasificación de textos.

- **Bernoulli NB en Scikit-learn**

Bernoulli NB implementa algoritmos de clasificación y entrenamiento para datos que se distribuyen de acuerdo a distribuciones multivariadas de Bernoulli, lo que quiere decir que pueden existir múltiples características, pero se supone que cada una de las variables solamente puede tomar valores binarios. Por este motivo, esta implementación requiere que las muestras se representen como si fuesen vectores de características solamente con valores binarios. En los casos que los tipos de datos que las características no sean binarios, el algoritmo los transforma a datos binarios.

La regla de decisión se basa en la siguiente fórmula (ver Figura *15*):

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

Figura 15.- Bernoulli Naive Bayes en Scikit-learn – regla de decisión.

Este algoritmo comparado con los otros de la misma familia suele funcionar mejor cuando el conjunto de datos más cortos. Como sucede con los otros algoritmos, su principal uso es para el procesamiento de documentos, aunque no es nuestro propósito,

entendemos que para otros trabajos también podrían funcionar bien y además hay que tener en cuenta que son bastante rápidos.

3.3.2.2. K vecinos más cercanos (KNN)

El algoritmo K vecinos más cercanos está basado en instancia y es de tipo supervisado dentro de los algoritmos de aprendizaje automático. Se puede utilizar, tanto para clasificación como para regresión. Es un método bastante sencillo de comprender, por lo que se puede utilizar para dar los primeros pasos dentro del mundo del aprendizaje automático. Principalmente se utiliza para clasificar los valores buscando los puntos de datos lo más similares posibles teniendo en cuenta la cercanía, utilizando los resultados obtenidos durante la etapa de entrenamiento y aplicar lo que el algoritmo aprende durante esta etapa sobre qué criterios aplicar para comprobar si los puntos se encuentran a poca distancia.

Cuando hablamos que es basado en instancia, queremos decir que no aprende explícitamente, lo que hace es memorizar las distintas instancias de entrenamiento que se utilizan como base del conocimiento para aplicar posteriormente en la fase de predicción.

Si entramos en detalles sobre el funcionamiento, lo primero que debemos tener en cuenta es el significado de la K del nombre y quiere decir el número de vecinos que vamos a tener en cuenta para determinar la clase que vamos a asignar al punto que estamos evaluando, una vez explicado el significado de la K podemos indicar que se dividen en 3 fases principales:

1. Realiza el cálculo de la distancia entre el punto a clasificar y todos los demás puntos que tenemos en el conjunto de datos de entrenamiento (ver Figura *16*).

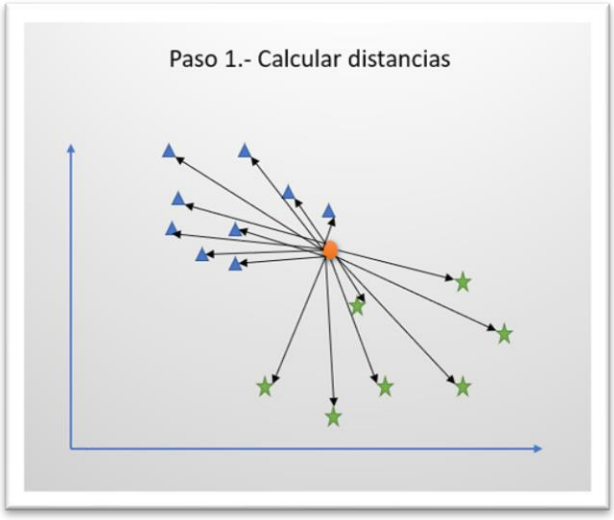

Figura 16.- KNN - calcular distancias.

2. Se seleccionan los K puntos que se encuentran más cerca del punto a evaluar (ver Figura *17*).

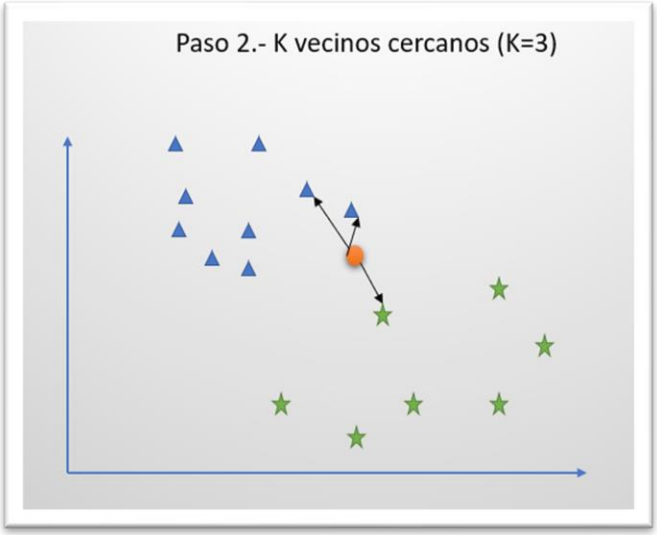

Figura 17.- KNN - K vecinos cercanos (K=3).

3. Comprobamos a que clase pertenecen cada uno de ellos y la clase mayoritaria será la que se asigne al punto a clasificar (ver Figura *18*).

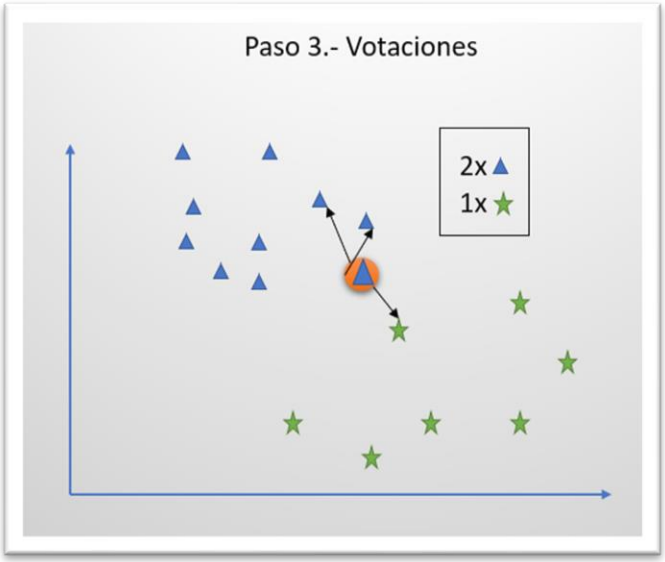

Figura 18.- KNN – Votaciones.

El valor de K puede cambiar bastante el funcionamiento del algoritmo, tendrá mucha más relevancia en los puntos que se encuentren cerca de las fronteras de las clases. Lo lógico es que el valor de K sea un valor impar para los desempates, si fuese un valor par los empates podrían suceder con normalidad. También hay que tener en cuenta que a más puntos K, el algoritmo necesitará más tiempo para obtener los resultados.

Otro de los puntos a tener en cuenta cuando se utiliza este algoritmo, es la manera de medir la distancia entre los puntos. Normalmente se suele utilizar la distancia euclídea para determinar una medida de cercanía entre dos puntos, aunque hay que tener en cuenta que existen otras alternativas a utilizar, como puede ser la similitud del coseno, que mide el ángulo de los vectores teniendo en cuenta el menor para indicar que está más cerca.

Por indicar uno de los inconvenientes principales es que cada vez que se quiere calcular a que clase pertenece un punto, tenemos que realizar los cálculos sobre todos los puntos del conjunto de datos, lo que supone una carga de trabajo bastante grande y que requiere mucha memoria y capacidad de procesamiento. Por este motivo, como es comprensible, este algoritmo tiene unos tiempos de proceso asumibles cuando el conjunto de datos no es muy grande y cuando no se tienen muchas características.

3.3.2.3. Bosques aleatorios

El algoritmo de clasificación de bosques aleatorios es del tipo de aprendizaje supervisado, también se pueden utilizar para tareas de regresión. Es un algoritmo fácil de usar y muy flexible. Fue desarrollado por Leo Breiman y Adele Cutler, se compone de varios árboles de decisión que en el conjunto crean el bosque aleatorio.

Los árboles de decisión son algoritmos de aprendizaje supervisado que se utilizan para resolver problemas tanto de clasificación como de regresión. Se llaman árboles de regresión cuando la variable dependiente es continua y árboles de clasificación cuando es de tipo cualitativo. La idea principal es ir realizando un esquema con múltiples bifurcaciones que se anidan en forma de árbol hasta llegar a las hojas, que son el último nivel. Estas hojas determinan la predicción de la clase que dicho algoritmo asocia a la variable dependiente en los casos de clasificación que son los relevantes para este trabajo. Según se han ido pasando niveles se ha ido redirigiendo por las ramas del árbol teniendo en cuenta que propiedades cumplía en cada uno de los nodos y dependiendo de ellas se envía por un camino del árbol o por otro.

Como ejemplo, podríamos realizar uno para determinar el descuento en una tienda web a aplicar a los clientes dependiendo de determinadas condiciones, podríamos tener en cuenta el número de productos que van a comprar, si el valor se encuentra entre 1 y 5, nos dirigiremos a la parte izquierda del árbol, si es mayor que 5, nos dirigimos a la derecha. Una vez que nos encontramos en el segundo nivel, vamos a tener valorar el importe del pedido, si nos encontramos en la parte izquierda del árbol, si el pedido es inferior a 100 €, obtendremos un descuento del 5%, si el precio es igual o superior a 100 €, el descuento que se aplicará será del 10%, si por el contrario nos hemos ido por la rama derecha del segundo nivel, para pedidos con un importe inferior a 100 €, obtenemos un descuento del 10% y si es igual o superior a 100 €, el descuento sería del 25% (ver Figura *19*).

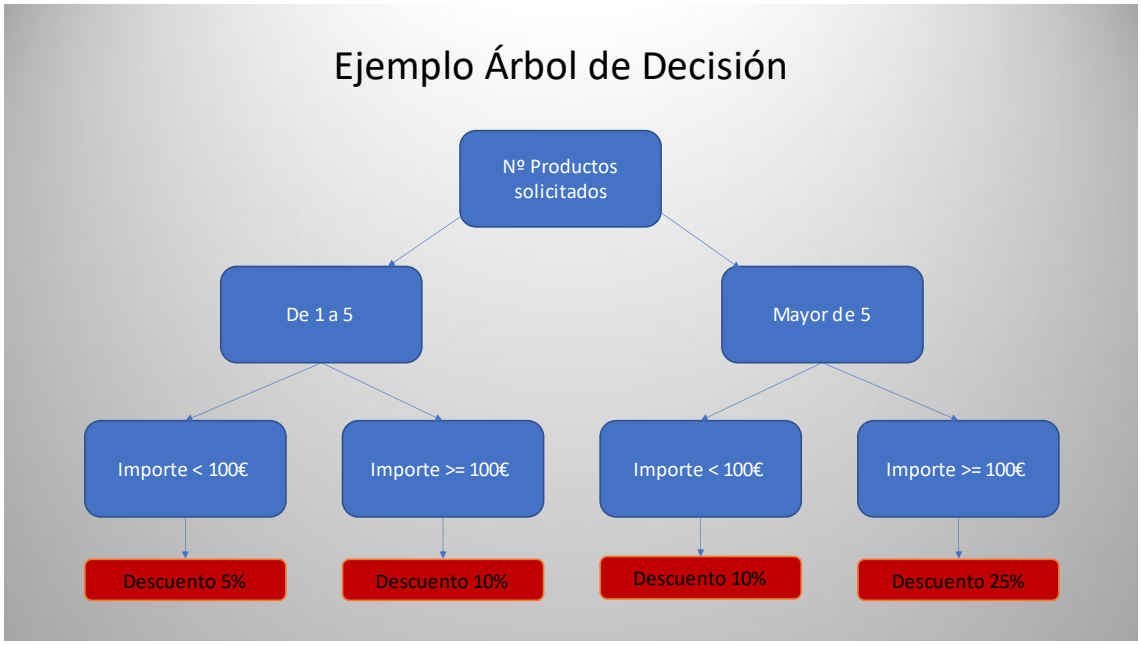

Figura 19.- Árbol de decisión.

Una vez hecha una introducción sobre los árboles de decisión, podemos explicar con más detalle que son y como trabajan los bosques aleatorios. Como se ha comentado anteriormente, un bosque aleatorio es simplemente un conjunto de árboles de decisión que entre todos forman el bosque, pero estos serán combinados utilizando *bagging*, esto significa que cada uno de los árboles va a realizar su trabajo con distintas porciones de datos. Cada uno de los árboles realiza las tareas necesarias para determinar la clase a la que pertenece cada dato que es necesario evaluar para realizar la predicción y una vez recorrido el árbol, se da una respuesta de la clase que predice dicho árbol. Cuando ya se tienen las predicciones de cada uno de los árboles que componen el bosque aleatorio, se tienen en cuenta los votos para cada una de las clases existentes y la respuesta del algoritmo de bosques aleatorios será la mayoritaria entre la suma de todos los votos (ver Figura *20*).

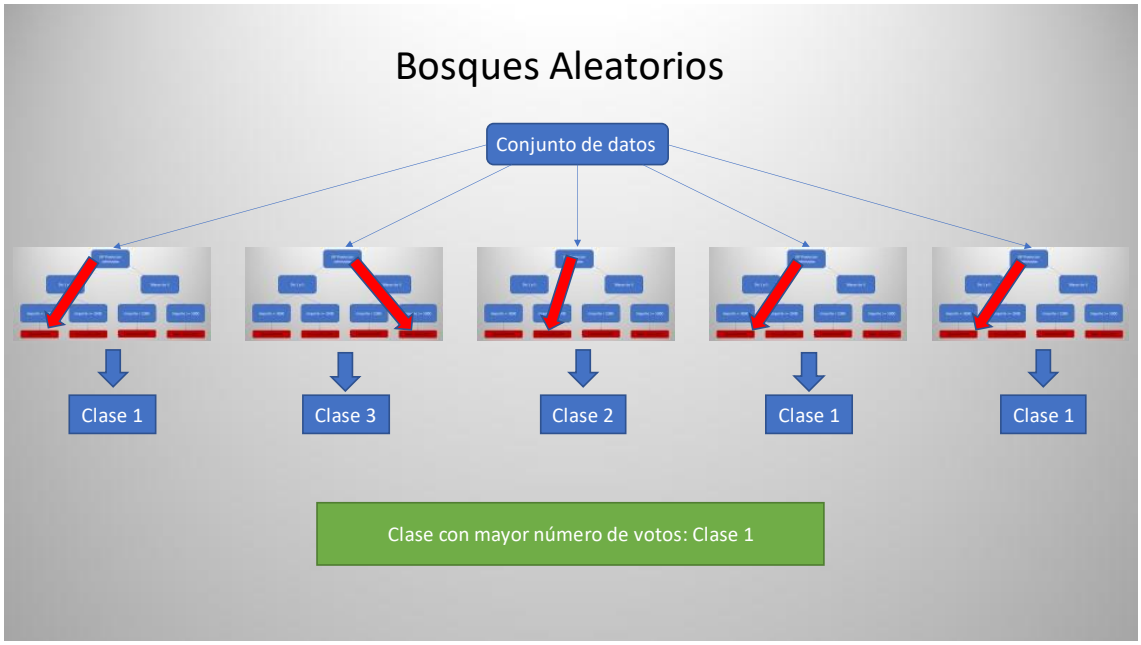

Figura 20.- Bosques aleatorios (con 5 árboles de decisión).

3.4. Representación gráfica

La visualización del trabajo origen de este proyecto se realizó en Tableau y para realizar la nueva visualización se ha utilizado ATOTI que es una herramienta de software libre y ofrece una mejor integración con tecnologías de backend Big Data.

En un principio se había pensado seguir utilizando Tableau para realizar el trabajo, pero hay que tener en cuenta que como siempre existen pros y contras a la hora de decantarse por una herramienta. Esta herramienta se ha expandido de una manera muy rápida debido a la facilidad de uso, casi cualquier persona puede obtener unos trabajos bastante buenos en poco tiempo, también podemos decir que funciona con múltiples tipos de datos, ya sean estructurados o no estructurados.

Ventajas Tableau

- Tiene una gran capacidad para crear muy buenas visualizaciones de datos con una alta calidad.

- La facilidad de uso es una de las principales ventajas, por lo que de esta forma existen numerosos usuarios que pueden trabajar con la herramienta de manera eficiente.
- Es un entorno robusto y confiable, puede funcionar satisfactoriamente incluso en trabajos de Big Data, por lo que destaca su alto rendimiento.
- Posee una comunidad y foros amplia que permite consultar y resolver dudas fácilmente.

Inconvenientes Tableau

- A pesar de existir licencias gratuitas, si se necesitan realizar trabajos de calidad es necesario comprar licencias para que puedas aprovechar al máximo todas sus funcionalidades.
- En versión gratuita para estudiantes no permite realizar consumo de datos en tiempo real.

Teniendo en cuenta todo esto, y sobre todo el gran inconveniente de no poder realizar consumo de datos en tiempo real, pensando en posibles trabajos futuros, se determinó utilizar ATOTI que podemos considerarlo como una herramienta lo suficientemente buena como para poder cubrir las necesidades de este trabajo y más teniendo en cuenta que actualmente se encuentra creciendo de manera exponencial.

3.4.1. ATOTI

La visualización de los datos se puede realizar utilizando otras librerías sobre los Jupyter Notebook, pero ATOTI ofrece la posibilidad de eliminar algunas de las limitaciones que tienen, como pueden ser que no se permite la visualización interactiva, pandas Dataframe no permiten el manejo de grandes volúmenes de datos o que no se soporta nativamente el análisis multidimensional (ATOTI_a, 2020). Se podrían utilizar otras opciones que no tienen las limitaciones indicadas (aplicaciones Bussines Inteligence (BI), Databricks, cubos OnLine Analytical Processing (OLAP), …), el problema viene de la integración entre estas herramientas. Por lo tanto, ATOTI es la manera más fácil de

integrar el uso de Jupyter Notebooks utilizando como lenguaje Python y poder manejar cubos OLAP (ver Figura 21) (ATOTI_b, 2020).

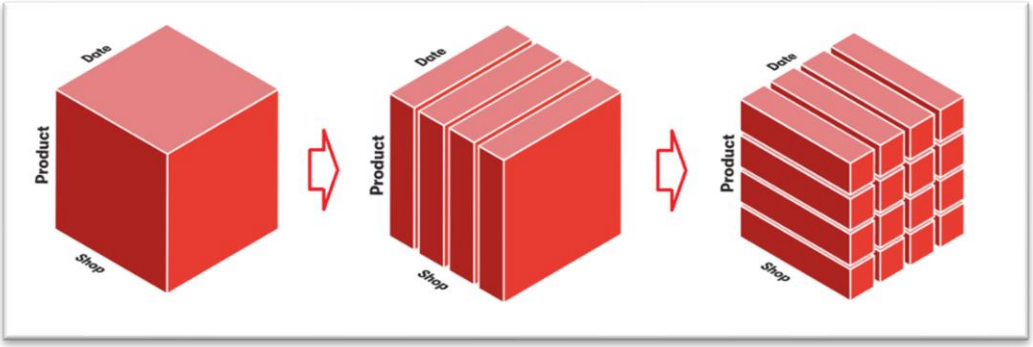

Figura 21.- Ejemplos de cubos OLAP *(ATOTI_a, 2020)*

Los beneficios que se consiguen integrando cubos OLAP en los Notebooks son:

- ✓ Permite un análisis multidimensional y la posibilidad de visualizar varios escenarios.
- ✓ Da la posibilidad de tener visualizaciones totalmente dinámicas.
- ✓ Los cuadros de mando resultantes se pueden compartir fácilmente con otros usuarios para permitir la colaboración entre ellos.

