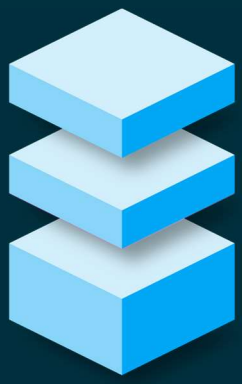


LAKE

Trabajo de Fin de Máster, Escuela profesional de nuevas tecnologías, CICE. Máster en Big Data

Álvaro Martínez, Guillermo Herranz, Marta Pérez, Rubén Márquez y Pablo Andreu

Madrid, 2021

INDICE

1. Planteamiento de la Empresa y Caso de uso.....4

1.1 *Planteamiento de la empresa.....5*

1.2 *Caso de uso.....6*

1.2.1 Visión.....6

1.2.2 Descripción.....6

1.2.3 Metas y objetivos.....6

1.2.4 Medidas de desempeño.....7

1.2.5 Desarrollo.....7

1.2.6 Rendimiento.....7

1.2.7 Suposiciones.....8

1.2.8 Restricciones.....8

1.2.9 Entorno tecnológico propuesto.....8

1.2.10 Hitos principales.....10

1.2.11 Evaluación.....11

1.2.12 Análisis de impacto de la organización.....11

1.2.13 Control de cambios.....12

2. Arquitectura del Proyecto.....13

3. Búsqueda y Datos.....15

3.1 *Descarga de datos de accidentes de tráfico.....15*

3.2 *Descarga de datos de estado meteorológico.....17*

4. Persistencia del Proyecto.....20

4.1 *Conexión con MongoDB Atlas.....20*

4.2 *Carga de datos con MongoDB Atlas.....21*

4.3 *Descarga de datos con MongoDB Atlas.....21*

5. Ingesta y Limpieza de Datos en Paralelo.....22

5.1 *Procesamiento en paralelo.....22*

5.1.1 Librería Dask.....23

5.1.2 Eficiencia de Dask.....26

5.2 *Conjunto de Datos de Personas Involucradas en Accidentes en la Ciudad de Madrid (2010 – 2020).....30*

5.3 *Conjunto de Datos del Estado Meteorológico Diario de la Ciudad de Madrid (2010 – 2020).....40*

5.4 *Conjunto de Datos de Personas Involucradas en Accidentes en la Ciudad de Madrid junto con el Estado Meteorológico.....43*

6. Visualización (Análisis exploratorio)……..46

7. Modelos Analíticos……..62

7.1 *XGBoost……62*

7.1.1 Matriz de confusión……64

7.1.2 Precisión del modelo……65

7.1.3 Importancia de variables……66

7.2 *Random Forest……67*

7.3 *Regresión Logística……68*

8. Glosario……70

9. Referencias……71

1. Planteamiento de la Empresa y Caso de uso

Trabajando en equipo y buscando cada uno un par de ideas, hicimos un brain storming. Este nos llevó a elegir la idea de crear un algoritmo que nos calculara las mejores rutas de los autobuses de un municipio pequeño y con los datos del aforo, nos calculara cuáles autobuses eran necesarios para conseguir un ahorro económico y energético en el municipio.

Solicitamos los datos del aforo de la comunidad de Madrid, pero no nos los concedieron, con lo que tuvimos que hacer una nueva reunión para presentar nuevas ideas. En esta reunión salió elegida la idea de realizar un algoritmo predictivo que aportara las horas más adecuadas a los streamers para realizar sus streams. El objetivo eran los streamers del juego FIFA 2021. Pero por problemas de acceso a datos de la plataforma tuvimos que concentrarnos en una nueva idea.

En la siguiente reunión teníamos claro que necesitábamos un Dataframe público y disponer de todos los datos que necesitáramos. Así que después de aportar cada uno sus ideas salieron elegido el proyecto de predicción de lesividad de los accidentes de tráfico de Madrid.

1.1 Planteamiento de la empresa

Lake es una empresa que ofrece servicios analíticos y estratégicos basados en el tratamiento de grandes cantidades de información.

La ciencia de datos está adquiriendo una dimensión de extrema relevancia para aquellas compañías que busquen desarrollarse, crecer y tener un conocimiento profundo de su entorno en 360º, es decir, todos los aspectos en los que intervenga una compañía determinada, desde lo más cercano como sus propios productos, hasta aquella información que pudiera estar más alejada como pudieran ser las estrategias de su competencia. Por estas razones creemos que la congregación de estos elementos y la aplicación de sus servicios a distintas compañías es un proyecto ambicioso, con altas expectativas de crecimiento y una utilidad abrumadora.

El primer proyecto de Lake centra sus esfuerzos en la detección de patrones que ayuden a prevenir, en cierta medida, la ocurrencia de incidentes de tráfico en la Comunidad de Madrid.

A través del Portal de datos abiertos del Ayuntamiento de Madrid, el objetivo principal es ser capaces de definir estrategias que ayuden a minimizar el impacto de los accidentes de tráfico que se produzcan en la Comunidad y sus respectivos distritos.

Mediante la acumulación de información proveniente del portal antes citado, el tratamiento de Big Data correspondiente y la realización de modelos predictivos, el objetivo será producir un algoritmo capaz de detectar anomalías, irregularidades y patrones que ayuden a reducir las repercusiones por la ocurrencia de accidentes de tráfico. Con los resultados que se esperan obtener creemos ser capaces de poder redirigir los medios sanitarios necesarios con respecto al tipo de incidente de tráfico que se produzca, pudiendo de esta manera, aumentar la eficiencia de medios sanitarios, reducir costes y lo más importante, llegar a salvar vidas.