El motor de agregación está desarrollado en Java y está integrado con las tecnologías Python de una forma optimizada para no perder ninguna de las capacidades. Partiendo de esta base se desarrolla ATOTI, que se empaqueta y se distribuye en una librería de Python que los usuarios pueden descargar y utilizar de la misma manera que lo hace con cualquier otra librería.

Para empezar a trabajar con esta librería una vez instalada, hay que ejecutar JupiterLab y crear una sesión de ATOTI en el código (ver Figura 22), partiendo de esa sesión ya iremos creando todas las estructuras necesarias para poder cargar los datos y montar el cubo OLAP. Cuando ya tenemos creada la sesión, podemos cargar el conjunto de datos partiendo de ficheros (*.csv, .json, .xml, …*) directamente o de ficheros que ya se encuentran cargados en pandas Dataframe.

```
# importaciones necesarias
import atoti as tt

session = tt.create_session()
```

Figura 22.- Creación de sesión ATOTI

Cuando ya se encuentran los datos cargados en la sesión, podemos crear el cubo OLAP pasando como parámetros los datos y como opcional el nombre que vamos a asignar al cubo. Una vez creado el cubo, se puede comenzar a trabajar con sus jerarquías, sus niveles y sus medidas. Con estas tres características podemos combinarlas para obtener y representar gráficamente los datos.

4. Propuesta

En este apartado se van a explicar los distintos trabajos que se han realizado con los datos de origen para poder realizar los trabajos de aprendizaje automático y la representación gráfica en el cuadro de mandos.

4.1. Conjuntos de datos y preprocesamiento

Comenzamos explicando los conjuntos de datos origen y posteriormente explicaremos las transformaciones que se han realizado para ofrecer el formato adecuado. Estos ficheros de datos se pueden obtener en distintos formatos, para este trabajo se ha considerado descargar los datos en formato *.csv*. Se obtienen desde la web de datos abiertos del Ayuntamiento de Madrid (Datos Abiertos Madrid, 2022).

4.1.1. Datos del tráfico de Madrid

El ayuntamiento de Madrid pone a disposición de cualquier ciudadano una serie de datos sobre el tráfico que son facilitados por las distintas estaciones de medida que se encuentran distribuidas por toda la ciudad y que facilitan datos con un intervalo de 15 minutos.

Se comienza realizando la descarga de los ficheros de datos del tráfico, se compone de dos tipos distintos, el primero de ellos tiene todos los datos de medida del tráfico.

CAMPO	**DESCRIPCIÓN**
ID	Identificador del punto de medida del tráfico.
FECHA	Fecha de cuando se ha realizado la medición.
TIPO_ELEM	Tipo de punto de medida (Calle30 (M30) o no pertenecen a la Calle30 (URB)).
INTENSIDAD	Intensidad del Punto de Medida en el periodo de 15 minutos (vehículos/hora). Un valor negativo implica la ausencia de datos.

OCUPACIÓN	Tiempo de Ocupación del Punto de Medida en el periodo de 15 minutos (%). Un valor negativo implica la ausencia de datos.
CARGA	Carga de vehículos en el periodo de 15 minutos. Parámetro que tiene en cuenta intensidad, ocupación y capacidad de la vía y establece el grado de uso de la vía de 0 a 100. Un valor negativo implica la ausencia de datos.
VMED	Velocidad media de los vehículos en el periodo de 15 minutos (Km./h). Sólo para puntos de medida interurbanos M30. Un valor negativo implica la ausencia de datos.
ERROR	Indicación de si ha habido al menos una muestra errónea o sustituida en el periodo de 15 minutos. N: no ha habido errores ni sustituciones; E: los parámetros de calidad de alguna de las muestras integradas no son óptimos; S: alguna de las muestras recibidas era totalmente errónea y no se ha integrado.
PERIODO_INTEGRACIÓN	Número de muestras recibidas y consideradas para el periodo de integración.

Tabla 6.- Descripción de campos de los datos de tráfico.

Una vez descargados los datos, configuramos el equipo para lanzar Apache Spark bajo un Jupyter Notebook y desde este realizaremos todo el preprocesamiento de datos aprovechando las virtudes de Spark para trabajar de forma paralela sobre Big Data.

Una vez lanzado el Jupyter Notebook, se genera el SparkContext y la SparkSession, se realiza la carga de los datos desde el fichero .csv hasta un Dataframe de Spark. Cuando ya lo tenemos cargado, vamos a mostrar el esquema de la estructura de datos cargada para comprobar que se ha realizado de forma correcta (ver Figura 23 y Figura 24).

```
df.printSchema()

root
 |-- id: integer (nullable = true)
 |-- fecha: string (nullable = true)
 |-- tipo_elem: string (nullable = true)
 |-- intensidad: integer (nullable = true)
 |-- ocupacion: double (nullable = true)
 |-- carga: integer (nullable = true)
 |-- vmed: double (nullable = true)
 |-- error: string (nullable = true)
 |-- periodo_integracion: integer (nullable = true)
```

Figura 23.- Esquema del Dataframe de tráfico.

- ✓ El siguiente paso es realizar el filtrado de los datos que tienen error de lectura, por lo que ejecutamos el comando de filtrado de Spark SQL sobre el campo error a "N".
- ✓ Para comprobar el volumen de datos que tenemos realizamos un *count* y obtenemos que después del filtrado el número de registros es 131 647 948.
- ✓ Continuamos eliminando alguna de las columnas que contienen datos que no van a ser necesarios para el aprendizaje automático, por lo que quitamos las columnas: *error* y *periodo_integración*.
- ✓ Debemos tener en cuenta que los ficheros de calidad del aire solamente tienen datos en las horas en punto y los del tráfico cada 15 minutos, por lo que es necesario eliminar los datos que no vamos a poder cruzar con los ficheros de calidad del aire. Para ello ha sido necesario sobre el campo fecha, separar la fecha de la hora y crear una columna para guardar estos datos, y también se ha separado de la hora, los minutos y se ha creado una columna con los minutos. Con esta última columna de minutos, hemos filtrado todos los que no tienen datos de hora en punto '00' y con esta operación nos hemos quedado solamente con los datos de horas en punto.

```
%%time
#separamos el día y la hora para filtrar por las horas en punto
df_t_19_sepFecha = df_t_19_deleteCol.withColumn('dia', split(df_t_19
_deleteCol['fecha'], ' ').getItem(0)) \
       .withColumn('hora', split(df_t_19_deleteCol['fecha'], ' ').ge
tItem(1))

#separamos los minutos
df_t_19_sepMin = df_t_19_sepFecha.withColumn('min', split(df_t_19_se
pFecha['hora'], ':').getItem(1))

#nos quedamos con las horas en punto
df_t_19_HoraEnPunto = df_t_19_sepMin.filter((col('min') == '00'))
df_t_19_HoraEnPunto.show()
```

Figura 24.- Tratamiento de datos del tráfico - filtrado de datos en horas en punto.

Al ejecutar el código anterior, podremos ver en el Jupyter Notebook el resultado del contenido que se muestra en el Dataframe con los datos de las horas en punto (ver Figura 25).

```
+----+------------------+---------+--------+-----+----+---------+
|  id|fecha|inten|ocup|carga|vmed|tipo_elem|dia|hora|min|
+----+------------------+---------+--------+-----+----+---------+
|1001|2019-01-01 00:00:00|2340|11.0|0|63.0|M30|2019-01-01|00|00|
|1001|2019-01-01 01:00:00|2340|11.0|0|63.0|M30|2019-01-01|01|00|
|1001|2019-01-01 02:00:00|2340|11.0|0|63.0|M30|2019-01-01|02| 00|
|1001|2019-01-01 03:00:00|2340|11.0|0|63.0|M30|2019-01-01|03| 00|
|1001|2019-01-01 04:00:00|2340|11.0|0|63.0|M30|2019-01-01|04| 00|
|1001|2019-01-01 05:00:00|2340|11.0|0|63.0|M30|2019-01-01|05| 00|
|1001|2019-01-01 06:00:00|2340|11.0|0|63.0|M30|2019-01-01|06| 00|
|1001|2019-01-01 07:00:00|2340|11.0|0|63.0|M30|2019-01-01|07| 00|
|1001|2019-01-01 08:00:00|2340|11.0|0|63.0|M30|2019-01-01|08| 00|
|1001|2019-01-01 09:00:00|2340|11.0|0|63.0|M30|2019-01-01|09| 00|
|1001|2019-01-01 10:00:00|2340|11.0|0|63.0|M30|2019-01-01|10| 00|
|1001|2019-01-01 11:00:00|2340|11.0|0|63.0|M30|2019-01-01|11| 00|
|1001|2019-01-01 12:00:00|2340|11.0|0|63.0|M30|2019-01-01|12| 00|
|1001|2019-01-01 13:00:00|1020|3.0|0|50.0|M30|2019-01-01|13| 00|
|1001|2019-01-01 14:00:00|2376|8.0|0|57.0|M30|2019-01-01|14| 00|
|1001|2019-01-01 15:00:00|1968|6.0|0|62.0|M30|2019-01-01|15| 00|
|1001|2019-01-01 16:00:00|1176|3.0|0|63.0|M30|2019-01-01|16| 00|
|1001|2019-01-01 17:00:00|1164|3.0|0|55.0|M30|2019-01-01|17| 00|
|1001|2019-01-01 18:00:00|2304|7.0|0|60.0|M30|2019-01-01|18| 00|
|1001|2019-01-01 19:00:00|3096|12.0|0|55.0|M30|2019-01-01|19| 00|
+----+------------------+---------+--------+-----+----+---------+
----------+--------+---+
only showing top 20 rows

CPU times: user 3.31 ms, sys: 0 ns, total: 3.31 ms
Wall time: 168 ms
```

Figura 25.- Datos del tráfico - Filtrado de los datos en horas en punto.

El segundo fichero contiene los datos de las estaciones de medida, datos que se refieren a su identificador y posicionamiento.

Descripción de los campos de posicionamiento de las estaciones de medida del tráfico (ver Tabla *7*):

CAMPO	DESCRIPCIÓN
TIPO_ELEM	Tipo de punto de medida (M30 o URB).
DISTRITO	Indica el distrito donde se encuentra el punto de medida.
ID	Identificador del punto de medida.
COD_CENT	Código del punto de medida.
NOMBRE	Nombre del punto de medida.
UTM_X	Punto de localización X del sistema Universal Transverse Mercator (UTM).

UTM_Y	Punto de localización Y del sistema UTM.
LONGITUD	Coordenada de longitud del punto de medida.
LATITUD	Coordenada de latitud del punto de medida.

Tabla 7. -Descripción datos de posicionamiento de los puntos de medida del tráfico.

Al igual que se ha realizado con el fichero de los datos de medidas del tráfico, se realiza una carga de este fichero en un Spark Dataframe y en este caso solamente es necesario quedarnos con las columnas que indican la identificación y el posicionamiento de cada uno de los puntos de medida, estos datos son los que tenemos en las columnas ID, LONGITUD y LATITUD.

Cuando ya hemos realizado todas estas transformaciones con los datos origen del tráfico, guardamos los datos de nuevo en formato *.csv*, para ellos utilizamos la operación *write* de los Spark Dataframe que realiza esta operación con una tarea MapReduce que optimiza el trabajo, por lo que la salida que vamos a tener son una serie de ficheros *.csv*, que han sido generados por cada uno de los procesos paralelos que ha realizado Spark y también tendremos un fichero SUCCESS que indica el resultado del trabajo.

4.1.2. Datos de calidad del aire de Madrid

El ayuntamiento de Madrid también pone a disposición de cualquier ciudadano una serie de datos sobre la calidad del aire que son facilitados mediante mediciones que se realizan cada hora en las estaciones de medida que tiene distribuidas por toda la ciudad.

Se comienza descargando los ficheros de los datos horarios de calidad del aire en formato *.csv*, se tiene un fichero por cada mes del año, dado el gran volumen de datos, sobre todo con respecto al tráfico, se ha descargado lo referente al año 2019 que ha sido el último año antes del COVID y que no tiene datos que se viesen alterados por las distintas situaciones vividas durante este tiempo.

En este apartado vamos a trabajar con dos conjuntos de datos que prepararemos limpiando o modificando los datos y realizaremos la unión de ellos para tener solamente un fichero de datos que se refiere a la calidad del aire. El primer fichero es el que se refiere a los datos de las medidas que se realizan en las estaciones.

Descripción de los campos de los datos de calidad del aire (ver Tabla *8*):

CAMPO	DESCRIPCIÓN
PROVINCIA	Provincia donde se encuentra situada la estación de medida.
MUNICIPIO	Municipio donde se encuentra situada la estación de medida.
ESTACION	Identificador de la estación de medida.
MAGNITUD	Magnitud a la que se refieren los valores de medida.
PUNTO_MUESTREO	Incluye información relativa al punto de muestreo.
ANO	Año en el que se está realizando la medición.
MES	Mes en el que se está realizando la medición.
DIA	Día en el que se está realizando la medición.
H01	Dato de la primera hora del día.
V01	Indica si el valor medido en la primera hora es válido o no (V o F).
H02	Dato de la segunda hora del día.
…	…
H23	Dato de la última hora del día.
V23	Indica si el valor medido en la última hora es válido o no (V o F).

Tabla 8.- Descripción de campos de los datos de calidad del aire.

El segundo fichero contiene los datos de las estaciones de medida, datos que se refieren a su nombre, posicionamiento, localización, tipos de gases que se miden en ella, etc.

Descripción de los campos de posicionamiento de las estaciones de calidad del aire (ver Tabla *9*):

CAMPO	DESCRIPCIÓN
CODIGO	Código de identificación de la estación de medida.
CODIGO_CORTO	Código corto de identificación de la estación de medida.
ESTACION	Indica el nombre de la estación.
DIRECCION	Dirección donde se encuentra situada la estación.
LONGITUD_ETRS89	Longitud de la estación en el sistema geodésico European Terrestrial Reference System 1989 (ETRS89).
LATITUD_ETRS89	Latitud de la estación en el sistema geodésico ETRS89.
ALTITUD	Altitud a la que se encuentra sobre el nivel del mar.
COD_TIPO	Código del tipo de estación.

NOM_TIPO	Nombre del tipo de estación.
NO2	Indica con una X si la estación posee medidor de NO2.
SO2	Indica con una X si la estación posee medidor de SO2.
CO	Indica con una X si la estación posee medidor de CO.
PM10	Indica con una X si la estación posee medidor de PM10.
PM2_5	Indica con una X si la estación posee medidor de PM25.
O3	Indica con una X si la estación posee medidor de O3.
BTX	Indica con una X si la estación posee medidor de BTX.
COD_VIA	Código de la vía donde se encuentra la estación.
VIA_CLASE	Clase de la vía donde se encuentra la estación.
VIA_PAR	
VIA_NOMBRE	Nombre de la vía donde se encuentra la estación.
Fecha alta	Fecha de alta de la estación.
COORDENADA_X_ETRS89	Coordenada X de la estación en el sistema geodésico ETRS89.
COORDENADA_Y_ETRS89	Coordenada Y de la estación en el sistema geodésico ETRS89.
LONGITUD	Coordenada de longitud de la estación.
LATITUD	Coordenada de latitud de la estación.

Tabla 9.- Descripción de campos de los datos de los puntos de medida de calidad del aire.

Una vez que ya se han presentado los conjuntos de datos en crudo, vamos a describir los distintos tratamientos que se han realizado sobre los ellos.

Al igual que con el procesamiento de los datos del tráfico, comenzamos levantando en entorno de Spark y ejecutando el Jupyter Notebook desde Spark. Continuamos creando el SparkContext y el SparkSession, que serán necesarios para poder realizar las tareas de preprocesado de los datos sobre los Spark Dataframes.

Cargamos le fichero que contiene la medida de todos los datos de calidad del aire en el Spark Dataframe:

- ✓ Con estos datos, observamos que es necesario rellenar con ceros a la izquierda los campos hora y día, para tener el mismo formato que los datos del tráfico.
- ✓ Para obtener el campo fecha concatenamos los datos de año, mes y día y creamos una nueva columna con el dato resultado.

- ✓ Realizamos un *select* teniendo en cuenta todas las columnas menos año, mes y día, debido a que ya tenemos este dato en la columna fecha.

Como podemos observar en el conjunto de datos, vienen agrupados todos los datos de las lecturas de cada hora del día en un solo registro, por lo que vamos a crear un Dataframe por cada una de las 24 horas del día, tendrá solamente los datos de la hora a la que haga referencia y se renombrarán las columnas H01, H02, etc., por la columna VALOR y lo mismo se realizará con las columnas V01, V02, etc., por la columna VALIDADO, también crearemos otra columna con el dato de la hora y se llamará HORA. Concatenamos estos 24 Dataframe y tenemos el resultado deseado. En los datos de origen solamente hay una fila por día y vamos a pasar a tener 24 filas por día, cada una con un registro horario (ver Figura 26).

```
df_aire_19_completo.show(3)


+--------+--------+----------+-----+--------+--------+
|ESTACION|MAGNITUD|     FECHA|VALOR|VALIDADO|    HORA|
+--------+--------+----------+-----+--------+--------+
|       4|       1|2019-10-01|00010|       V|01:00:00|
|       4|       1|2019-10-02|00012|       V|01:00:00|
|       4|       1|2019-10-03|00008|       V|01:00:00|
+--------+--------+----------+-----+--------+--------+
only showing top 3 rows
```

Figura 26.- Calidad del aire - Conversión a un registro por cada hora.

Una vez que ya tenemos los datos en este formato, realizamos las siguientes operaciones (ver Figura *27*):

- ✓ Filtramos los datos que tienen el campo VALIDADO a N.
- ✓ Eliminamos la columna VALIDADO.
- ✓ Transformamos la columna fecha de String a Datetime.

```
%%time
#cambiamos el tipo de fecha de string a date
func =  udf (lambda x: datetime.strptime(x, '%Y-%m-%d'), DateType())

df_aire_19_fecha_date = df_aire_19_sin_Val.withColumn('FECHA_FORMAT',
func(col('FECHA')))
```

Figura 27.- Calidad del aire - transformación campo fecha de string a datetime.

- ✓ Creamos dos columnas nuevas con los datos del día de la semana, tanto numéricamente como con el nombre.

4.1.3. Fusión de datos de calidad del aire y tráfico

Una vez que ya tenemos realizado un primer preprocesamiento para poder realizar la unión de los datos, vamos a realizar las siguientes operaciones para unirlos y tener los datos en un solo Dataframe.

Continuamos cargando el fichero de posicionamiento de las estaciones de calidad del aire. Para cruzar los datos de calidad del aire y de tráfico se ha tenido en cuenta que el tráfico que hay en el norte de la ciudad tiene muy poca influencia a corto plazo en las estaciones de medida de calidad del aire que se encuentran en el sur. Por lo tanto, se han realizado los cálculos necesarios para poder ver la distancia que existe entre cada estación de calidad del aire y todos los puntos de medida del tráfico. Para ello se ha utilizado la fórmula de *haservine* (ver Figura 28 y Figura 29) (Es-academic.com, 2022) que nos permita realizar el cálculo entre dos puntos teniendo en cuenta la longitud y latitud de cada uno de ellos.

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Figura 28.- Fórmula de Haservine - distancia entra dos puntos *(Macalupu.com, 2022)*.

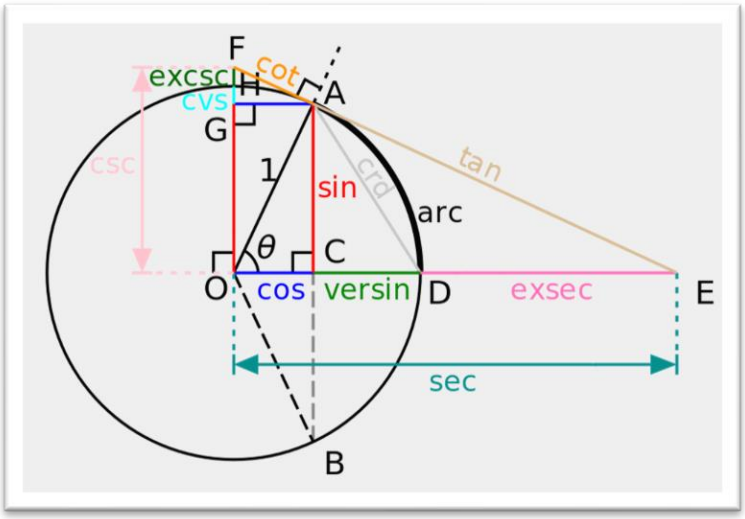

Figura 29.- Representación gráfica - fórmula de Haservine *(Community.esri.com, 2022)* .

Esta fórmula se ha aplicado dentro de una función que tiene como entrada los datos de latitud y longitud de dos puntos distintos y devuelve la distancia en kilómetros entre ellos (ver Figura *30*).

```
#Calcular distancia entre dos puntos con latitud y longitud
def dist_entre_2puntos(la1, lo1, la2, lo2):
    # radio de la tierra en kms
    R = 6373.0

    lat1 = radians(la1)
    lon1 = radians(lo1)
    lat2 = radians(la2)
    lon2 = radians(lo2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distancia = R * c

    return round(distancia, 3)
```

Figura 30.- Función que calcula la distancia en metros entre dos puntos

Con esta función, ya podemos ver la distancia entre los puntos de medida del tráfico y las estaciones de calidad del aire. Para ello, se ha creado otra función para tomar cada uno de los puntos de tráfico que se encuentran en un Dataframe, calculamos su estación más cercana y guardaremos en una columna el identificador de la estación y en otra columna los metros de distancia a los que se encuentran (ver Figura 31).

```
#calcula la estación de med de calidad del aire más cercana a cada punto de medida del tráfico
def dis_entre_medtraf_medcal (df_mtraf, df_mcal):
    num_rows_df_mtraf = len(df_mtraf.index)
    num_rows_df_mcal = len(df_mcal.index)
    df_mtraf['ESTACION_CAL'] = ""
    df_mtraf['DISTANCIA_EST_CAL'] = ""

    #recorremos todo el df de mtraf
    for i in range(num_rows_df_mtraf):

        la1 = df_mtraf.iloc[i]['latitud']
        lo1 = df_mtraf.iloc[i]['longitud']

        dist = 100000
        pos = 0
        #recorremos todo el df de mcal
        for j in range(num_rows_df_mcal):
            la2 = df_mcal.iloc[j]['LATITUD']
            lo2 = df_mcal.iloc[j]['LONGITUD']
            d = dist_entre_2puntos(la1, lo1, la2, lo2)
            if d < dist:
                estac = df_mcal.iloc[j]['ESTACION']
                dist = d

        df_mtraf.loc[[i],'ESTACION_CAL'] = int(estac)
        df_mtraf.loc[[i],'DISTANCIA_EST_CAL'] = dist
    return df_mtraf
```

Figura 31.- Función calcula estación más cercana por punto del tráfico.

Una vez que ya tenemos unidos los datos de posicionamiento de tráfico y calidad del aire, vamos a realizar la unión entre estos y los datos de medida del tráfico, de esta forma unificamos los datos en un mismo Dataframe para poder trabajar posteriormente en el aprendizaje automático. Realizamos la unión de los Dataframes teniendo en cuenta el campo "id" de las estaciones de medida del tráfico, que será el campo "id" en el Dataframe de los datos de medida del tráfico y "id_pos" en los datos de posicionamiento.

En este momento ya tenemos solamente 2 Dataframe, uno de tráfico (ver Figura 33) y otro de calidad del aire (ver Figura 32) medidos en la ciudad de Madrid y para el año 2019.

```
root
 |-- ESTACION: string (nullable = true)
 |-- MAGNITUD: string (nullable = true)
 |-- FECHA_FORMAT: date (nullable = true)
 |-- HORA: string (nullable = false)
 |-- DIA_SEM: integer (nullable = true)
 |-- DIA_SEM_NOMBRE: string (nullable = true)
 |-- VALOR: string (nullable = true)
```

Figura 32.- Estructura del Dataframe de calidad del aire.

```
root
 |-- id: integer (nullable = true)
 |-- dia: string (nullable = true)
 |-- hora_t: string (nullable = true)
 |-- intensidad: integer (nullable = true)
 |-- ocupacion: double (nullable = true)
 |-- carga: integer (nullable = true)
 |-- vmed: double (nullable = true)
 |-- tipo_elem: string (nullable = true)
 |-- longitud: double (nullable = true)
 |-- latitud: double (nullable = true)
 |-- ESTACION_CAL: integer (nullable = true)
 |-- DISTANCIA_EST_CAL: double (nullable = true)
```

Figura 33.- Estructura del Dataframe del tráfico.

Partiendo de estas estructuras, podemos realizar la unión de los dos Dataframes teniendo como parámetro de unión los siguientes campos “ESTACION” y “ESTACION_CAL”, “FECHA_FORMAT” y “fecha_format_t” y también “HORA” y “hora_t”.

Finalizamos todas estas operaciones realizando un *select* para quedarnos solamente con las columnas necesarias y tendremos una estructura de datos adecuada para los siguientes trabajos sobre los datos (ver Figura *34*).

```
%%time
df_aire_trafico.show(3)


+--------+--------+-----------+--------+----+---+---+---+-------+-
-------------+-----+----+---------+-------------------+-----------
----+----------+---------+-----+----+-----------------+
|ESTACION|MAGNITUD|FECHA_FORMAT|    HORA|ANIO|MES|DIA| HH|DIA_SEM|D
IA_SEM_NOMBRE|VALOR|  id|tipo_elem|           longitud|         lat
itud|intensidad|ocupacion|carga|vmed|distancia_est_cal|
+--------+--------+-----------+--------+----+---+---+---+-------+-
-------------+-----+----+---------+-------------------+-----------
----+----------+---------+-----+----+-----------------+
|       4|       1|  2019-01-01|09:00:00|2019| 01| 01| 09|      3|
Tuesday|00011|1009|      M30|  -3.72490924272642| 40.416233663841|
828|      1.0|    0|68.0|            1.368|
|       4|       1|  2019-01-01|09:00:00|2019| 01| 01| 09|      3|
Tuesday|00011|1010|      M30|   -3.7250433138461|40.4163123231285|
636|      1.0|    0|54.0|            1.372|
|       4|       1|  2019-01-01|09:00:00|2019| 01| 01| 09|      3|
Tuesday|00011|1011|      M30|-3.7230756918931496| 40.418233915259|
480|      1.0|    0|57.0|            1.111|
+--------+--------+-----------+--------+----+---+---+---+-------+-
-------------+-----+----+---------+-------------------+-----------
----+----------+---------+-----+----+-----------------+
only showing top 3 rows

CPU times: user 31.4 ms, sys: 4.17 ms, total: 35.6 ms
Wall time: 1min 36s
```

Figura 34.- Contenido del resultado de la unión de todos los conjuntos de datos.

Cuando ya tenemos los datos en un solo dataset, vamos a separarlos por tipos de magnitud. Esta separación es para poder tratar los distintos conjuntos de datos para algunos de los tipos de gases más nocivos para la salud. Dentro de los datos tenemos distintas magnitudes (ver Tabla 10), en este estudio nos vamos a centrar en cuatro que pueden considerarse de las que más influyen en la calidad del aire.

Magnitud	Descripción	Fórmula
1	Dióxido_de_Azufre	SO2
6	Monóxido_de_Carbono	CO
7	Monóxido_de_Nitrógeno	NO
8	Dióxido_de_Nitrógeno	NO2
9	Partículas2.5µm	PM2.5
10	Partículas10µm	PM10
12	Óxidos_de_Nitrógeno	NOx
14	Ozono	O3

Tabla 10.- Magnitudes de calidad del aire

De todas estas vamos a tratar los datos de NO_2, PM2.5, PM10 y O3. Guardamos los datos de cada una de las medidas en un *.csv* para poder recuperar los datos por separado.

4.2. Análisis exploratorio

Una vez que tenemos el conjunto de datos agrupado, es necesario realizar un análisis exploratorio. Se trata de realizar una serie de tareas para conocer la base de los datos que tenemos, entenderemos y evaluaremos para tratar de encontrar y resolver casos atípicos o incorrectos y de esta forma evitamos vulnerabilidades en los datos que indujesen a error en las tareas de aprendizaje automático.

Según se indica en la web de Ministerio de Asuntos Económicos y Transformación Digital, podemos definir el Análisis Exploratorio de Datos (AED) de la siguiente manera:

"El AED consiste en aplicar un conjunto de técnicas estadísticas dirigidas a explorar, describir y resumir la naturaleza de los datos, de tal forma que podamos garantizar su objetividad e interoperabilidad." (Digital M. d., 2022).

El AED se puede dividir en una serie de fases que se realizan sobre los datos (ver Figura 35):

- ✓ Análisis descriptivo.
- ✓ Ajuste de tipos de variables.
- ✓ Detección y tratamiento de datos ausentes.
- ✓ Identificación de datos atípicos.
- ✓ Correlación de variables.

Figura 35.- Fases de un AED *(Digital M. d., 2022)*.

Como el guardado de los datos se ha realizado utilizando la función .to_csv de Spark Dataframes, esta función utiliza internamente una tarea MapReduce, por lo que nos encontramos con los datos guardados en una carpeta y dentro de la misma tenemos una lista de ficheros *.csv*, vamos a realizar una lectura de todos los ficheros *.csv* que tenemos en la carpeta y los concatenaremos para tener todos los datos agrupados en un Dataframe.

Para realizar la clasificación del aprendizaje automático es necesario tener un campo con distintas clases para los valores, por lo que vamos a realizar una clasificación en cinco niveles por cada una de las magnitudes, con esto resultará que agruparemos los datos dependiendo del valor medido y dependiendo de la magnitud que se trate. Cuando hayamos realizado esta operación, podemos eliminar la columna VALOR y la columna MAGNITUD.

Comenzamos realizando un estudio de los tipos de datos y comprobamos si se pueden reducir los tipos para ocupar menos espacio en disco, para ellos se han creados dos funciones para comprobar si los datos enteros y los datos con decimales se ajustan adecuadamente a los tipos donde se están guardando (ver Figura *36* y Figura 37).

```
# Creamos una funcion que tiene como entrada un valor min y max y devuelve
# el tipo min para guardar los datos
def ajustar_tipos_int(min, max):
    if min >= 0:     # en los casos que no tenga negativos buscamos tipos sin signo
        if max < np.iinfo(np.uint8).max:                # comprobamos que el
                                                        #max no sea mayor de
                                                        # lo que acepta uint8
            tipo_optimo = np.uint8
        elif max < np.iinfo(np.uint16).max:             # comprobamos que el
                                                        #max no sea mayor de
                                                       #lo que acepta uint16
            tipo_optimo = np.uint16
        elif max < np.iinfo(np.uint32).max:             # comprobamos que el
                                                        #max no sea mayor de
                                                       #lo que acepta uint32
            tipo_optimo = np.uint32
        else:                           # si el max es mayor de lo que acepta uint32
                                       #el tipo adecuado es uint64
            tipo_optimo = np.uint64
    else:            # en los casos que tenga negativos buscamos tipos con signo
        if min > np.iinfo(np.int8).min and max < np.iinfo(np.int8).m
ax: # comprobamos el max y el min del tipo
            tipo_optimo = np.int8
        elif min > np.iinfo(np.int16).min and max < np.iinfo(np.int16
).max: # comprobamos el max y el min del tipo
            tipo_optimo = np.int16
        elif min > np.iinfo(np.int32).min and max < np.iinfo(np.int32
).max: # comprobamos el max y el min del tipo
            tipo_optimo = np.int32
        elif min > np.iinfo(np.int64).min and max < np.iinfo(np.int64
).max: # comprobamos el max y el min del tipo
            tipo_optimo = np.int64
    return tipo_optimo
```

Figura 36.- Función que realiza el estudio del ajuste de tipos enteros.

```
# Creamos una función para ver los tipos de datos, max y min de cada una de las
# columnas numéricas y booleanas
# Dentro de esta funcion se llama a otra funcion que comprueba el tipo óptimo para los
# enteros
def resumen_tipos_datos(dataf):
    tipos_enteros = ["int8", "int16", "int32", "int64", "uint8", "uin
t16", "uint32", "uint64"]
    for col in dataf.columns:
        if dataf[col].dtype != object:   # No tenemos en cuenta
                                         # los strings
            print("El tipo de la columna {0} es {1}, el valor mínimo e
s {2} y el máximo es {3}".format(col, dataf[col].dtype,dataf[col].min(
), dataf[col].max()))
        if dataf[col].dtype in tipos_enteros:   # Comprobamos que
                                                #es un tipo entero
            print("Es un tipo adecuado es: {0}".format(ajustar_tipos_i
nt(dataf[col].min(), dataf[col].max())))
```

Figura 37.- Función que realiza el estudio para el ajuste de tipos de datos.

Con los resultados obtenidos ajustamos los tipos de datos para no desperdiciar espacio en disco.

Continuamos realizando una representación de los datos estadísticos de todos los campos numéricos. Con esto podemos ver el número de datos por columna, la media, la desviación estándar, el mínimo y máximo y también los cuartiles.

Para poder ver de manera gráfica la distribución de los datos, generamos todos los histogramas de los datos que contiene el Dataframe.

Agrupamos los datos de las clases y realizamos un conteo para ver la distribución de los datos en ellas, podemos ver para todas las medidas que los datos se agrupan principalmente en 2 clases y que una de las clases no suele contener casi datos (ver Figura *38*).

Comprobamos si algunas de las columnas tienen datos faltantes, cuando hemos detectado las columnas que tienen datos faltantes, hemos rellenado los datos con los valores medios para esa columna, de esta forma eliminamos esos datos faltantes.

Para finalizar comprobamos la correlación que existe entre la clase que vamos a intentar predecir y las otras columnas de datos.

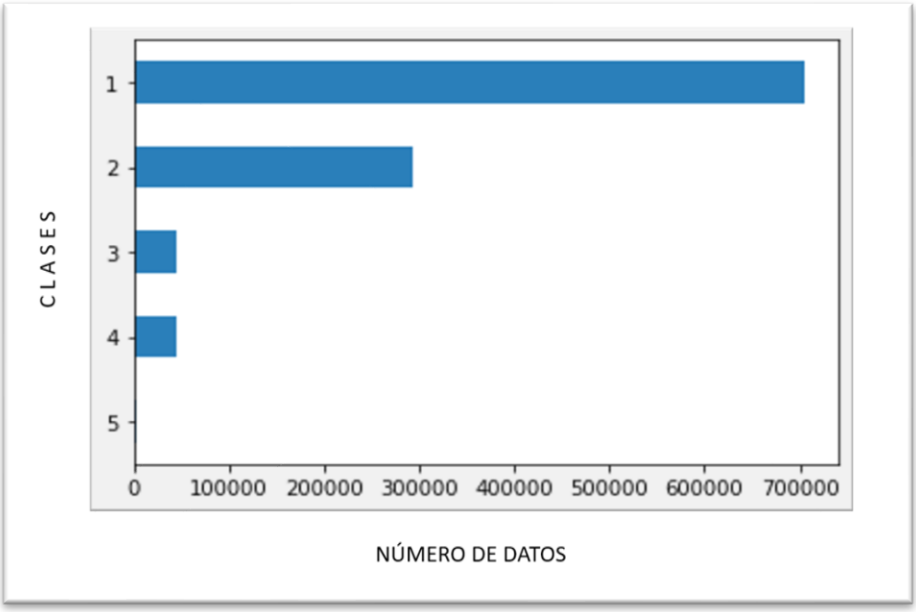

Figura 38.- Representación de los datos agrupados por clases (PM25).

4.3. Algoritmos de ML

En este apartado vamos a explicar los algoritmos de ML que se han utilizado en las tareas de clasificación, se han utilizado los mismos por cada una de las cuatro magnitudes que se han seleccionado. Una vez que ya se ha finalizado el análisis exploratorio y tenemos los datos preparados, solamente nos queda realizar el escalado de los datos para que todos se encuentren en el mismo rango y de esta manera los algoritmos no de más importancia a valores más grandes de un determinado campo simplemente porque sean mayores. En este caso se ha decidido utilizar MinMaxScaler que realiza un escalado de datos ajustándolo en un rango entre 0 y 1 (Scikit-learn, 2022). Con los datos ya escalados podemos comenzar a realizar el entrenamiento y predicción con los distintos algoritmos de clasificación. Para todos los algoritmos se ha dividido el conjunto de datos en datos de entrenamiento y datos de test, el tamaño de los datos de entrenamiento es el 80% de los datos y dejamos el 20% para test, para ello se ha utilizado la función *train_test_split* que se encarga de realizar los dos conjuntos de datos de manera aleatoria. Se han elegido estos porcentajes de datos para cada conjunto, debido a que utilizando entre un 70% y un 80% del conjunto de datos para entrenamiento, se suele evitar el sobreajuste de los resultados. En todos los casos se realiza validación cruzada con cv = 5 cuando se han utilizado los datos de entrenamiento, debido a su mayor volumen para reducir los tiempos de ejecución y cv = 10 cuando se han utilizado los datos de test, que

son un 20% de los datos totales, por lo que son 4 veces menos que los datos de entrenamiento. Además, se utiliza un mayor número de *cv* para los datos de test debido a que son los que posteriormente utilizaremos para evaluar los distintos modelos de aprendizaje.

Se utilizará el Accuracy (en castellano, Exactitud) como medida para evaluar los resultados de los algoritmos conjuntamente con la matriz de confusión. Se ha seleccionado esta forma de evaluar los datos, debido a que nos da el valor del porcentaje de los datos que se han acertado, aunque hay que tener cuidado al utilizarlo porque si las clases no están bien balanceadas puede dar un valor alto y no ser un buen algoritmo de predicción, en nuestros casos es muy normal que los peores escenarios no tienen muchos datos, pero la mayoría de los datos se encuentran entra las dos primeras clases y algunos entre la clase 3 y 4, el caso más extremo tiene pocos datos. Si en nuestro caso solamente hubiésemos tenido 2 clases y no balanceadas, no sería la medida más apropiada. De todas formas, para detectar el problema de predecir la clase mayoritaria que se puede producir cuando se utilizar el Accuracy, se evalúan los resultados también teniendo en cuenta la matriz de confusión, con ella podemos ver si la predicción se realiza siempre sobre los datos de la misma clase y en ese caso, aunque el valor del Accuracy fuese bueno, tendríamos que descartar el algoritmo que siempre indica la misma clase.