1.2 Caso de uso

Uno de los medios de transporte más usados es el coche, en 2020 se ha cobrado 870 vidas como resultado de accidentes de tráfico, muchos de los cuales se podrían haber evitado. Existen zonas más peligrosas o con más probabilidad de accidentes, debido a las condiciones de la vía o de la fluidez del tráfico de la zona, por ejemplo. Debido a esto se hace necesario un estudio exhaustivo junto con modelos predictivos que ayuden a la administración a ser más eficaces en la gestión de los accidentes. Y les ayude en su camino por convertir Madrid en una Smart City.

1.2.1 Visión

Buscando un Madrid más seguro, aprovecharemos los datos en abierto para predecir futuros accidentes y así permitir a la administración usar sus medios de forma más eficiente, en el control de los accidentes de tráfico.

1.2.2 Descripción

El proyecto sienta sus bases en el análisis de los datos. Observando y analizando los datos de los accidentes de tráfico de los últimos 10 años, pretendemos ofrecer una evaluación de los accidentes de tráfico. Buscando responder a preguntas como: ¿cuándo se produce el mayor número de accidentes?, ¿dónde se producen?, ¿cuáles son las zonas más peligrosas?, ¿dónde se produce un mayor número de atropellos?, ¿cómo son los accidentes con ciclistas?, etc.

A demás realizaremos unos modelos predictivos que nos arrojen información sobre qué día de la semana y hora se produce el mayor número de accidentes y en qué zonas.

Para todo ello trabajaremos con herramientas de análisis de datos y visualización como Python, R, Power BI, etc.

1.2.3 Metas y objetivos

El objetivo del proyecto se centra en el estudio de los datos de tráfico para encontrar insights que muestren la mayor información posible de los accidentes de tráfico.

Se pretende hacer crecer la eficiencia de la respuesta de la administración frente a los accidentes de tráfico. Mediante la exposición de los datos y la información que nos revelan. Así como mediante modelos predictivos que avancen datos relevantes sobre estos accidentes.

1.2.4 Medidas de desempeño

Dentro de este apartado analizaremos cuales son los KPIs (Key Performance Indicators) que mejor se alinean con nuestro proyecto e idea de negocio.

Dividiremos dichos indicadores en dos zonas, una proveniente al desarrollo del proyecto y otra que haga referencia al rendimiento del proyecto una vez puesto en marcha.

1.2.5 Desarrollo

- Obtener toda la información relacionada con los accidentes de tráfico de Madrid
- Mantener actualizada la información antes descrita.
- Realizar un modelo analítico y predictivo plenamente funcional, útil y escalable a zonas demográficas más amplias, dónde la población este más concentrada.

1.2.6 Rendimiento

- Detectar días de la semana cuando se producen mayor número de accidentes.
- Detectar franjas horarias dónde se producen mayor número de accidentes.
- Detectar zonas conflictivas con mayor número de accidentes.
- Modelos predictivos que nos avancen esta información.
- Detectar información útil sobre peatones y ciclistas.

1.2.7 Suposiciones

Dentro de este apartado detallaremos los ámbitos que nuestro proyecto pretende impactary los objetivos que idealmente se podrían alcanzar:

- Acuerdos con Administraciones públicas.
- Detección a tiempo de accidentes graves.
- Escalabilidad del modelo predictivo.
- Internacionalización del proyecto.

1.2.8 Restricciones

Dentro del desarrollo del proyecto, existen varios factores delimitantes que pueden ser un factor clave en la evolución de este, o por el contrario restricciones que compliquen nuestrodesarrollo.

El acuerdo con las Administraciones públicas es el punto más relevante y el factor más delimitante, el punto de inflexión para poder poner en marcha el modelo a nivel real. Otra posible restricción es la posibilidad de tener problemas a la hora de recopilar datos, que no sean suficientes o que tengan algún tipo de error. Por último, es posible que la cantidad de datos recolectados no sea lo suficientemente grande, para que el modelo predictivo no sealo suficientemente preciso y por tanto no resulte útil.

1.2.9 Entorno tecnológico propuesto

En cuanto a los aspectos tecnológicos que serán necesarios para el desarrollo del proyectoe implementación del producto, se podrán dividir en dos campos:

- Software propuesto.
    - Sistemas operativos
        - Windows 10/7
        - Distribuciones de Linux
    - Editores de código
        - PyCharm [1]: Entorno de desarrollo integrado diseñado para la programación en Python.
        - Spyder [2]: Entorno de desarrollo integrado de libre acceso orientado a la programación en Python para el análisis de datos.

- Jupyter Notebook [3]: Entorno de desarrollo interactivo basado en la web para la programación en Python con el uso de celdas de ejecución.
- Google Colaboratory [4]: Entorno de desarrollo en la nube de Google orientado a la programación en Python con el uso de celdas de ejecución.
- R Studio [5]: Entorno de desarrollo integrado para la programación en R, orientado a las estadísticas y creación de gráficas.

  - Lenguajes de programación
    - Python
    - R
  - Gestores de bases de datos
    - Mongo DB [6]: Sistema de base de datos NoSQL, orientado a documentos y de código abierto.
  - Librerías
    - Pandas [7]: Biblioteca software que ofrece la manipulación y análisis de datos mediante la creación de estructuras de datos y operaciones.
    - Dask [8]: Biblioteca software orientada a la programación y ejecución de tareas utilizando procesamiento en paralelo.
    - Scikit-Learn [9]: Mayor librería en Python para la programación de herramientas de aprendizaje automático.
    - Selenium [10]: Biblioteca software para la manipulación de entornos web de forma automatizada.
    - PyMongo [11]: Librería para la conexión y peticiones a bases de datos MongoDB.
  - Visualización
    - PowerBI [12]: Herramienta de Microsoft destinada a la elaboración de visualizaciones interactivas y proporcionar capacidades de inteligencia empresarial.