4.3.1. Algoritmos Gaussianos

Para el entrenamiento de los datos con los algoritmos Gaussianos, se ha creado una función que se le pasa como parámetro la clase del algoritmo que queremos ejecutar, junto con el nombre y los datos de entrenamiento y validación, con todos estos datos, se realiza el entrenamiento, se calcula el Accuracy y se crea la matriz de confusión. Con los datos calculados, se realiza la visualización en un mapa de calor la matriz de confusión, los distintos resultados de la validación cruzada junto con la media de los datos obtenidos (ver Figura *39*).

```
# función que recibe los parámetros con los datos de entrenamien
to y test, también el modelo de aprendizaje automático
# y el nombre del modelo para mostrar en el texto
# entrena el modelo, realiza las predicciones y calcula la preci
sión, además muestra la matriz de confusión

def ejecutarModelo(X_train, X_test, y_train, y_test, modelo, nom
breModelo):

    modelo.fit(X_train, y_train)

    y_pred =  modelo.predict(X_test)
    score = accuracy_score(y_test, y_pred)
    mat = confusion_matrix(y_pred, y_test)

    ax = sns.heatmap(mat, annot=True, cmap='Blues', fmt='d')

    ax.set_title('Matriz de confusión\n\n');
    ax.set_xlabel('\n Valores Pred')
    ax.set_ylabel('Valores Reales ');

    # Etiquetamos los ejes
    ax.xaxis.set_ticklabels(sorted(y_train.unique()))
    ax.yaxis.set_ticklabels(sorted(y_train.unique()))

    # Se muestra la matriz de confusión
    plt.show()

    # Realizamos validación cruzada con los datos de test cv 10
    scorescv_test = cross_val_score(modelo, X_test, y_test, cv=1
0)

    # Realizamos validación cruzada con los datos de entrenamien
to cv 5
    scorescv_train = cross_val_score(modelo, X_train, y_train, c
v=5)

    # Calculamos los valores medios obtenidos en el accuracy
    scorescv_test_mean = scorescv_test.mean()
    scorescv_train_mean = scorescv_train.mean()

    # Imprimimos los valores obtenidos
    print('\nUtilizando ', nombreModelo, ' obtenemos los siguien
tes resultados de accuracy:')
    print(' - Datos de test', scorescv_test_mean, '\n - Datos de
entrenamiento ', scorescv_train_mean)
    print('\n\nUtilizando ', nombreModelo, ' obtenemos los sigui
entes resultados de accuracy de todas las validaciones cruzadas
realizadas:')
    print(' - Datos de test', scorescv_test, '\n - Datos de entr
enamiento ', scorescv_train)
```

Figura 39.- Función de ejecución de los distintos modelos gaussianos.

4.3.2. Algoritmo KNN

Al igual que se ha realizado para ejecutar los modelos Naive Bayes, se ha creado una función para poder ejecutar distintos algoritmos KNN pasando como parámetro distinto valor para el número de vecinos a tener en cuenta. En esta función se realiza el entrenamiento de lo modelo con los datos que se han pasado como parámetro, se realiza la predicción de los datos, con ellos calculamos la matriz de confusión, se calcula el Accuracy de las predicciones. Al finalizar todo el proceso se representa la matriz de confusión y los resultados obtenidos de las predicciones. Para este modelo se van a realizar ejecuciones teniendo en cuenta 3, 5 y 7 vecinos (ver Figura *40*). Se han utilizado estos valores, comenzando por el valor 3 y se ha ido comprobando si se mejoraban los resultados incrementando el valor de k, pero se ha comprobado que dependiendo de los distintos conjuntos de datos los resultados iban siendo muy parecidos, por lo que se ha optado por utilizar estos 3 valores de k y comparar los datos para cada caso en particular.

```
# función que recibe los parámetros con los datos de entrenamiento y
test, también el número k a aplicar
# entrena el modelo, realiza las predicciones y calcula la precisión
, además muestra la matriz de confusión
def modeloKNN(X_train, X_test, y_train, y_test,n):

    #Modelo de Vecinos más Cercanos
    modelo = KNeighborsClassifier(n_neighbors=n)
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)
    mat = confusion_matrix(y_pred, y_test)

    ax = sns.heatmap(mat, annot=True, cmap='Blues', fmt='d')

    ax.set_title('Matriz de confusión\n\n');
    ax.set_xlabel('\n Valores Pred')
    ax.set_ylabel('Valores Reales ');

    # Etiquetamos los ejes
    ax.xaxis.set_ticklabels(sorted(y_train.unique()))
    ax.yaxis.set_ticklabels(sorted(y_train.unique()))

    # Se muestra la matriz de confusión
    plt.show()

    # Realizamos validación cruzada con los datos de test cv 10
    scorescv_test = cross_val_score(modelo, X_test, y_test, cv =10)

    # Realizamos validación cruzada con los datos de entr. cv 5
    scorescv_train = cross_val_score(modelo, X_train, y_train, cv =5
)

    # Calculamos los valores medios obtenidos en el accuracy
    scorescv_test_mean = scorescv_test.mean()
    scorescv_train_mean = scorescv_train.mean()

    # Imprimimos los valores obtenidos
    print('\nUtilizando KNN con K = ', n, ' obtenemos los siguientes
resultados de accuracy:')
    print(' - Datos de test', scorescv_test_mean, '\n - Datos de ent
renamiento ', scorescv_train_mean)
    print('\n\nUtilizando KNN con K = ', n, ' obtenemos los siguient
es resultados de accuracy de todas las validaciones cruzadas realiza
das:')
    print(' - Datos de test', scorescv_test, '\n - Datos de entrenam
iento ', scorescv_train)
```

Figura 40.- Función para ejecuciones de KNN.

4.3.3. Algoritmo Bosques Aleatorios

Al realizar solamente un tipo de ejecución sobre el algoritmo de Bosques Aleatorios, no se ha creado una función para su ejecución, pero se han realizado de manera similar a los otros dos tipos de algoritmos utilizados. Se realizará el entrenamiento del modelo con los datos de entrenamiento, se calculan las predicciones y una vez realizado todo lo anterior, se imprimen los resultados obtenidos (ver Figura *41*).

```
modelo = RandomForestClassifier(n_estimators= 10)

algoritmoRF = modelo.fit(X_train, y_train)
y_pred = modelo.predict(X_test)

mat = confusion_matrix(y_pred, y_test)

ax = sns.heatmap(mat, annot=True, cmap='Blues', fmt='d')

ax.set_title('Matriz de confusión\n\n');
ax.set_xlabel('\n Valores Pred')
ax.set_ylabel('Valores Reales ');

# Etiquetamos los ejes
ax.xaxis.set_ticklabels(sorted(y_train.unique()))
ax.yaxis.set_ticklabels(sorted(y_train.unique()))

# Se muestra la matriz de confusión
plt.show()

# Realizamos validación cruzada con los datos de test cv 10
scorescv_test = cross_val_score(algoritmoRF, X_test, y_test, cv=
10)
# Realizamos validación cruzada con los datos de entr. cv 5
scorescv_train = cross_val_score(algoritmoRF, X_train, y_train,
cv=5)

# Calculamos los valores medios obtenidos en el accuracy
scorescv_test_mean = scorescv_test.mean()
scorescv_train_mean = scorescv_train.mean()

# Imprimimos los valores obtenidos
print('\nUtilizando Random Forest obtenemos los siguientes resul
tados de accuracy:')
print(' - Datos de test', scorescv_test_mean, '\n - Datos de ent
renamiento ', scorescv_train_mean)
print('\n\nUtilizando Random Forest obtenemos los siguientes res
ultados de accuracy de todas las validaciones cruzadas realizada
s:')
print(' - Datos de test', scorescv_test, '\n - Datos de entrenam
iento ', scorescv_train)
```

Figura 41.- Ejecución del entrenamiento de Bosques Aleatorios.

4.4. Visualización de los datos

En el siguiente apartado vamos a mostrar los distintos gráficos que se han generado para la representación de los conjuntos de datos de manera visual y como todos ellos han sido integrados para formar parte del cuadro de mandos.

4.4.1. Gráficos sobre estaciones

El primer gráfico realizado trata de representar el número de estaciones que realizan la medición de cada uno de los contaminantes que estamos teniendo en cuenta, no todas las estaciones realizan la medición de todos los gases, por este motivo realizando esta representación, podemos ver los puntos de medida que existen en la ciudad de cada parámetro de calidad del aire. Se puede ver el gráfico radial el número de estaciones que realizan la medida de cada valor (ver Figura *42*):

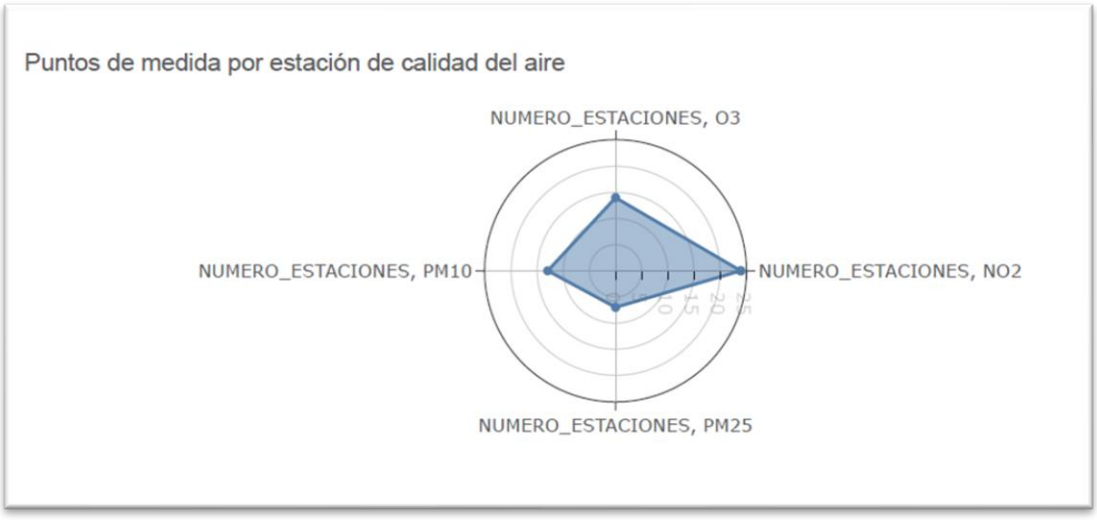

Figura 42.- Representación del número de estaciones que realizan la medida de cada valor.

En el siguiente gráfico se puede visualizar qué estaciones de medida tienen un dato de valor de medida de calidad del aire mayor y menor, con ello simplemente de un vistazo se podría tener la información de donde se localiza la estación que tiene mejor y peor calidad del aire, como vemos, las que aparecen a la izquierda de la imagen y con un tamaño del cuadro mayor son las que peor calidad del aire tienen y las que aparecen a la derecha y que son más pequeñas tienen un menor valor y eso indica mejor calidad del aire (ver Figura *43*).

Figura 43.- Medidas por cada una de las estaciones indicando nombre de la calle.

En el conjunto de datos, unos de los valores que tenemos son los que hacen referencia a la posición geográfica de las estaciones, tanto de medida de calidad del aire, de la que tenemos solamente 25, como los puntos de medida del tráfico que se encuentran distribuidos por numerosas calles a lo largo y ancho de la ciudad.

Se han obtenido gráficos del posicionamiento de las estaciones de medida de calidad del aire en la que podemos observar la distribución y se ha incluido para definir el tamaño de los puntos, el valor de los contaminantes a mayor tamaño mayor nivel de contaminación en esa estación de medida (ver Figura *44*).

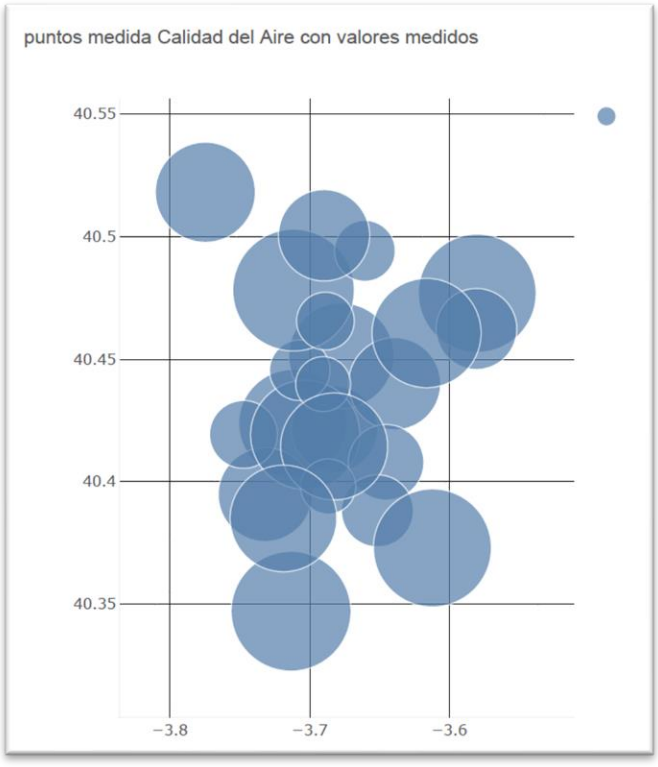

Figura 44.- Distribución geográfica estaciones de calidad del aire y su contaminación.

Para los puntos de medida del tráfico, se ha realizado una división en 2 gráficos independientes teniendo en cuenta si el punto de medida es en la Calle30 o en otra vía. Se ha creado un gráfico simple que solamente tiene en cuenta la posición a la que hace referencia la longitud y la latitud y luego se ha creado otro gráfico que tiene en cuenta el valor de intensidad del tráfico en cada punto y de este valor depende el tamaño del punto (ver Figura *45*).

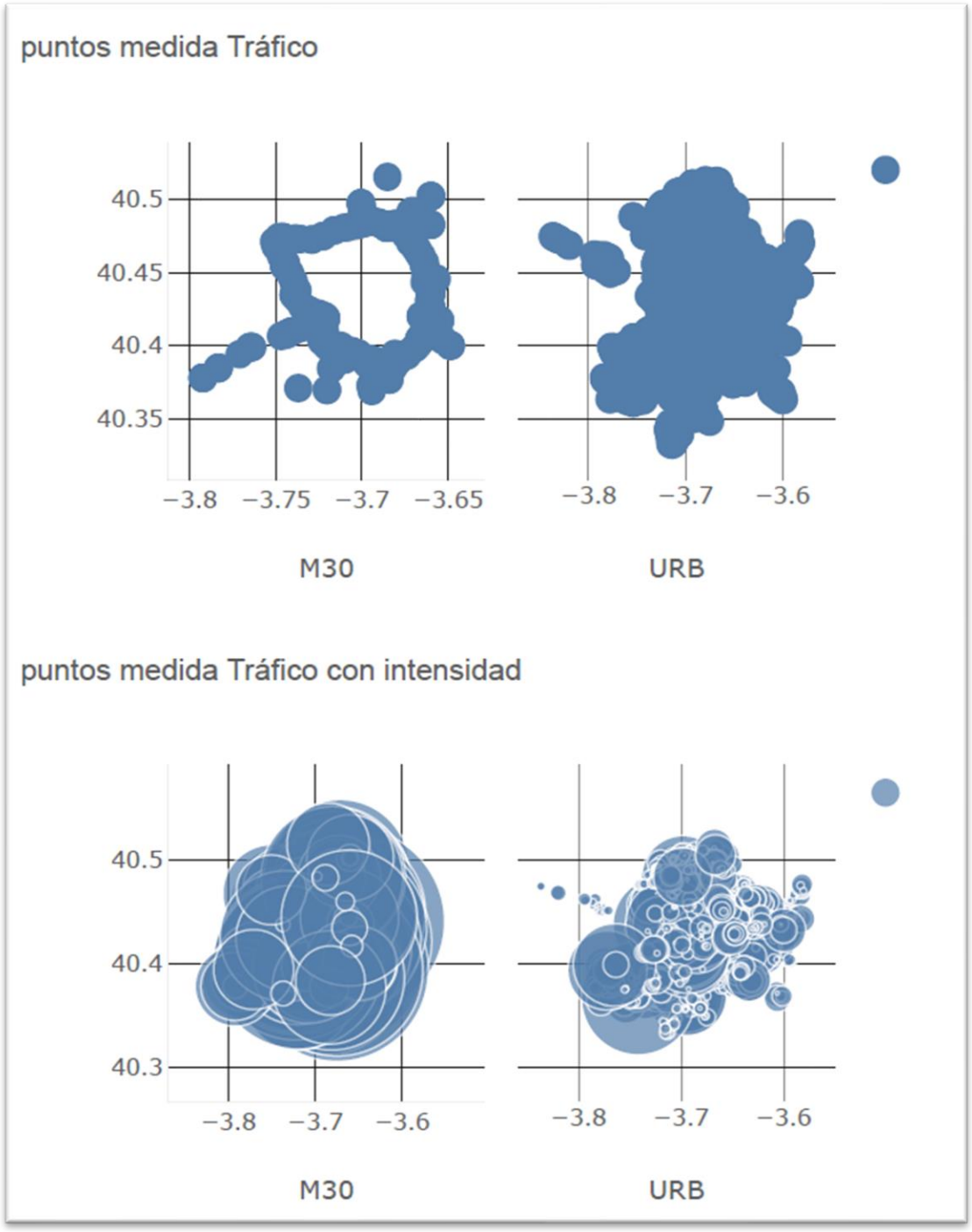

Figura 45.- Puntos de medida del tráfico - posición e intensidad.

4.4.2. Gráficos medidas calidad del aire

Para la elección de los gráficos a realizar se ha tenido en cuenta el factor principal del tiempo, de esta manera podemos ver cómo van cambiando los datos de calidad del aire a lo largo de un periodo.

4.4.2.1. Valores por meses

En esta representación gráfica se puede ver si existe alguna influencia que tenga que ver claramente con la temporada del año y también se podrán observar diferencias teniendo en cuenta el tráfico que en ciertas temporadas disminuye (ver Figura *46*, Figura *47*, Figura *48* y Figura *49*).

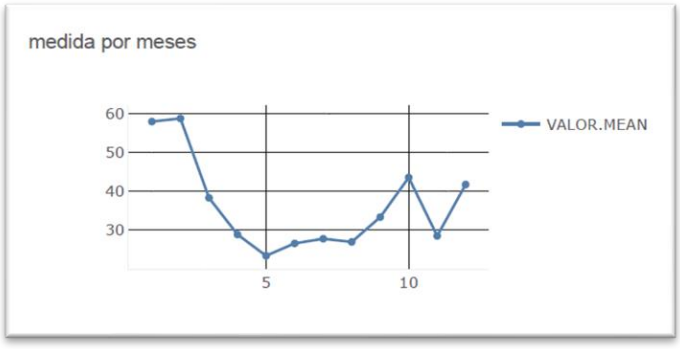

Figura 46.- Medida de dióxido de nitrógeno (NO2) agrupada por meses.

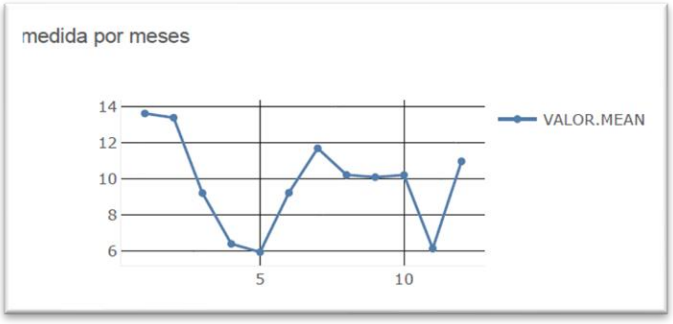

Figura 47.- Medida de PM 2.5 agrupada por meses.

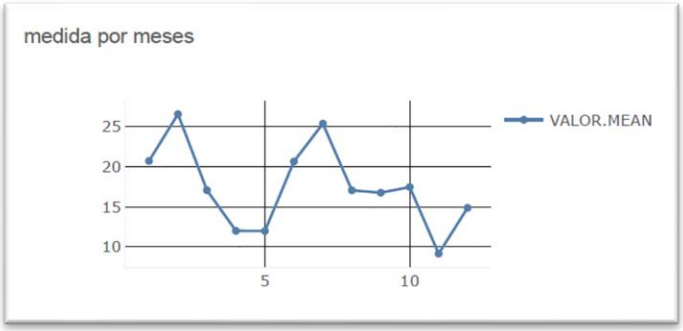

Figura 48.- Medida de PM 10 agrupada por meses.

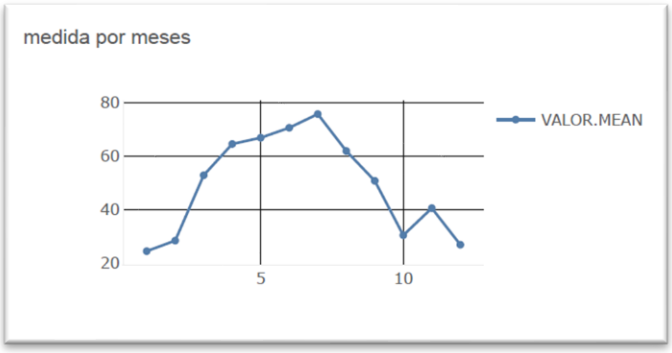

Figura 49.- Medida de ozono (O3) agrupada por meses.

4.4.2.2. Valores por días del mes

Si realizamos la agrupación por los días del mes, podremos ver si tiene cierta variación dependiendo de la parte del mes, a pesar de que no todos los meses tienen 31 días, el valor que se muestra es la media de los meses que si tienen datos para este día (ver Figura *50*, Figura *51*, Figura *52* y Figura *53*).

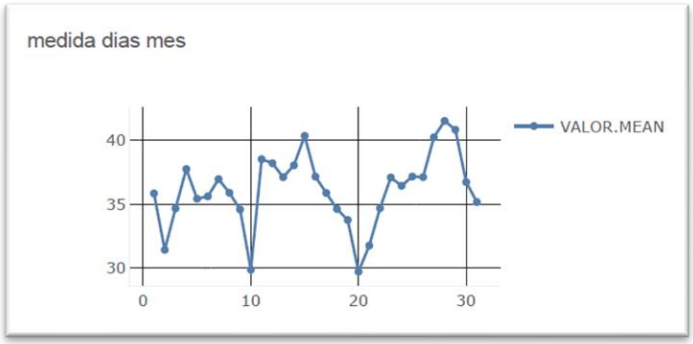

Figura 50.- Medida de dióxido de nitrógeno (NO2) agrupada por días del mes.

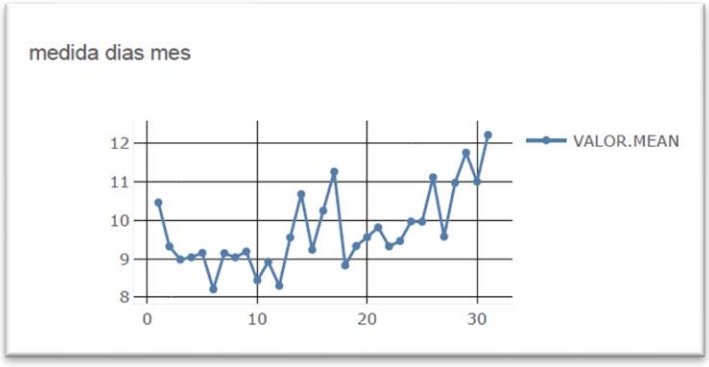

Figura 51.- Medida de PM 2.5 agrupada por días del mes.

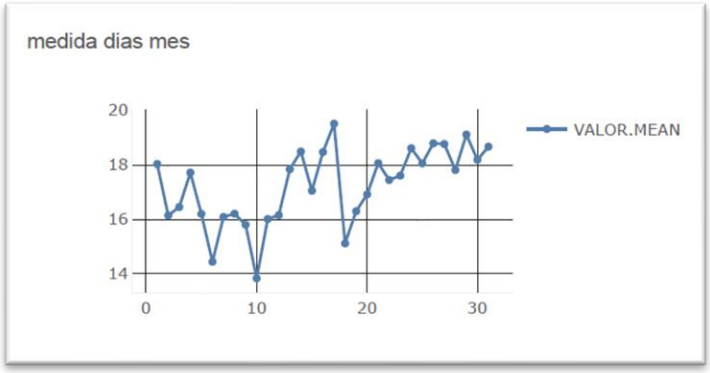

Figura 52.- Medida de PM 10 agrupada por días del mes.

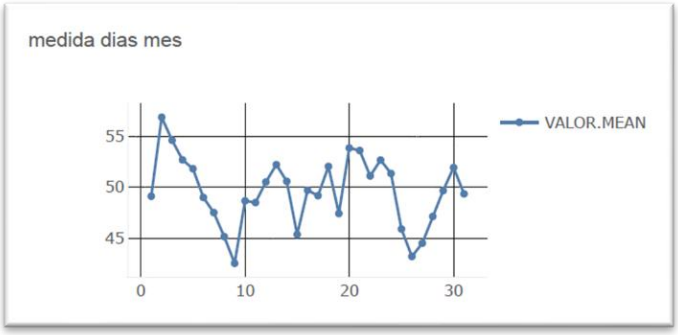

Figura 53.- Medida de ozono (O3) agrupada por días del mes.

4.4.2.3. Valores por días de la semana

Parece interesante poder tener los datos agrupados por cada día de la semana para ver si tiene influencia, sobre todo por la gran diferencia que existe entre el tráfico durante los días de fin de semana con respecto a los laborables (ver Figura *54*, Figura *55*, Figura *56* y Figura *57*).

Figura 54.- Medida de dióxido de nitrógeno (NO2) agrupada por días de la semana.

Figura 55.- Medida de PM 2.5 agrupada por días de la semana.

Figura 56.- Medida de PM 10 agrupada por días de la semana.

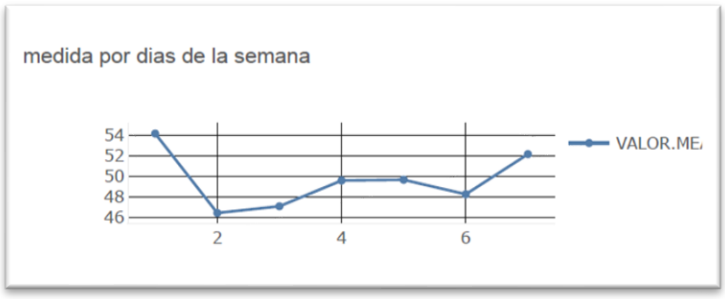

Figura 57.- Medida de ozono (O3) agrupada por días de la semana.

4.4.2.4. Valores por horas

Quizá los gráficos donde se puede ver bastante la incidencia del tráfico en los valores, puede ser el que muestra los datos agrupados por horas, que es donde se ver

claramente la diferencia de tráfico en las ciudades debido a las horas de entrada y salida del trabajo (ver Figura *58*, Figura *59*, Figura *60* y Figura *61*).

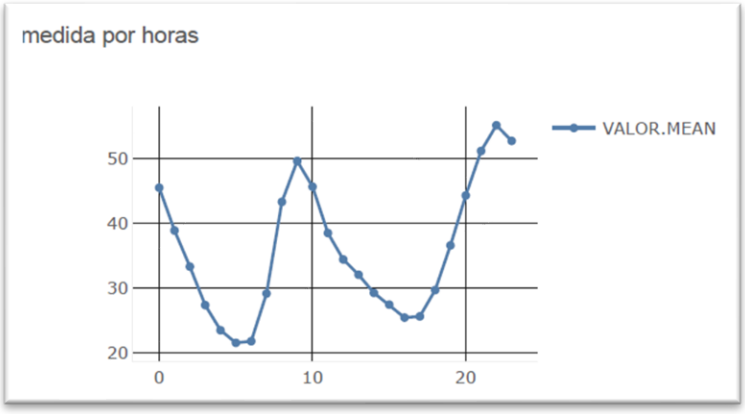

Figura 58.- Medida de dióxido de nitrógeno (NO2) agrupada por horas.

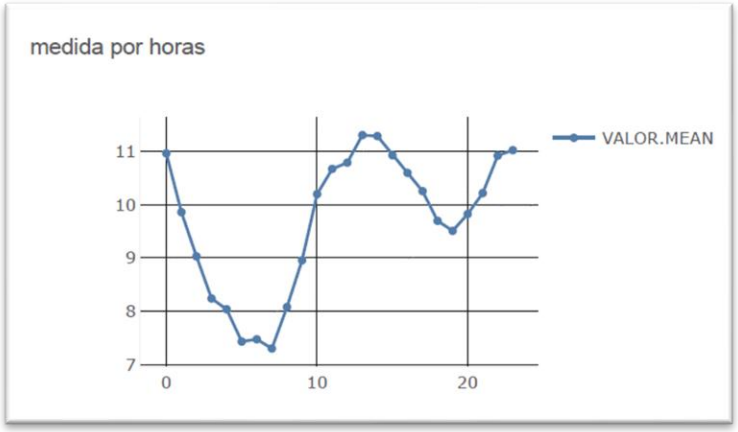

Figura 59.- Medida de PM 2.5 agrupada por horas.

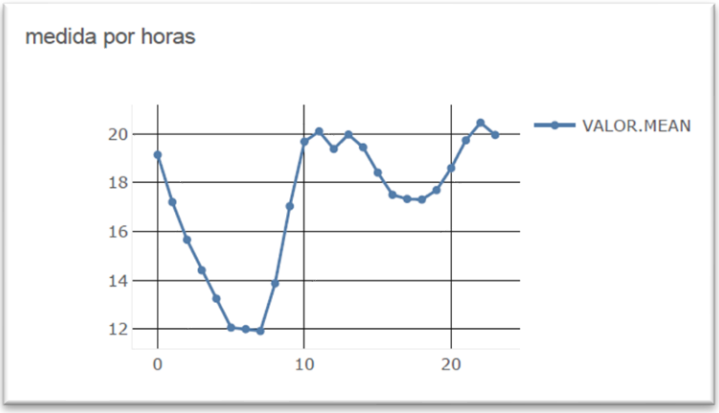

Figura 60.- Medida de PM 10 agrupada por horas.

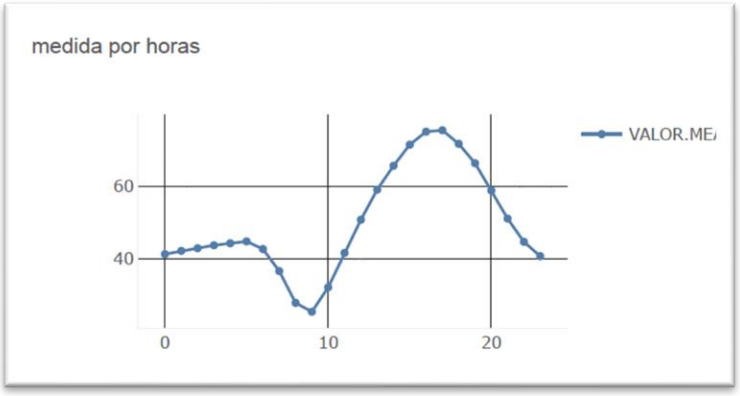

Figura 61.- Medida de ozono (O3) agrupada por horas.

4.4.3. Gráficos medidas tráfico.

Para poder determinar si el tráfico influye en la calidad del aire para determinados parámetros, tenemos que realizar unas agrupaciones similares a las realizadas con las medidas de los gases, de esta manera podemos detectar si en los picos y en los valles del tráfico, si se replica el mismo patrón o parecido en las gráficas de medida de los distintos gases.

4.4.3.1. Valores por meses

En este gráfico se podrá determinar qué meses son los que más tráfico tienen y los que se reduce (ver Figura *62*).

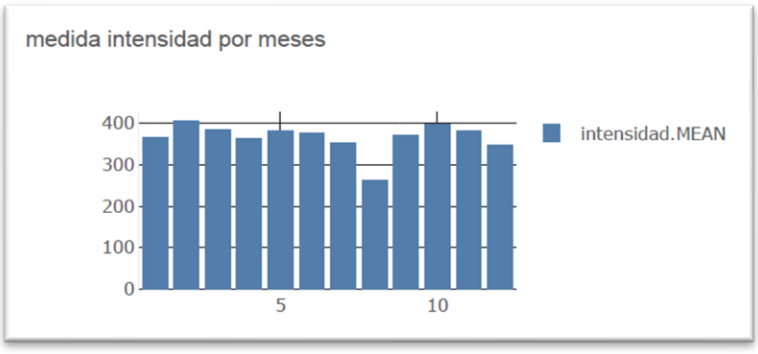

Figura 62.- Medida de intensidad del tráfico agrupado por meses.

4.4.3.2. Valores por días del mes

En el siguiente gráfico quedará de manifiesto si a principio, a mediados o a finales de mes se detecta algún cambio en el flujo del tráfico en la ciudad (ver Figura *63*).

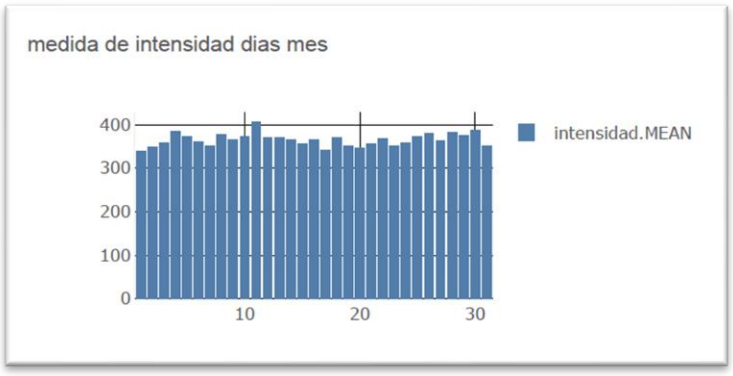

Figura 63.- Medida de intensidad del tráfico agrupado por días del mes.

4.4.3.3. Valores por días de la semana

En el siguiente gráfico se podrá comprobar la diferencia entre los distintos días y uno de los datos más relevantes será el que indica la diferencia entre los días del fin de semana y los días laborables (ver Figura *64*).

Figura 64.- Medida de intensidad del tráfico agrupado por días de la semana.

4.4.3.4. Valores por horas

Uno de los gráficos donde se prevé detectar mayor diferencia es el que muestra los datos del tráfico dependiendo de la hora del día, hay que tener en cuenta que en principio las horas de la noche suelen tener menos movimiento en la ciudad (ver Figura *65*).

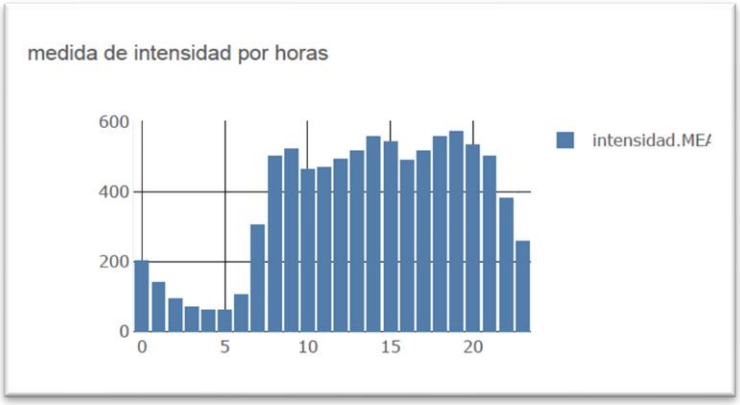

Figura 65.- Medida de intensidad del tráfico agrupado por horas.

5. Evaluación y resultados.

En este capítulo se van a estudiar los resultados obtenidos, tanto en la parte de aprendizaje automático como en la de visualización de los datos.

5.1. Métricas utilizadas para evaluación de resultados

5.1.1. Exactitud (Accuracy)

La exactitud realiza la medición del porcentaje de las predicciones sobre las que el modelo acierta. Es una de las principales métricas para poder evaluar si un modelo realiza mejores predicciones que otros. Para calcular la exactitud se ha utilizado la función *accuracy_score* de Sklearn que realiza el cálculo con la siguiente fórmula (ver Figura *66*):

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Figura 66.- Fórmula para calcular el Accuracy.

5.1.2. Matriz de confusión

Dentro del aprendizaje automático, una forma de representar lo capaz que es un sistema de realizar predicciones, la matriz de confusión es una herramienta que nos permite visualizar el resultado que se obtiene una vez que se ha ejecutado un algoritmo de aprendizaje supervisado. Cada una de las columnas que componen la matriz representa una clase y cada una de las filas representa las instancias en la clase real. Con estos datos podemos ver que tipos de aciertos y errores estamos teniendo en el modelo que estamos evaluando una vez realizado el proceso de aprendizaje (ver Figura *67*).

Figura 67.- Ejemplo matriz de confusión con 2 clases.

Los posibles casos que se representan serán:

- **Verdadero positivo (VP):** El valor real es positivo y el resultado obtenido por el algoritmo también que era positivo.
- **Verdadero negativo (VN):** El valor real es negativo y el resultado obtenido por el algoritmo también que el resultado era negativo.
- **Falso negativo (FN):** El valor real es positivo, y el resultado obtenido por el algoritmo es negativo. Esto es lo que en estadística se conoce como error tipo II.
- **Falso positivo (FP):** El valor real es negativo, y el resultado obtenido por el algoritmo es positivo. Esto es lo que en estadística se conoce como error tipo I.

5.2. Resultados

En este apartado se van a evaluar los resultados obtenidos por los modelos de aprendizaje automático para cada una de las cuatro medidas sobre las que se han ejecutado los algoritmos. Con estos resultados se podrá concretar cuál de los modelos realiza mejores predicciones sobre los datos proporcionados.

Todos los datos que se muestran son los resultados que se han obtenido utilizando los datos de test, que como se ha indicado anteriormente son el 20 % de los datos totales.

5.2.1. Dióxido de Nitrógeno (NO_2)

A continuación, se van a mostrar los resultados obtenidos en las distintas ejecuciones para el estudio de Dióxido de Nitrógeno, utilizando todos los algoritmos estudiados en el trabajo y las variaciones que se realizan de cada uno de ellos si procede (ver Tabla *11*.- Resultados para NO2).

Algoritmo	Accuracy
Gaussian NB	0,6477
Multinomial NB	0,6593
Bernoulli NB	0,6748
Complement NB	0,5252
KNN (k = 3)	0,7277
KNN (k = 5)	0,7271
KNN (k = 7)	0,7266
Bosques Aleatorios (10 estimadores)	0,8254

Tabla 11.- Resultados para NO2.