- Hardware propuesto
  - Equipos informáticos (sobremesa/portátil)
  - Diferentes periféricos (ratón, teclado, monitores, etc.)

1.2.10 Hitos principales

Dentro de los principales hitos de este proyecto, y teniendo en cuenta la índole de este, un Trabajo de Fin de Máster, las variables más relevantes son el tiempo, pues la fecha de entrega limite es inamovible.

Y el desarrollo y ejecución del modelo predictivo con la suficiente exactitud para poder convertirse en algo útil y de interés para las administraciones.
Desglosando el trabajo a realizar podemos enumerar estos hitos principales:

El proyecto lo vamos a guiar por las entregas que tenemos establecidas en el Máster:

- **Primera entrega 21 de enero**

  Los hitos principales de esta entrega son las reuniones cada viernes o domingo y lafinalización el 27 de la primera entrega

- **Segunda entrega 19 de febrero**

  Los hitos principales de esta entrega son las reuniones de cada viernes o domingo y la división de este tramo de trabajo en 3 hitos:

  1. Semana del 28 al 3 de enero de ingesta de datos
  2. Semanas del 4 al 17 de enero de procesamiento de los datos
  3. Semanas del 18 al 15 de febrero de la creación del modelo analíticos o modelos.

- **Tercera entrega 13 de abril**

  Los hitos principales de esta entrega serán las reuniones cada viernes o domingo y el trabajo del 16 de febrero al 13 de Abril en la presentación delos datos y la revisión del caso de Uso actualizando los cambios que hayan sido necesarios en el proyecto.

- **Cuarta entrega 14 de Mayo**

  Los hitos principales de esta entrega serán las reuniones cada viernes o domingo. Dejaremos esta parte para la preparación de la presentación final del proyecto y la posible inclusión de ideas para hacer crecer el proyecto.

1.2.11 Evaluación

Cumplimiento de regulaciones, nuestro proyecto tendrá que cumplir con esta regulación:

- Ley Orgánica de Protección de Datos de Carácter Personal: cualquier proyecto debe cumplir con esta regulación europea. En este caso es de suma importancia que los datos que usemos sean de dominio público o cuenten con el consentimiento de quien nos los haya proporcionado.

1.2.12 Análisis de impacto de la organización

En caso de sufrir algún incidente interno o externo en nuestra organización seguiremos estos pasos para analizar la gravedad del problema y sus posibles soluciones:

- Recopilación de información: se recopilará información a través de diferentes portales públicos, para comprobar si resulta viable llevar a cabo nuestro proyecto.
- Identificación de las funciones y de los procesos: se decidirá de forma conjunta cómo se procederá y quien asumirá cada rol en la recuperación del problema.
- Evaluación de impactos operacionales: se evaluará el grado de impacto de interrupción del problema con tres niveles: “operación crítica para el negocio”, “es una parte importante del negocio, pero no es crítica” o “es leve”.
- Establecimiento de los tiempos de recuperación: después de haber calculado su importancia, se evalúa la sensibilidad temporal de cada operación según estos tres indicadores:
- Objetivo de punto de recuperación (RPO): se asocia a la pérdida aceptable de laempresa durante el desastre.
- Objetivo de tiempo de recuperación (RTO): hace referencia a la cantidad de tiempo que la empresa tiene que gastar hasta restaurar la actividad normal.
- MTD: tiempo máximo de inactividad que puede soportar la organización.
- Optimización de los recursos: se realiza un plan de recuperación para optimizar el uso de los recursos en un negocio. Por ello, se procede a reconocer cuales son los recursos críticos, es decir, aquellos que son prioritarios e imprescindibles durante la etapa de recuperación.

- Enumeración de procesos alternos: se elabora un análisis de impacto del negocio enumerando procesos alternos que pueden reemplazar los procesos que son críticos durante el tiempo de recuperación, estas alternativas funcionarían de forma temporal para ayudar a superar la crisis.
- Generación de un Informe de Impacto de Negocio: se evalúa el impacto y sedeterminan los umbrales de riesgo ante las diferentes situaciones.

1.2.13 Control de cambios

Se han realizado varios cambios en el proyecto. Primero comenzamos con la idea de hacer un transporte más eficiente y respetuoso con el medioambiente. Analizando la frecuencia y el aforo de los autobuses de Villaviciosa de Odón. Los datos teníamos que solicitarlos a la Comunidad de Madrid que no nos los proporcionó. Por lo tanto, tuvimos que cambiar de idea, pilotamos una nueva idea sobre la plataforma Twich y el juego FIFA 2021. Para analizar los streams y ayudara los streamers a hacer sus directos más eficaces. Pero nos encontramos con problemas en el acceso a los datos necesarios con lo que tuvimos que cambiar nuevamente la dirección del proyecto. El nuevo proyecto, el cual finalmente realizamos es un análisis sobre los accidentes de tráfico en Madrid.

2. Arquitectura del Proyecto

El conjunto del proyecto se ha dividido utilizando el conjunto de puntos que se nos ofrece desde la línea del proyecto de CICE. De esta forma podemos diferenciar 5 grandes fases:

- ***Búsqueda y Datos:*** En esta parte del proyecto llevaremos a cabo la descarga de los datos que utilizaremos a lo largo del proyecto utilizando diferentes herramientas aprendidas durante el máster como Web Scraping o el uso de APIs, y utilizando el lenguaje de programación Python y Google Colab.