Las matrices de confusión obtenidas son (ver Figura 68 y Figura 69):

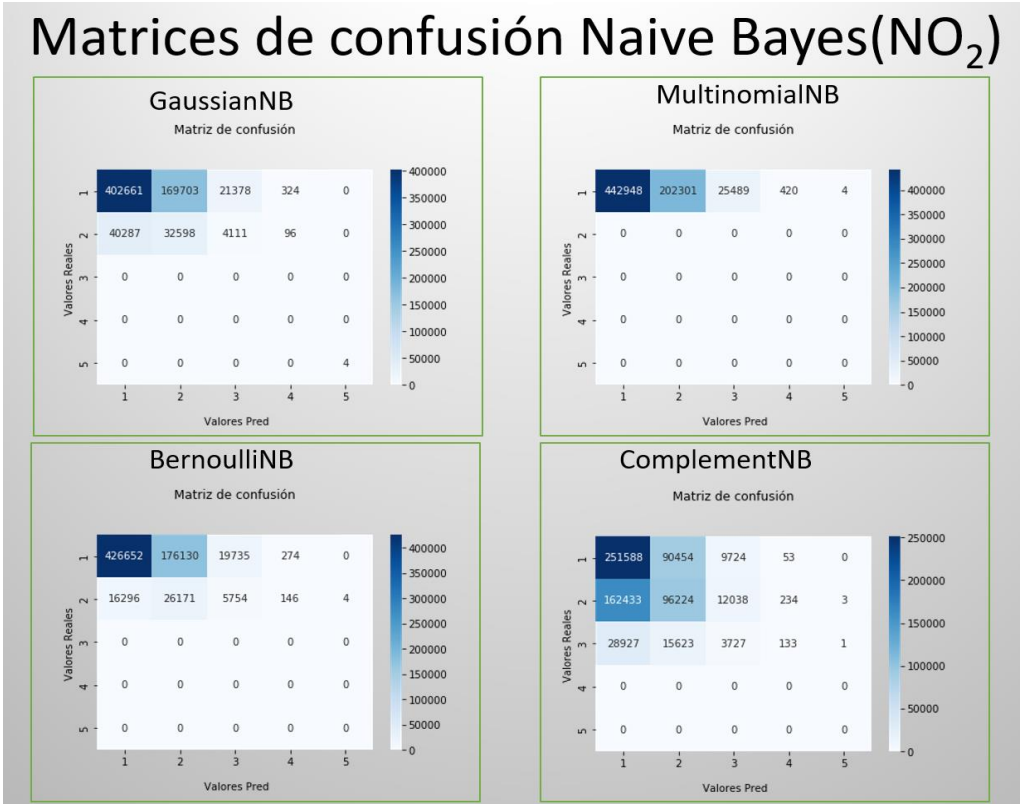

Figura 68.- Matrices de confusión NO2 Naive Bayes.

Los resultados de todos los algoritmos de la familia de NB no obtienen unos resultados de Accuracy superiores de 0.67, incluso vemos que *Complement NB* obtiene el peor resultado con 0.52.

Los algoritmos KNN obtienen unos resultados muy parecidos entre ellos, entre 0.7266 y 0.7277 y bastante mejores que los obtenidos con los algoritmos NB, según se va aumentando el valor de k, el resultado va siendo ligeramente peor, por lo que vemos que utilizando k=3 es el mejor parámetro a utilizar.

Utilizando el algoritmo de Bosques Aleatorios se obtienen los mejores resultados alcanzando un valor de Accuracy de 0.83.

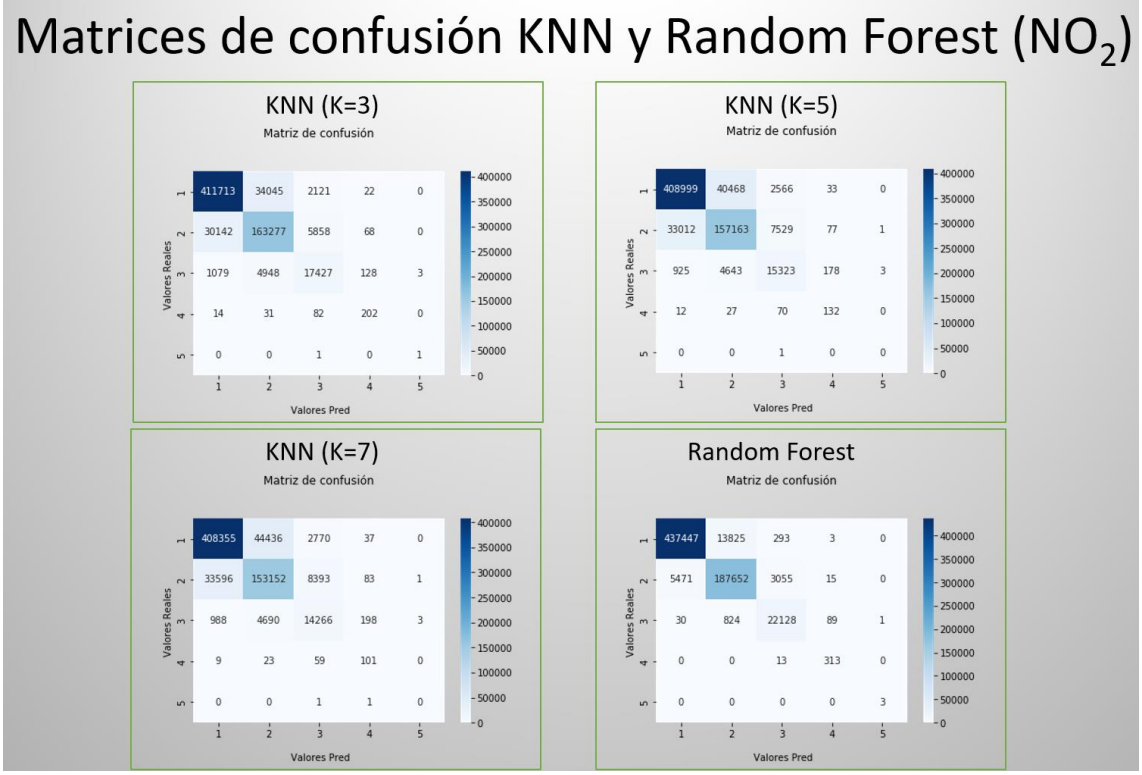

Figura 69.- Matrices de confusión NO2 KNN y Bosques Aleatorios.

5.2.2. Materia particulada < 2.5 micras (PM 2.5)

En este apartado se van a mostrar los resultados obtenidos en las distintas ejecuciones para el estudio de Materia particulada < 2.5 micras, utilizando todos los algoritmos estudiados en el trabajo y las variaciones que se realizan de cada uno de ellos si procede (ver Tabla *12*.- Resultados para PM 2.5).

Algoritmo	Accuracy
Gaussian NB	0,6361
Multinomial NB	0,6439
Bernoulli NB	0,6438
Complement NB	0,4798
KNN (k = 3)	0,6328
KNN (k = 5)	0,6364
KNN (k = 7)	0,6429
Bosques Aleatorios (10 estimadores)	0,7253

Tabla 12.- Resultados para PM 2.5.

Los resultados de todos los algoritmos de la familia de Naive Bayes (NB) no obtienen unos resultados de Accuracy superiores de 0.64, incluso vemos que *Complement* NB obtiene el peor resultado con 0.48.

Los algoritmos KNN obtienen unos resultados muy parecidos entre ellos, entre 0.63 y 0.64 y bastante mejores que los obtenidos con los algoritmos NB, según se va aumentando el valor de k, el resultado va siendo ligeramente mejor, por lo que vemos que utilizando k=7 es el mejor parámetro a utilizar. Los resultados son muy parecidos a los que se han obtenido con los algoritmos NB.

Utilizando el algoritmo de Bosques Aleatorios se obtienen los mejores resultados alcanzando un valor de Accuracy de 0.73.

Las matrices de confusión obtenidas son (ver Figura 70 y Figura 71):

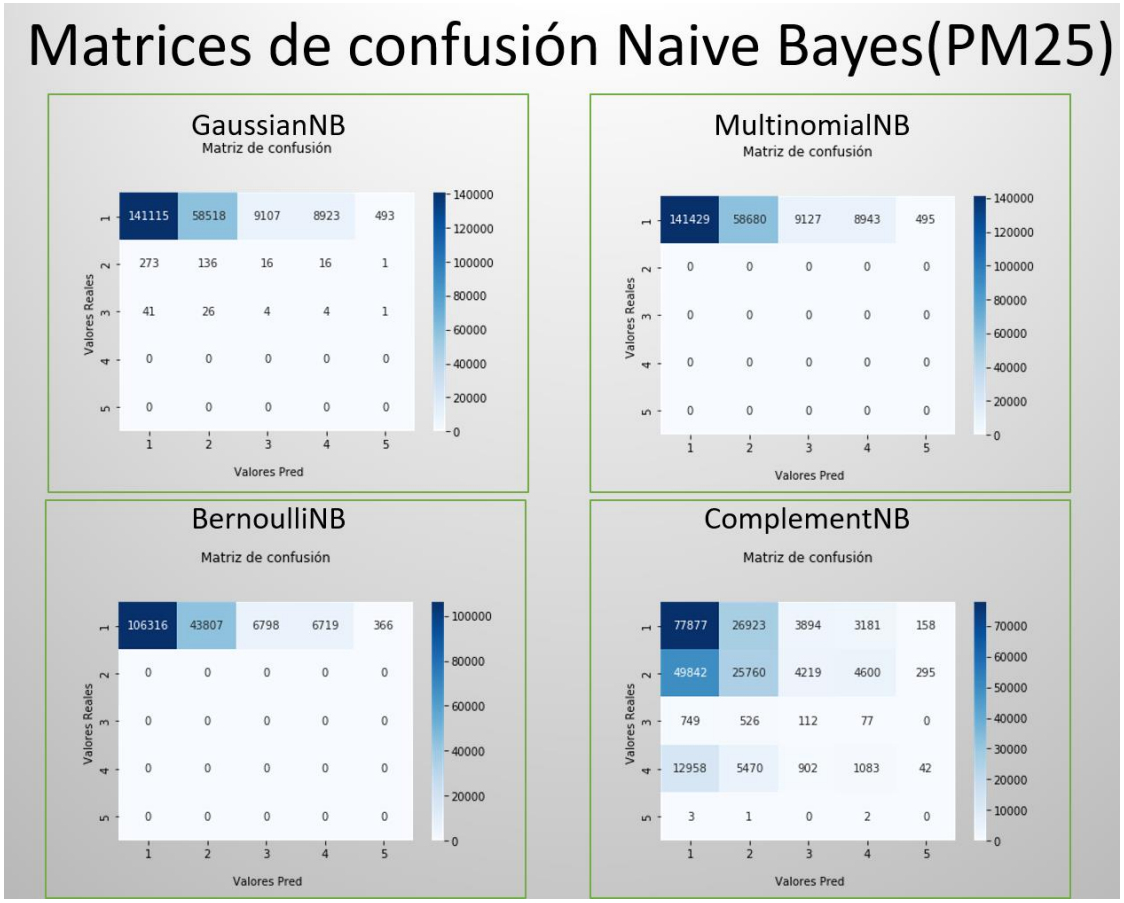

Figura 70.- Matrices de confusión PM25 Naive Bayes.

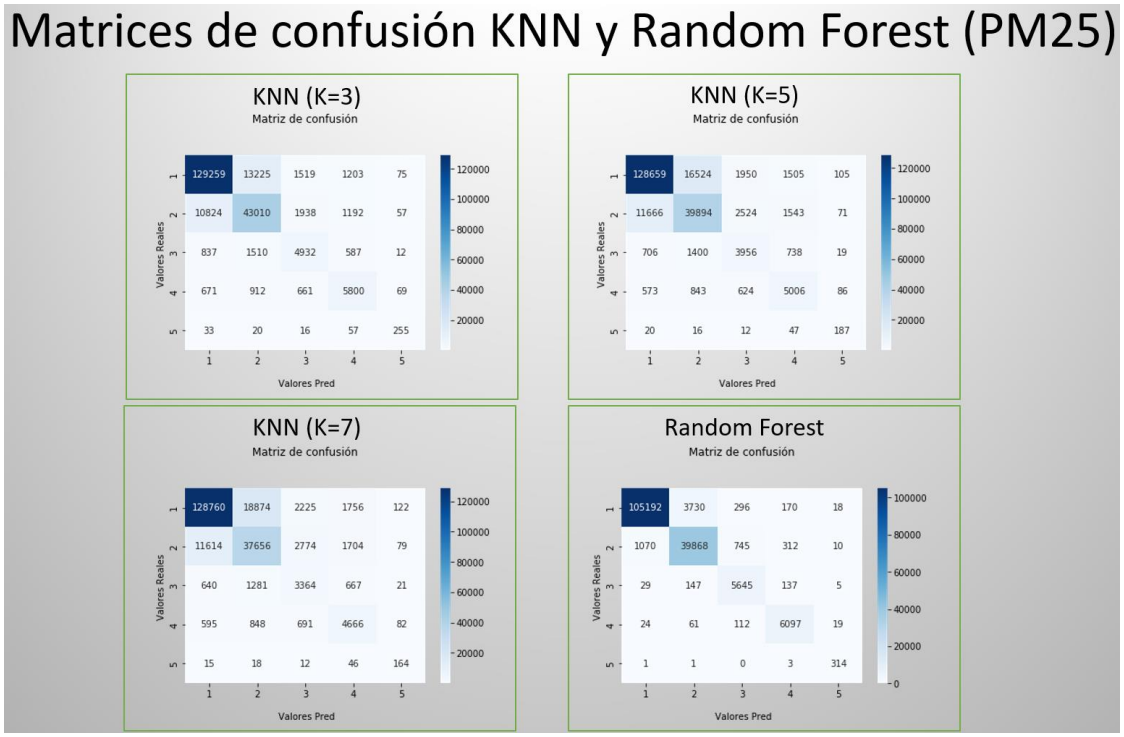

Figura 71.- Matrices de confusión PM25 KNN y Bosques Aleatorios.

5.2.3. Materia particulada < 10 micras (PM 10)

Vamos a mostrar los resultados obtenidos en las distintas ejecuciones para el estudio de Materia particulada < 10 micras, utilizando todos los algoritmos estudiados en el trabajo y las variaciones que se realizan de cada uno de ellos si procede (ver Tabla *13*).

Algoritmo	Accuracy
Gaussian NB	0,6825
Multinomial NB	0,6997
Bernoulli NB	0,6998
Complement NB	0,4886
KNN (k = 3)	0,6787
KNN (k = 5)	0,6890
KNN (k = 7)	0,6940
Bosques Aleatorios (10 estimadores)	0,7617

Tabla 13.- Resultados para PM 10.

Los resultados de todos los algoritmos de la familia de Naive Bayes (NB) no obtienen unos resultados de Accuracy superiores de 0.7, incluso vemos que *Complement* NB obtiene el peor resultado con 0.49.

Los algoritmos KNN obtienen unos resultados muy parecidos entre ellos, entre 0.68 y 0.69 y en este caso no son mejores que los obtenidos con los algoritmos NB, según se va aumentando el valor de k, el resultado va siendo ligeramente mejor, por lo que vemos que utilizando k=7 es el mejor parámetro a utilizar.

Utilizando el algoritmo de Bosques Aleatorios se obtienen los mejores resultados alcanzando un valor de Accuracy de 0.76.

Las matrices de confusión obtenidas son (ver Figura 72 y Figura 73):

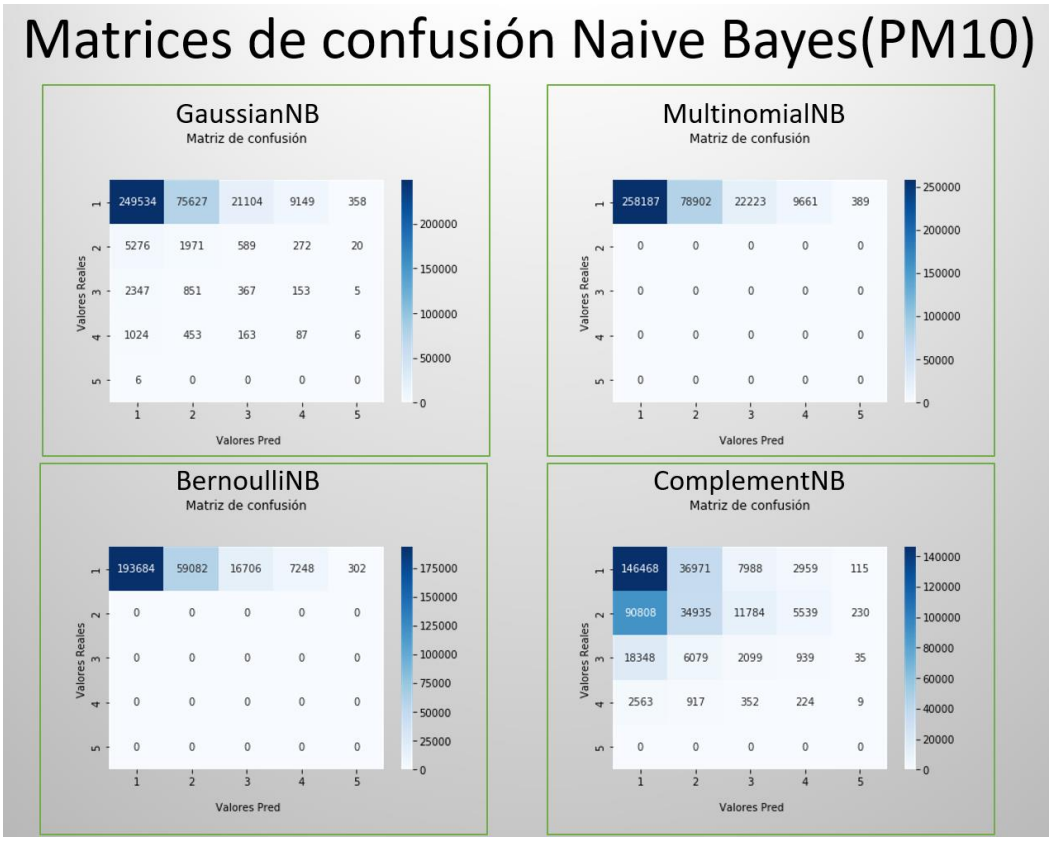

Figura 72.- Matrices de confusión PM10 Naive Bayes.

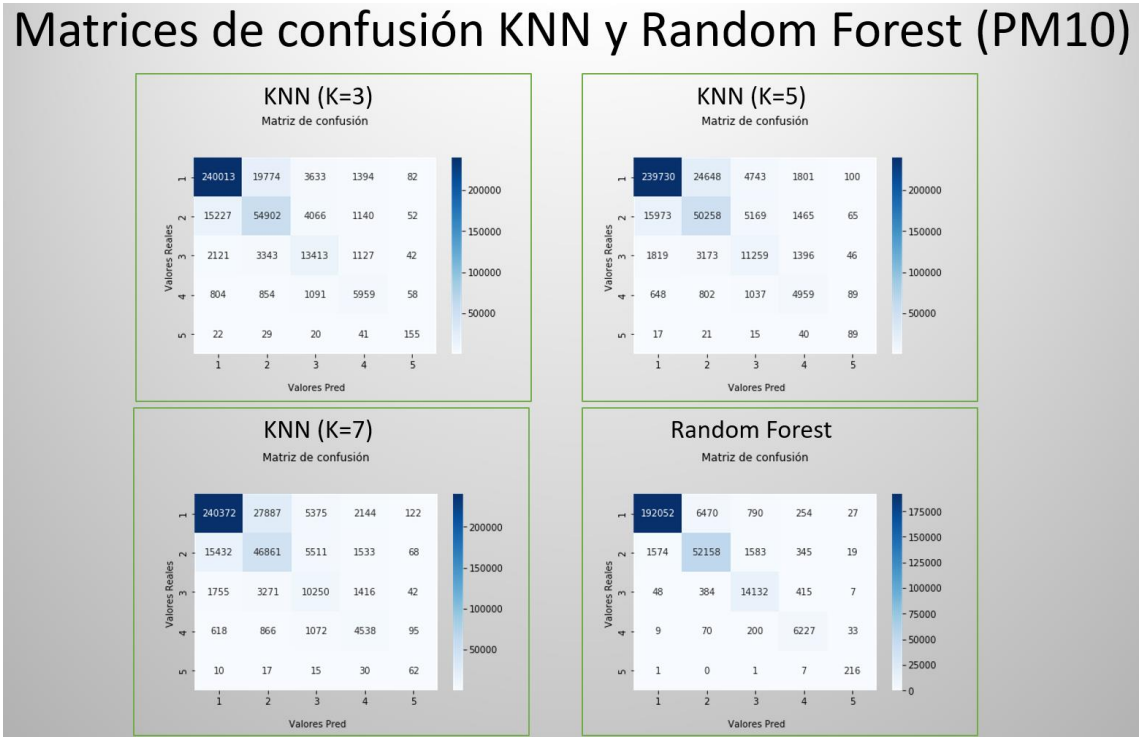

Figura 73.- Matrices de confusión PM10 KNN y Bosques Aleatorios.

5.2.4. Ozono (O3)

Finalizamos mostrando los resultados obtenidos en las distintas ejecuciones para el estudio de Ozono, utilizando todos los algoritmos estudiados en el trabajo y las variaciones que se realizan de cada uno de ellos si procede.

Algoritmo	Accuracy
Gaussian NB	0,5047
Multinomial NB	0,4498
Bernoulli NB	0,4545
Complement NB	0,3877
KNN (k = 3)	0,7112
KNN (k = 5)	0,7070
KNN (k = 7)	0,7035
Bosques Aleatorios (10 estimadores)	0,8827

Tabla 14.- Resultados para O3.

Los resultados de todos los algoritmos de la familia de Naive Bayes (NB) no obtienen unos resultados de Accuracy superiores de 0.5, incluso vemos que *Complement* NB obtiene el peor resultado con 0.39.

Los algoritmos KNN obtienen unos resultados muy parecidos entre ellos, entre 0.70 y 0.71 y bastante mejores que los obtenidos con los algoritmos NB, según se va aumentando el valor de k, el resultado va siendo ligeramente peor, por lo que vemos que utilizando k=3 es el mejor parámetro a utilizar.

Utilizando el algoritmo de Bosques Aleatorios se obtienen los mejores resultados alcanzando un valor de Accuracy de 0.88.

Las matrices de confusión obtenidas son (ver Figura 74 y Figura 75):

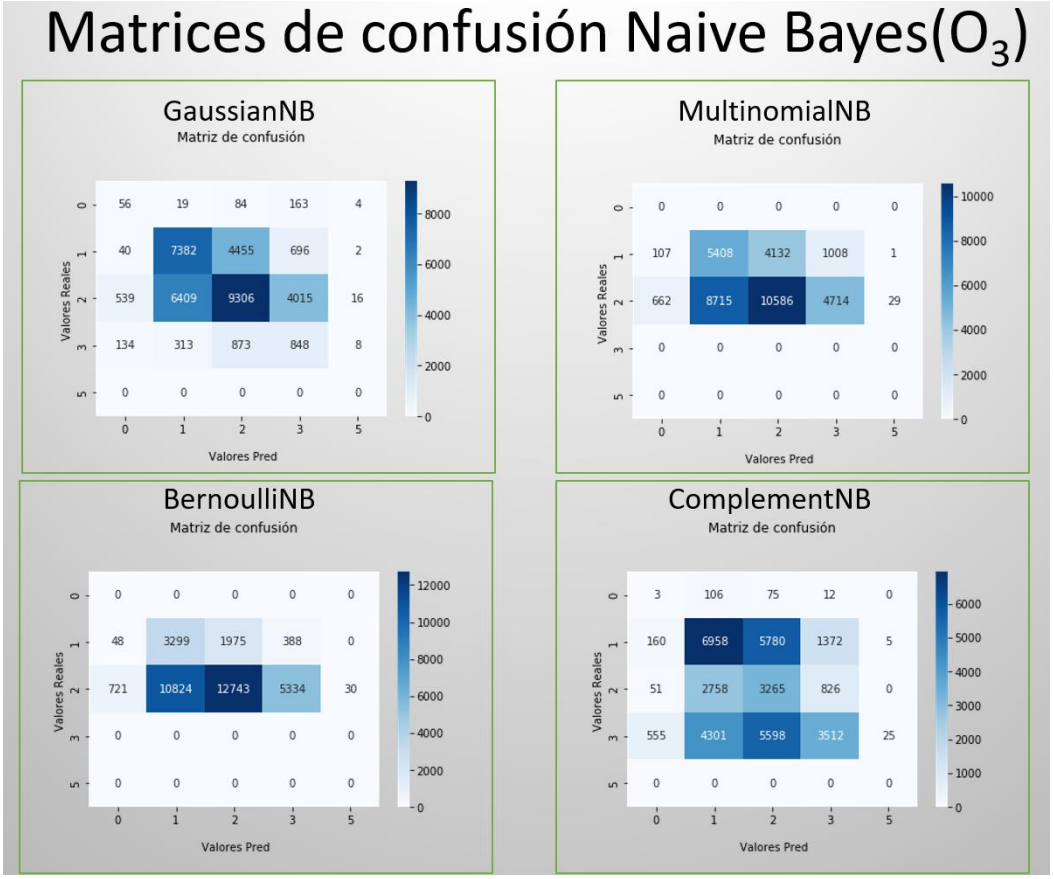

Figura 74.- Matrices de confusión O3 Naive Bayes.

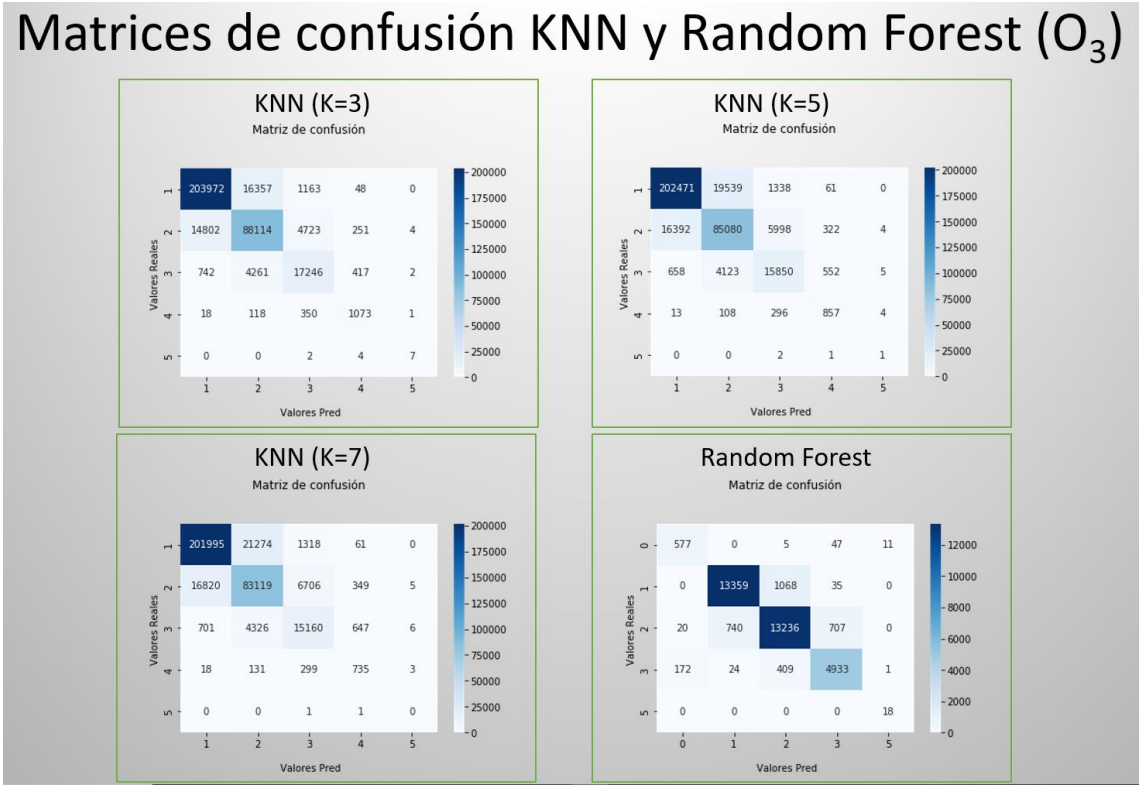

Figura 75.- Matrices de confusión O3 KNN y Bosques Aleatorios.

5.3. Evaluación de cuadro de mando

En este apartado se van a mostrar las principales pestañas de las que se compone el cuadro de mando realizado con ATOTI y utilizando los gráficos creados con los conjuntos de datos una vez preparados para la representación gráfica.

Lo primero que nos encontramos en el cuadro de mando son una serie de combos a rellenar para poder afinar las búsquedas si lo vemos necesario, se puede seleccionar la magnitud, el mes, el día de la semana y la hora, si seleccionamos algún valor en cualquiera de ellos, se actualizarán los gráficos. En esta primera pantalla podemos ver un gráfico tipo radar que indica el número de estaciones que miden los cuatro tipos de gases de los que se han cargado los valores. En la parte inferior se muestran en la parte izquierda y con mayor tamaño las estaciones de medida que tienen valores más altos de los contaminantes seleccionados, en cada cuadrado de las estaciones se muestra la calle donde se encuentran instaladas (ver Figura *76*).

Figura 76.- Pantalla inicial cuadro de mandos

En la siguiente pestaña, se muestran gráficos referentes al posicionamiento geográfico de las distintas estaciones de medida, en la parte izquierda podemos ver la representación de la posición de cada una de las estaciones de medida de calidad del aire y dependiendo del tamaño de cada punto representado indica que a mayor tamaño mayor valor de contaminante. En la parte derecha se muestran los datos de los puntos de medida del tráfico separados en dos grupos, los que se encuentran en la Calle30 (M30) o los que se encuentran en una calle distinta a la Calle30 y que se consideran puntos de medida urbanos (URB), en el gráfico superior solamente se representa la posición y en el inferior la posición y además la intensidad del tráfico, a mayor tamaño de punto, mayor intensidad de tráfico (ver Figura 77Figura *77*).

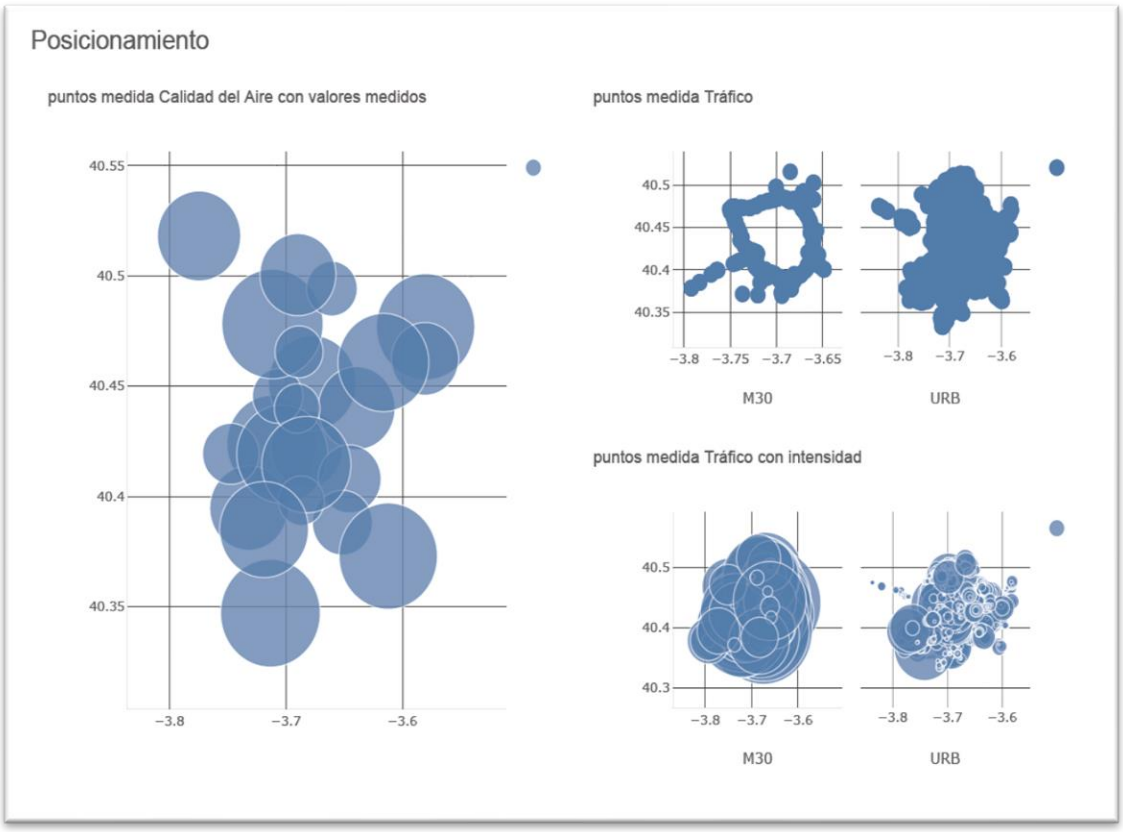

Figura 77.- Cuadro de mandos - posicionamiento puntos de medida.

Continuando con la siguiente pestaña, nos encontramos con gráficos que representan tanto medida de calidad del aire como del tráfico. En la parte superior se puede seleccionar si se quiere uno o varias magnitudes, en el caso del ejemplo se muestra seleccionado NO_2, y también se podría seleccionar una calle determinada de las que tienen estaciones de medida. En la parte izquierda se representa en la gráfica superior el valor de la magnitud agrupado por meses del año y en la inferior el valor agrupado por día del mes, si nos fijamos en la columna de la derecha podemos ver, en la gráfica superior, el valor de intensidad del tráfico agrupado por meses y en la gráfica inferior ese mismo dato, pero agrupado por día del mes (ver Figura 78).

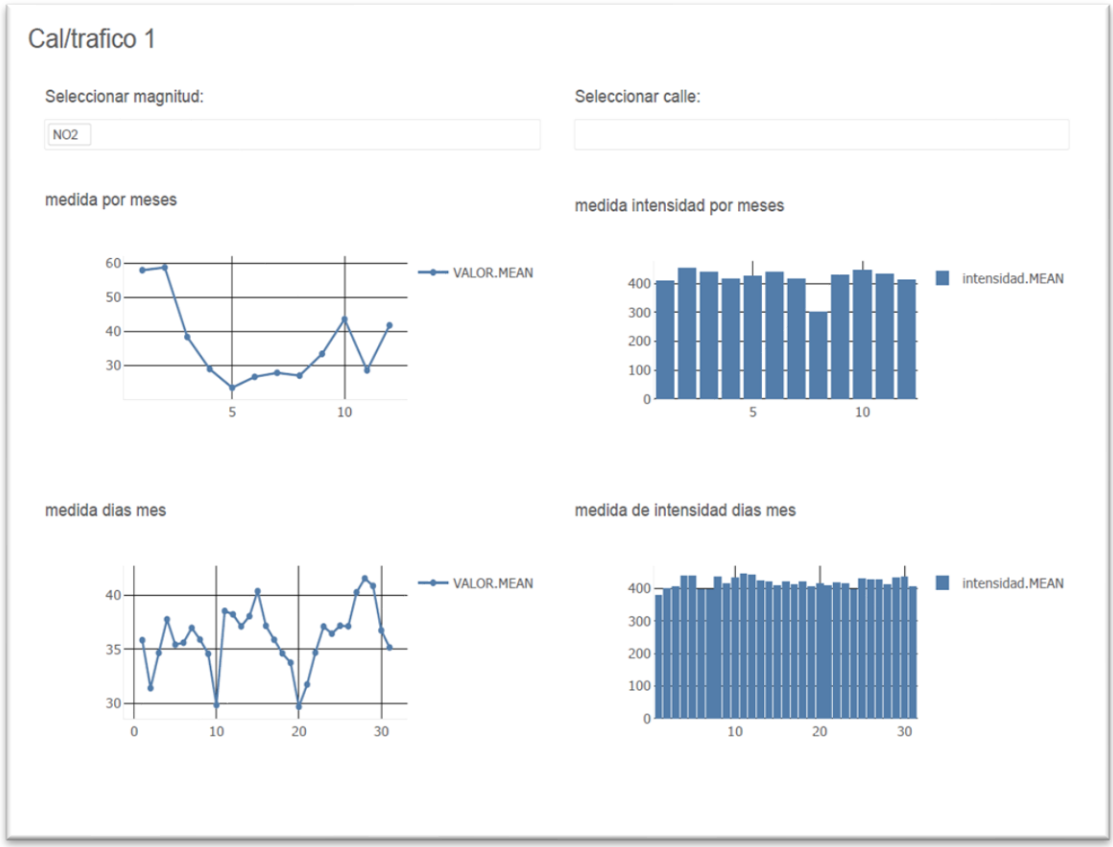

Figura 78.- Cuadro de mandos - calidad del aire/tráfico

En la siguiente pestaña, nos encontramos también valores de la magnitud y de intensidad. En la parte superior nos encontramos con los mismos combos que en la pestaña anterior, por lo que podemos seleccionar la magnitud y la estación de la que queremos obtener la medida. Los gráficos que aparecen en la parte izquierda son referentes al valor de medida de la magnitud, en la parte superior, agrupado por el día de

la semana, comenzando por el domingo, y en la parte inferior los datos agrupados por hora del día. Los gráficos que aparecen en la parte derecha y que tienen como objetivo mostrar los valores de intensidad del tráfico, el superior muestra los datos agrupados por día de la semana, también comenzando por el domingo, y en el inferior los datos agrupados por hora del día (ver Figura 79Figura *79*).