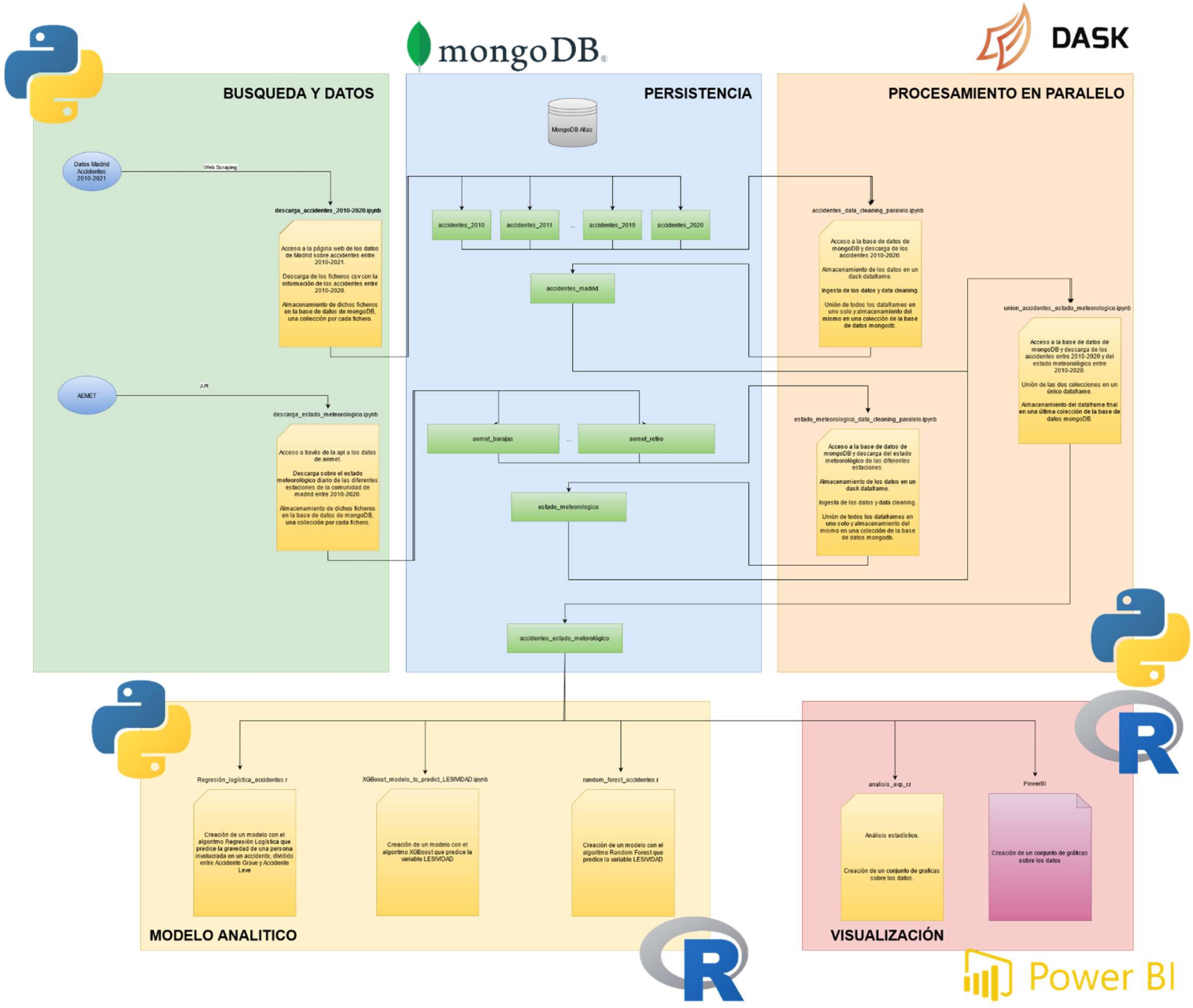
- ***Persistencia:*** Consiste en la parte de almacenamiento de los datos, para ello hemos decidido utilizar el gestor de bases de datos no relacional MongoDB, a través de un conjunto de colecciones.

- ***Procesamineto en Paralelo:*** Se ha realizado todo el proceso de limpieza y tratamiento de los datos, a través de diferentes notebooks escritos en Python y utilizando el procesamiento en paralelo de Dask.

- ***Modelo Analítico:*** Durante este proceso se ha llevado a cabo la construcción de dos modelos predictivos diferentes (uno escrito con Python y otro con R) con el objetivo de predecir la variable Lesividad de una persona involucrada en un accidente, además de la creación de un modelo predictivo más genérico para accidentes graves y leves.

- ***Visualización:*** Esta parte del proyecto consistía en la obtención y análisis estadístico de todo el conjunto de datos final, con el objetivo de descubrir cómo se encontraban los datos que íbamos a utilizar. Además, se han construido un conjunto de graficas para ayudar en el análisis.

A continuación, se muestra un diagrama con el resultado final de la arquitectura total del presente proyecto:


BUSQUEDA Y DATOS
PERSISTENCIA
PROCESAMIENTO EN PARALELO
MongoDB Atlas
Datos Madrid Accidentes 2010-2021
Web Scraping
descarga_accidentes_2010-2020.ipynb
Acceso a la página web de los datos de Madrid sobre accidentes entre 2010-2021.
Descarga de los ficheros csv con la información de los accidentes entre 2010-2020.
Almacenamiento de dichos ficheros en la base de datos de mongoDB, una colección por cada fichero.
accidentes_2010
accidentes_2011
accidentes_2019
accidentes_2020
accidentes_madrid
accidentes_data_cleaning_paralelo.ipynb
Acceso a la base de datos de mongoDB y descarga de los accidentes 2010-2020.
Almacenamiento de los datos en un dask dataframe.
Ingesta de los datos y data cleaning.
Unión de todos los dataframes en uno solo y almacenamiento del mismo en una colección de la base de datos mongodb.
union_accidentes_estado_meteorologico.ipynb
Acceso a la base de datos de mongoDB y descarga de los accidentes entre 2010-2020 y del estado meteorológico entre 2010-2020.
Unión de las dos colecciones en un único dataframe.
Almacenamiento del dataframe final en una última colección de la base de datos mongoDB.
AEMET
API
descarga_estado_meteorologico.ipynb
Acceso a través de la api a los datos de aemet.
Descarga sobre el estado meteorológico diario de las diferentes estaciones de la comunidad de madrid entre 2010-2020.
Almacenamiento de dichos ficheros en la base de datos de mongoDB, una colección por cada fichero.
aemet_barajas
aemet_retiro
estado_meteorologico
estado_meteorologico_data_cleaning_paralelo.ipynb
Acceso a la base de datos de mongoDB y descarga del estado meteorológico de las diferentes estaciones.
Almacenamiento de los datos en un dask dataframe.
Ingesta de los datos y data cleaning.
Unión de todos los dataframes en uno solo y almacenamiento del mismo en una colección de la base de datos mongodb.
accidentes_estado_meteorológico
Regresión_logistica_accidentes.r
Creación de un modelo con el algoritmo Regresión Logística que predice la gravedad de una persona involucrada en un accidente, dividido entre Accidente Grave y Accidente Leve
XGBoost_modelo_to_predict_LESIVIDAD.ipynb
Creación de un modelo con el algoritmo XGBoost que predice la variable LESIVIDAD
random_forest_accidentes.r
Creación de un modelo con el algoritmo Random Forest que predice la variable LESIVIDAD
MODELO ANALITICO
analisis_exp_1.r
Análisis estadístico.
Creación de un conjunto de graficas sobre los datos.
PowerBI
Creación de un conjunto de gráficas sobre los datos
VISUALIZACIÓN

3. Búsqueda y Datos

En primer lugar, para empezar a trabajar en nuestro Proyecto Final de Máster en Big Data, será necesario poseer un conjunto de datos relativamente extenso. Se pueden diferenciar dos grandes conjuntos de datos para el desarrollo del presente proyecto, uno relacionado con los accidentes y otro con el estado meteorológico.

3.1 Descarga de datos de accidentes de tráfico

El primer conjunto de datos consiste en datos de libre acceso y se encuentran en la página web Datos Madrid [13], la cual consisten en un portal de datos abiertos del ayuntamiento de la ciudad de Madrid, con el objetivo de promover el acceso a los datos del gobierno municipal e impulsar el desarrollo de herramientas creativas para atraer y servir a la ciudadanía de Madrid [14]. En este portal encontramos un registro de las personas implicadas en los accidentes ocurridos en la ciudad de Madrid y registrados por la Policía Municipal. Estos datos se encuentran desglosados a través de la página web Accidentes de tráfico de la Ciudad de Madrid por años, comenzando el registro en el año 2010 y finalizando en la actualidad de 2021, además de cada año se ofrece la descarga en formato csv y xlsx.

El proceso para descargar cada uno de los conjuntos de datos desde 2010 a 2020 (2021 no se utilizará debido a que es el año en curso y se encuentra en constante actualización) ha sido a través de la técnica Web Scraping [15], utilizando para ello la librería Selenium de Python. Para ello hemos desarrollado en un notebook de Google Colab (descarga_accidentes_2010-2020.ipynb) la descarga automática de los archivos en formato csv entre los años 2010 a 2020, accediendo al enlace de descarga de cada fichero. Una vez la descarga de cada fichero csv se ha completado, se procederá a leer uno a uno cada conjunto de datos en un dataframe de la librería Pandas, conversión a una lista de documentos json y almacenamiento en una colección de una base de datos de MongoDB Atlas [16].

Debido a una limitación en la inserción de documentos a la colección se insertarán las observaciones mes a mes. Además, en ciertos años, se han realizado algunas modificaciones sobre el nombre de las variables como por ejemplo los años 2016 y 2018 donde se modifica el nombre de la variable que incluye el número de víctimas para que coincida con los demás años, o el año 2020 donde se eliminan dos columnasque no dan valor a los datos.

El resultado final de la ejecución del presente notebook ha sido la creación de un total de 11 colecciones en la base de datos (una por cada año), en las cuales cada documento de cada colección representa a una persona involucrada en un accidente. La estructura de las colecciones es la siguiente:

Nombre de la colección	Número de muestras	Número de variables
accidentes_2010	26578	26
accidentes_2011	27342	26
accidentes_2012	26982	26
accidentes_2013	26839	26
accidentes_2014	27967	26
accidentes_2015	28172	26
accidentes_2016	29201	26
accidentes_2017	29795	26
accidentes_2018	30122	26
accidentes_2019	51806	14
accidentes_2020	32420	15

En total, sumando todas las colecciones obtenidas, contamos con un total de 337224 observaciones de personas involucradas en un accidente en la ciudad de Madrid entre los años 2010 a 2020. Antes de continuar, cabe mencionar una diferencia significativa entre los años y es que a partir del año 2019 la estructura de los datos se vio modificada, por lo que deberemos tener en cuenta este aspecto cuando se lleve a cabo la ingesta y limpieza de datos.