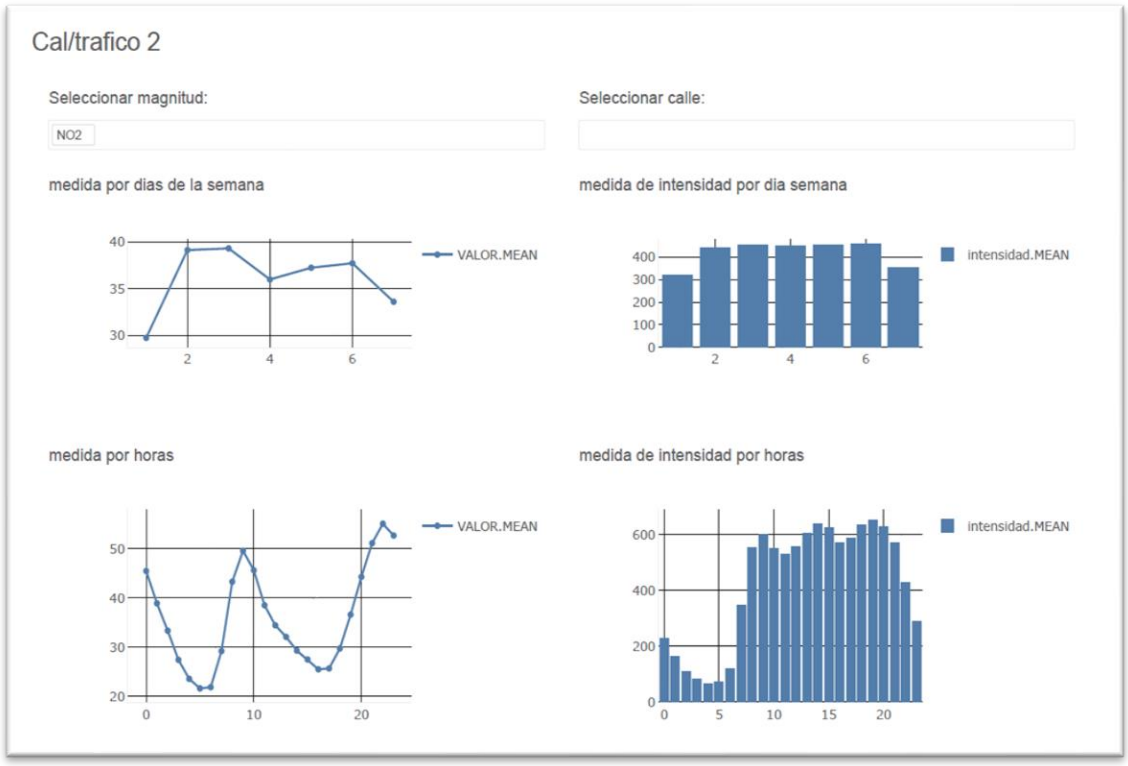

Figura 79.- Cuadro de mandos - calidad del aire/tráfico.

Para finalizar se muestran unas pestañas con los datos de las distintas magnitudes teniendo en cuenta el valor medio por cada día de la semana, separamos los datos de representación de días laborables y de días de fin de semana, la marca que aparece en rojo es el valor que indica que el dato es demasiado alto y podría ser perjudicial para la salud (ver Figura 80Figura *80*).

Utilizando todas las gráficas anteriores, los analistas de datos podrán ver cómo influye el tráfico en la contaminación del aire, en base a esto se podrán implantar medidas para reducir el tráfico a determinadas horas o días que son más críticos y comprobar si las medidas utilizadas realmente producen el efecto esperado en la calidad del aire. También se pueden utilizar los gráficos para intentar adelantarse a determinadas

circunstancias y buscar medidas que afecten lo menos posible a la vida cotidiana de los ciudadanos aunque se implanten durante un periodo de tiempo mayor.

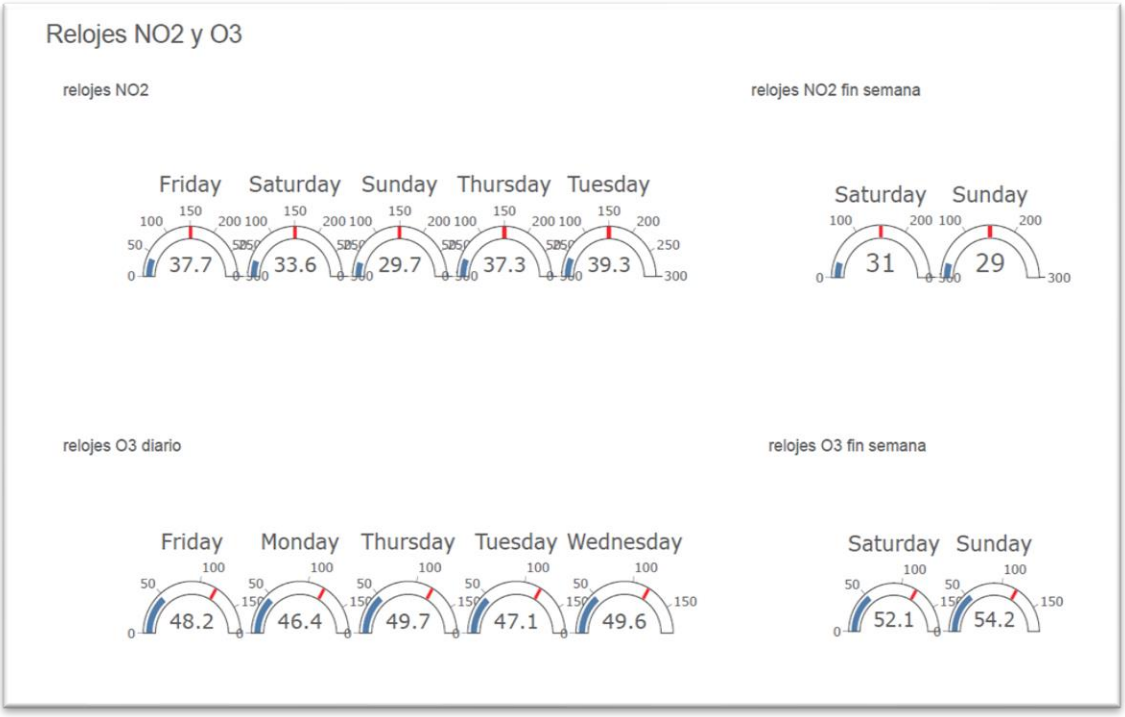

Figura 80.- Cuadro de mandos - magnitudes por día de la semana.

Si analizamos las gráficas obtenidas en el cuadro de mandos podemos ver lo siguiente:

- NO_2:

Teniendo en cuenta los datos agrupados por meses, se puede observar que los datos en los meses de invierno tienen unos valores más altos comparados con los meses de primavera y verano (ver Figura *46*). Si lo comparamos con la gráfica del tráfico agrupado por meses, vemos que no hay una gran diferencia entre los meses de invierno y los demás, solamente se comprueba que en los meses de julio y sobre todo en agosto, se reduce bastante la intensidad del tráfico (ver Figura *62*.- Medida de intensidad del tráfico agrupado por meses.Figura *62*), por lo que con los datos que tenemos no parece que el tráfico sea el factor determinante para que aumente en NO_2, aunque si tomamos como

referencia los datos de día de la semana (ver Figura *54*) y de horas del día (ver Figura *58*), si se puede ver que tienen bastante correlación las gráficas entre NO_2.

- PM25 y PM10:

Analizando los datos relativos a las PM25 y PM10 los tratamos conjuntamente al tener un comportamiento muy parecido. Podemos ver que teniendo en cuenta los datos agrupados por meses, existen unos datos más bajos en los meses de mayo, junio y noviembre (ver Figura *47* y Figura *48*) y al igual que en el caso de NO_2 estos gráficos no guardan correlación con los gráficos de intensidad del tráfico agrupados por meses o por día del mes (ver Figura *64*, ver y Figura *65*). Si ya tenemos en cuenta los gráficos de agrupación por días de la semana (ver Figura *54*) o por horas (ver Figura *58*), esta interpretación cambia, porque podemos ver que los gráficos tienen cierta similitud. Por lo tanto, si tomamos como referencia estos últimos datos, sí podemos decir que se aprecia cierta influencia en la intensidad del tráfico que hace subir o bajar los niveles de PM25 y PM10, cuando es más notable es en las horas nocturnas, que el tráfico se reduce a niveles muy bajos y los niveles de PM25 y PM10 comienzan a descender bastante con un pequeño retraso con respecto al tráfico y comienzan a subir cuando se reactiva el tráfico.

- O_3:

Para finalizar, comprobamos los datos que muestran las gráficas de O_3, si tenemos en cuenta los datos agrupados por meses (ver Figura *49*) se aprecia una subida en los meses de más calor con respecto a los más fríos y no se aprecia ninguna bajada en los meses de julio y agosto que son los que disminuye de manera significativa la intensidad del tráfico. Si pasamos directamente a los datos agrupados por semana (ver Figura *57*) se aprecia justamente a la inversa del tráfico, los sábados y domingos, que son cuando el tráfico disminuye, el valor de O_3 aumenta. Por último analizamos los datos agrupados por hora del día (ver Figura *61*), vemos que tiene el valor más bajo justamente en uno de los picos del tráfico que es sobre las 9 de la mañana y a partir de ese momento comienza a subir hasta las 5 de la tarde, hora en la que comienza el descenso sin encontrar una correlación con la intensidad del tráfico, podría parecer que tiene más influencia la temperatura ambiental, puesto que justamente en el gráfico de agrupación por meses,

tanto en la agrupación por horas del día, parece que podría seguir un patrón parecido a las meses más cálidos y a las horas del día más cálidas.

6. Conclusiones y trabajos futuros.

En este apartado se incluyen las principales conclusiones que se han podido obtener después de realizar todo el proceso del proyecto.

6.1. Comparativa algoritmos

Antes de comenzar con el análisis de los resultados para obtener las conclusiones hay que aclarar que las clases no estaban balanceadas totalmente, existe alguna clase predominante y da que pensar que los algoritmos de Naive Bayes (NB) pudieran obtener buenos valores en determinados casos. Teniendo en cuenta los resultados de la ejecución de los algoritmos de aprendizaje automático podemos concluir lo siguiente:

- NO_2:
  Podemos decir que, con este conjunto de datos y los algoritmos utilizados, los mejores resultados se obtienen utilizando Bosques Aleatorios y los peores con el algoritmo *Complement NB.*

- PM25:
  Con este conjunto de datos y los algoritmos utilizados, también los mejores resultados se obtienen utilizando Bosques Aleatorios, pero en este caso la diferencia es muy pequeña entre KNN y algunos algoritmos NB, los peores resultados, con diferencia, se obtienen con el algoritmo *Complement NB.*

- PM10:
  Con este conjunto de datos y los algoritmos utilizados, también los mejores resultados se obtienen utilizando Bosques Aleatorios y los peores también se obtienen con el algoritmo *Complement NB,* al igual que en el caso anterior, la diferencia entre los resultados de los algoritmos KNN y NB son muy pequeñas a excepción del *Complement NB.*

- O_3:

  Con este conjunto de datos y los algoritmos utilizados, también los mejores resultados se obtienen utilizando Bosques Aleatorios y con bastante diferencia, en este caso, los algoritmos KNN obtienen bastante mejores resultados que los algoritmos NB y los peores también se obtienen con el algoritmo *Complement NB* y en este caso en particular con unos resultados bastante peores, aunque le sucede a todos los algoritmos NB que tienen una bajada de resultados si los comparamos con los casos anteriores.

Hasta ahora, solamente se han tenido en cuenta los resultados del Accuracy, pero si analizamos los datos que nos muestran las matrices de confusión, vemos que los algoritmos de NB en la mayoría de los casos realizan sobreajuste debido a que los datos no se encuentran balanceados, por lo que algunos de los algoritmos en varios casos, solamente predicen 1 o 2 clases (ver Figura *68*, Figura *70* y Figura *72*) sobre todo se puede ver en Multinomial y Bernoulli, en estos casos, podríamos decir que a pesar de tener un Accuracy mejor que otros algoritmos, no sería bueno utilizarlos en predicciones. Los algoritmos KNN y Bosques Aleatorios, obtienen unas matrices de confusión con datos bien repartidos y realizan buenas predicciones para todas las clases distintas, teniendo en cuenta estos datos y los obtenidos con el Accuracy, podemos decir que son más adecuados para estos conjuntos de datos.

Teniendo en cuenta todos los resultados obtenidos con cada uno de los conjuntos de datos, podemos decir que claramente los algoritmos NB no funcionan bien, como podíamos intuir, debido a que normalmente tienen problemas cuando las clases de datos no se encuentran muy balanceadas, como es el caso. De todos los algoritmos, el *Complement* NB es, con bastante diferencia, el que peor comportamiento obtiene si tenemos en cuenta el Accuracy, pero como ya hemos comentado anteriormente, si se tienen en cuenta las matrices de confusión, ninguno de los algoritmos NB tienen un buen comportamiento y en todos los casos los mejores resultados los obtiene los Bosques Aleatorios, aunque en algunos casos KNN se acerca bastante. Dentro de los algoritmos KNN, en este TFM se han obtenido resultados muy parecidos independientemente del valor de K utilizado.

6.2. Comparativa del cuadro de mandos

Analizando los cuadros de mandos, se puede observar que si tenemos en cuenta los gráficos mensuales, ninguna de las medidas que se realizan para medir la calidad del aire guarda una clara relación entre las subidas y bajadas del tráfico.

Si tenemos en cuenta los datos de días del mes, no vemos que se cumpla ningún claro patrón en el tráfico, por lo que no podemos aplicar diferencias claras del tráfico por día del mes.

Cuando observamos los datos agrupados por día de la semana, podemos ver que las medidas de NO_2 guardan bastante similitud en gráficos con las alteraciones del tráfico, en los casos de las mediciones de PM25 y PM10, vemos que también se produce el mismo patrón. Si analizamos los datos de O_3 vemos que es justamente a la inversa, los sábados y domingos este valor aumenta.

Para finalizar, se comprueban las gráficas de las horas, en las que se puede ver perfectamente como aumenta y disminuye el tráfico dependiendo de la hora del día. Si lo comparamos con las medidas de calidad del aire, NO_2, PM25 y PM10 guardan una relación que aumenta y disminuye en función del tráfico, aunque en los dos últimos esto se produce con un pequeño retraso de alguna hora. Por el contrario, las mediciones de O_3 vemos que tienen muy poca relación, justamente cuando se está produciendo uno de los picos del tráfico es cuando este valor es el mínimo y a partir de ese momento comienza a subir hasta las 17 horas y luego vuelve a descender sin verse alterado por el segundo pico de tráfico que se produce por las tardes.

En conclusión, podemos de indicar que no parece que O_3 tenga una relación directa con el tráfico y para los otros valores, si tenemos en cuenta los datos mensuales, todo apunta que hay otros factores que no son dependientes del tráfico y que también influyen en el aumento de estos valores. Si analizamos los datos agrupados por día de la semana o por hora del día, vemos que alguna influencia si parecen tener.

6.3. Trabajos futuros

Como trabajos futuros, podemos separarlo en las tres partes principales en las que se puede dividir todo el proceso realizado:

- Obtención y preprocesado de datos.
  - ✓ Automatizar el proceso de descarga de los datos, realizando consultas cada x tiempo para comprobar si existen nuevos conjuntos de datos.
  - ✓ Incluir nuevos conjuntos de datos relacionados con otros aspectos que puedan influir en la calidad del aire, como pueden ser los meteorológicos, el número de árboles existentes en la ciudad, los tipos de vehículos que circulan, etc.
  - ✓ Se podría realizar todo este procesamiento utilizando soluciones en la nube tanto para el almacenamiento como para el procesado de los datos.
- Aprendizaje automático.
  - ✓ Utilizar otros modelos de aprendizaje y comparar los datos obtenidos con los que se han realizado en este trabajo.
  - ✓ Se podría realizar utilizando soluciones en la nube con la posibilidad de realizar escalado en caso necesario.
  - ✓ Comparar los modelos utilizando otros tipos de métricas.
- Visualización de los datos.
  - ✓ Realizar una visualización dinámica que vaya actualizando los datos en tiempo real.

Bibliografía

ATOTI_a. (2020). *atoti.io*. Obtenido de atoti.io: https://www.atoti.io/atoti-interactive-visualization-in-python-notebooks/

ATOTI_b. (2020). *atoti.io comenzando*. Obtenido de atoti.io: https://docs.atoti.io/latest/tutorial/tutorial.html#Getting-started

Bill Chambers, M. Z. (2018). *Spark: The Definitive Guide.*

Catálogo Datos Madrid. (Mayo de 2022). Obtenido de Web Datos Madrid : https://datos.madrid.es/egob/catalogo.csv

Commons.wikimedia.org. (16 de Mayo de 2022). *Commons.wikimedia.org*. Obtenido de https://commons.wikimedia.org

Community.esri.com. (11 de Febrero de 2022). *Community.esri.com*. Obtenido de community.esri.com: https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128

Datos Abiertos Madrid. (Mayo de 2022). Obtenido de Web Datos Madrid: https://datos.madrid.es/portal/site/egob

Digital, M. d. (15 de Mayo de 2022). *datos.gob.es*. Obtenido de datos.gob.es: https://datos.gob.es/es/documentacion/guia-practica-de-introduccion-al-analisis-exploratorio-de-datos

Digital, S. d. (15 de mayo de 2022). *Análisis exploratorio de datos*. Obtenido de datos.gob.es: https://datos.gob.es/es/documentacion/guia-practica-de-introduccion-al-analisis-exploratorio-de-datos

Ec europa. (Mayo de 2022). Obtenido de ec europa: http://ec.europa.eu/transport/themes/urban/studies/doc/2010_12_ars_the_european_traveler.pdf

Ec.europa.eu transporte. (Mayo de 2022). Obtenido de ec.europa.eu: http://ec.europa.eu/transport/themes/urban/studies/doc/2012-04-urban-freight-transport.pdf

Es-academic.com. (Mayo de 2022). *Es-academic.com*. Obtenido de Es-academic.com: https://es-academic.com/dic.nsf/eswiki/1288404#:~:text=Es%20un%20caso%20especial%20de,2%20(%CE%B8%2F2)

Estado, B. O. (27 de enero de 2017). *BOE*. Obtenido de BOE: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2017-914

Eurlex. (21 de septiembre de 2005). Obtenido de eurlex europa: https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:52005DC0446&from=EN

Europea, C. (12 de diciembre de 2011). *eurlex*. Obtenido de eurlex europa: https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:32011D0850&from=es

Ley europea movilidad urbana. (Mayo de 2022). Obtenido de ley europea movilidad urbana: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32020L1057

Macalupu.com. (febrero de 2022). *Macalupu.com*. Obtenido de Macalupu.com: https://macalupu.com/archives/calcula-la-distancia-entre-dos-coordenadas-geograficas-en-excel-con-una-funcion/

Madrid, E. A. (2021). *madrid.es*. Obtenido de madrid.es: https://www.madrid.es/UnidadesDescentralizadas/Calidad/CARTAS%20SERVICIOS/SistemaCartasServicio/28%20CS_Calidad_aire/DEFINITIVA/CS_Calidaddelaire_FolletoCartel2021.pdf

Madrid, E. A. (2021). *madrid.es*. Obtenido de madrid.es: https://www.madrid.es/portales/munimadrid/es/Inicio/Medidas-especiales-de-movilidad/Protocolo-de-contaminacion/Protocolo-de-actuacion-para-episodios-de-contaminacion-por-dioxido-de-nitrogeno/?vgnextfmt=default&vgnextoid=fd8718cea863c410VgnVCM1000000b205a0

Madrid, E. A. (s.f.). *datos.madrid*. Obtenido de datos.madrid: https://datos.madrid.es/

Scikit-learn. (Mayo de 2022). *Scikit-learn.org*. Obtenido de Scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Tráfico, O. A. (19 de Mayo de 2022). *Sede DGT*. Obtenido de Sede DGT: https://sede.dgt.gob.es/es/vehiculos/distintivo-ambiental/#