3.2 Descarga de datos de estados meteorológicos

Para el segundo conjunto de datos hemos optado por utilizar datos del estado meteorológico en el momento que ocurrieron los accidentes del primero conjunto de datos. Para ello se ha hecho uso de la página web de la Agencia Estatal de Meteorología (AEMET) [17], la cual proporciona una API [18] para el acceso a una amplia red de información relacionada con el estado meteorológico en España. De esta forma, para la descarga del segundo conjunto de datos se ha hecho uso de esta segunda técnica enseñada en el Máster, el uso de una API.

Al igual que en el proceso anterior se ha desarrollado un notebook en Google Colab (descarga_estado_meteorologico.ipynb), mediante el cual se han realizado peticiones GET a la API REST de AEMET utilizando la librería requests [19] de Python. Para este proceso fue necesario poseer un api_key, la cual se consiguió fácilmente a través de la página web de AEMET. Las peticiones se realizarán al endpoint ***"https://opendata.aemet.es/opendata/api/valores/climatologicos/diarios/datos/fech aini/<fechainicial>/fechafin/<fechafinal>/estacion/<codigoestacion>"***, este endpoint nos proporciona datos diarios generales del estado meteorológico en función de los parámetros proporcionados:

- ***fechainicial***: Cadena de texto con la fecha inicial del rango de días que solicitamos en formato AAAA-MM-DDTHH:MM:SSUTC.
- ***fechafinal***: Cadena de texto con la fecha final del rango de días que solicitamosen formato AAAA-MM-DDTHH:MM:SSUTC.
- ***codigoestacion***: Código de la estación meteorológica de la cual se solicitan losdatos.

Las peticiones se han realizado a las estaciones meteorológicas de Madrid, aunque estas son un total de 13 estaciones, muchas de ellas se encuentran fuera de la ciudad de Madrid, y debido a que solamente se poseen accidentes de la ciudad de Madrid, se han utilizado únicamente las estaciones que se encuentran dentro de la ciudad, las cuales son: 3129 (barajas), 3194U (ciudad universitaria), 3196 (cuatro vientos) y 3195 (retiro).

Para la implementación de las peticiones cabe mencionar que dicha API posee una limitación y es que no se pueden realizar peticiones de rangos de fechas superiores a 4 años, debido a que para nuestra solución se necesitan 10 años, se han realizado 3 peticiones por cada estación meteorológica (2010-01-01 a 2014-12-31, 2015-01-01 a 2019-12-31, 2020-01-01 a 2020-12-31). De esta forma, por cada estación se realizan 3 peticiones y se une el resultado de las 3 peticiones en un único dataframe Pandas. Finalmente, por cada conjunto de datos de cada estación se ha creado una colección en la base de datos de MongoDB Atlas donde se ha almacenado.

El resultado final de la ejecución del presente notebook ha sido la creación de un total de 4 colecciones en la base de datos (una por cada estación meteorológica de la ciudadde Madrid), en las cuales cada documento de cada colección representa un día con información sobre temperatura, viento, precipitación, etc.

La estructura de las colecciones es la siguiente:

Nombre de la colección	Número de muestras	Número de variables
aemet_barajas	4018	20
aemet_ciudad_universitaria	3851	15
aemet_cuatro_vientos	4018	20
aemet_retiro	4018	19

En total sumando todas las colecciones obtenidas, contamos con un total de 15905 observaciones de días por estación meteorológica durante los 10 años. Cabe mencionar que todos los conjuntos de datos poseen el mismo número de muestras salvo la de ciudad universitaria, esto es debido a que por diferentes problemas técnicos dicha estación no proporciono datos ciertos días durante la década, por lo que habrá que tratar como obtendremos estos datos perdidos en los siguientes puntos. Además, el número de variables es cambiante entre las diferentes estaciones, debido a que ciertos datos no se miden en algunas estaciones por lo que, en el futuro tratamiento de los datos se deberá utilizar aquellas variables que sean conjuntas a todas las estaciones.

4. Persistencia del Proyecto

Para trabajar de forma conjunta con los datos disponibles por parte de todo el equipo, se decidió hacer uso de un servicio de almacenamiento de información en la nube como es MongoDB Atlas. Este es un servicio específico de MongoDB y por lo tanto implementa una base de datos no relacional, que almacena conjuntos de datos en la nube que pueden ser usados por cualquier usuario que haya sido acreditado para ello.

En esta aplicación se creó un equipo de trabajo formado por los integrantes del grupo al que se le dio acceso a un "clúster" formado por 3 "nodos", los cuales dan el servicio a las peticiones de los usuarios al clúster. En dicho clúster se ha creado una base de datos, en la cual se almacenan un conjunto de datos repartidos en "colecciones". Los conjuntos de datos del punto anterior han quedado registrados en dicha base de datos en colecciones diferentes, y a lo largo de este punto se crearán nuevas colecciones.

Cabe destacar que esta herramienta es gratuita y por lo tanto posee limitaciones, ya que solamente se permite almacenar hasta un máximo de 520 MB por cluster, aspecto que hemos tenido que tener en cuenta.

4.1 Conexión con MongoDB Atlas

Para realizar la conexión a nuestro clúster de MongoDB Atlas a través de los dos lenguajes de programación que han sido utilizados en el proyecto, ha sido necesario utilizar una librería para cada lenguaje, PyMongo para Python y mongolite [20] para R.

Para conectarnos al clúster ha sido necesario utilizar la siguiente cadena de texto con la conexión:

"mongodb+srv://<username>:<password>@cluster0.ryeld.mongodb.net/<database>?retryWrites=true&w=majority", los parámetros necesarios para la conexión son los siguientes:

- ***username***: nombre del usuario que desea conectarse al cluster y por lo tantodeberá poseer los permisos correspondientes.
- ***password***: contraseña correspondiente al usuario anterior.
- ***database***: nombre de la base de datos a la cual se desea conectar.

Cabe mencionar que MongoDB Atlas tiene implementado una lista de IPs que pueden conectarse al clúster, en nuestro caso debido a que somos varios integrantes con varios ordenadores y además se han utilizado herramientas en la nube como Google Colab, se ha optado por establecer el cluster a público y que cualquiera pueda acceder a dichos datos.

4.2 Carga de Datos con MongoDB Atlas

Una vez hemos realizado la conexión con el clúster de MongoDB Atlas, si deseamos almacenar un conjunto de datos en una colección, en primer lugar, deberemos crear una colección y a continuación proceder a añadir los datos utilizando la función insertMany [21]. Para utilizar esta función es necesario que los datos se encuentren en formato array, en donde cada elemento del mismo se debe corresponder a un documento en formato json (en Python se correspondería con una estructura diccionario). Cabe destacar que hay una limitación del número de documentos que se pueden añadir con la función insertMany, por lo que es posible que en ciertas ocasiones sea necesario repartir la carga en diferentes partes.

4.3 Descarga de Datos con MongoDB Atlas

Por otro lado, para descargar los datos de las diferentes colecciones de la base de datos de MongoDB Atlas, se deberá de hacer colección a colección y utilizando la función find [22], la cual devuelve el total de documentos json de la colección correspondiente. Este resultado, deberá estructurarse en formato dataframe y por último se eliminará la columna _id, la cual se corresponde con el ObjectID que genera MongoDB por cada documento y el cual no aporta valor a nuestro conjunto de datos.

5. Ingesta y Limpieza de Datos en Paralelo

Como hemos indicado en el punto anterior de Búsqueda y Datos, se han obtenido dos grandes conjuntos de datos. El conjunto de individuos involucrados en accidentes de la ciudad de Madrid repartidos por años desde 2010 hasta 2020 y el estado meteorológico entre los años 2010 y 2020 repartidos entre las 4 estaciones de la ciudadde Madrid.

Estos dos grandes conjuntos de datos al estar repartidos en diferentes partes, no se podrá realizar un análisis y la creación del correspondiente modelo predictivo sin antesgenerar un único conjunto de datos que englobe todos estos datos. Para ello, este proceso de ingesta se ha dividido en tres pasos:

- Creación del conjunto de datos de individuos involucrados en accidentes de la ciudad de Madrid entre 2010 a 2020, para ello se deberá formatear cada variablede cada año a un formato único y realizar la unión de los datos de los diferentesaños.
- Creación del conjunto de datos de información meteorológica de diferentes estaciones de la ciudad de Madrid entre 2010 a 2020, para ello se deberá elegir las variables comunes que tienen todas las estaciones y por lo tanto se utilizarán más adelante, y realizar la unión de los datos de las diferentes estaciones.
- Creación del conjunto de datos final con la información de los individuos involucrados en los accidentes de la ciudad de Madrid entre 2010 a 2020 junto con el estado meteorológico del día en el que se produjo el accidente.

5.1 Procesamiento en Paralelo

Para realizar el proceso anteriormente descrito de ingesta y limpieza de datos, se ha optado por utilizar un procesamiento en paralelo, de esta forma se utilizarán diferentes particiones en donde se procesarán los datos al mismo tiempo. Este proceso de ingesta y limpieza de datos se ha realizado a través del lenguaje de programación Python mediante la implementación de diferentes notebooks en Google Colab, por lo que se ha utilizado una librería Dask que da las herramientas para realizar un procesamiento en paralelo.

5.1.1 Librería Dask

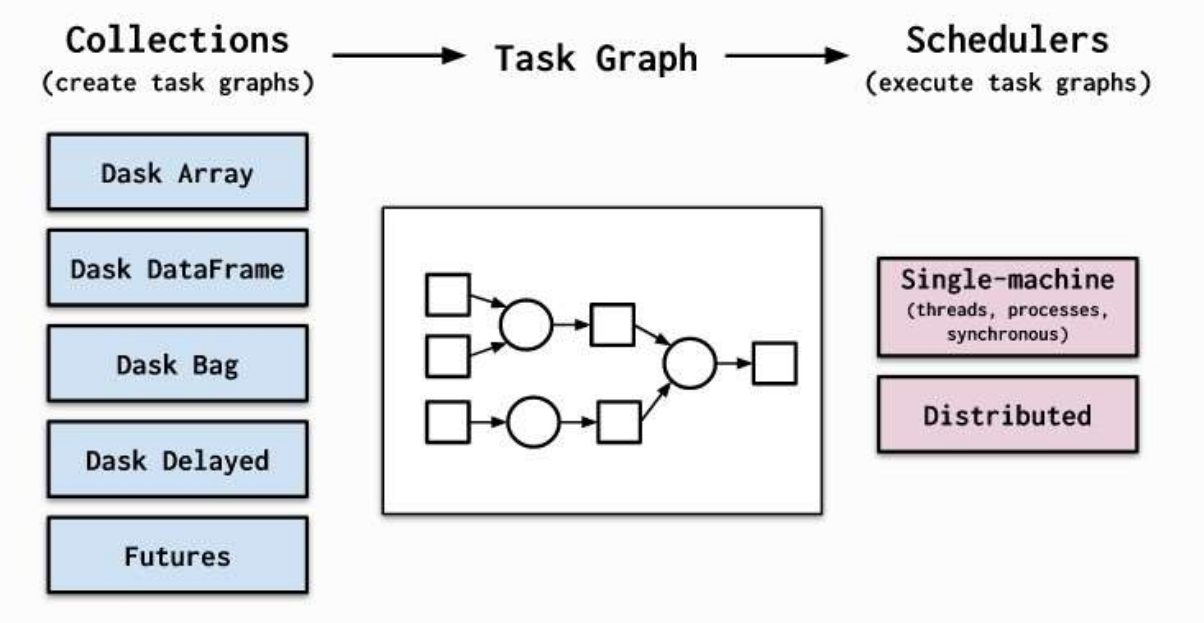
Dask consiste en una librería de Python que ofrece diferentes herramientas de paralelismo avanzado para el análisis de datos, lo que permite un alto rendimiento. Dask está compuesta de dos grandes partes [23]:

- Programación dinámica de tareas: Utilizado para optimizar diferentes procesosque requieren de un elevado costo computacional.
- Colecciones Big Data: Como pueden ser arrays, dataframes o listas que tienen la peculiaridad que se encuentran paralelizadas y por lo tanto las tareas que realicemos sobre estas colecciones se realizarán al mismo tiempo sobre todo el conjunto.

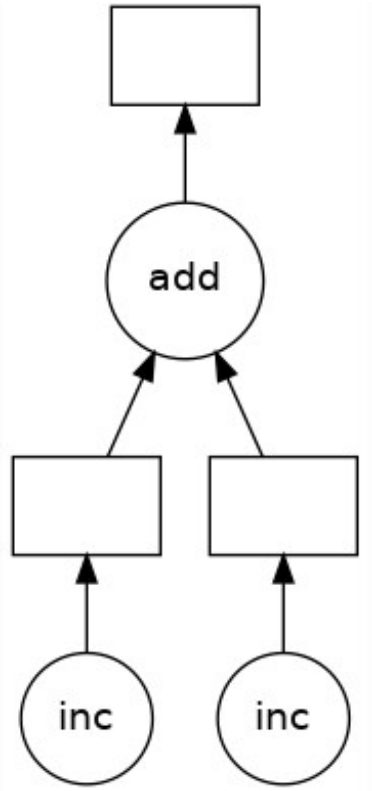
Dask posee 6 grandes virtudes [24]:

- ***Familiar***: hace uso de APIs y estructuras de datos altamente conocidas en la comunidad de Python, como pueden ser los dataframes de Pandas o los arraysde NumPy.
- ***Flexible***: proporciona una interfaz de organización de tareas para cargas de trabajo personalizadas y para la integración en proyectos.
- ***Nativo***: permite la computación distribuida en Python puro.
- ***Rápido***: funciona con baja sobrecarga, baja latencia y mínima serializaciónnecesaria para los algoritmos numéricos.
- ***Escalable***: es capaz tanto de ejecutarse de forma resiliente en clusters con milesde núcleos, como en ordenador portátil en un solo proceso.
- ***Capacidad de respuesta***: proporciona una respuesta rápida y diagnósticos paraayudar a los humanos.

Esta última virtud va ligada a la forma de trabajar que tiene Dask a bajo nivel, ya que tiene implementada la lazy execution [25] o ejecución perezosa en español. Esto significa que cuando se crea una colección en Dask y comenzamos a realizar tareas, estas tareas realmente no se están ejecutando, sino que se está construyendo lo que se conoce como task graph o grafo de tareas en español. El task graph consiste en un grafo/árbol donde cada nodo representa una tarea y las relaciones el flujo de datos, este grafo representa el flujo de ejecución de la colección para que el procesamiento se pueda realizar en paralelo. En la siguiente figura se puede ver la estructura de ejecución de Dask. Además, de un ejemplo sencillo de task graph.



Dask Schedule Execution (https://docs.dask.org/en/latest/)



Dask example of parallel execution (https://tutorial.dask.org/01x_lazy.html)

Para realizar el procesamiento en paralelo correctamente se hará uso de la librería Dask, y en especial se utilizará la colección Dask dataframe, a través de la cual se crearán un conjunto de particiones. Estas particiones son equivalentes a dataframes de la librería Pandas. Sobre esta matriz de datos de Dask, se ejecutarán diferentes tareas y funciones, las cuales se realizarán simultáneamente sobre todos los dataframes de Pandas que la forman. Cabe destacar que las funciones que se pueden aplicar a los Dask dataframes, son equivalentes a las funciones de Pandas, por lo que se podrá hacer una limpieza de datos satisfactorio haciendo uso del procesamiento en paralelo.

Por último, se ha elegido Dask como herramienta de procesamiento en paralelo en comparación a otras herramientas de computación distribuida más populares como por ejemplo Apache Spark, porque está escrito en su totalidad en Python (que es el lenguaje de programación que hemos utilizado para la limpieza de datos), además resulta ser más ligero y fácil de utilizar y manejar. Sin embargo, Dask y Apache Spark posee varios elementos que comparten, por lo que no resultan ser herramientas tan diferentes [26].

5.1.2 Eficiencia de Dask

Antes de comenzar a realizar la limpieza de los datos en paralelo con la librería Dask, se ha optado por realizar un experimento que mida cuanto de eficiente resulta ser el procesamiento en paralelo en nuestro problema.

Para ello, se ha implementado un pequeño notebook en Google Colab (eficiencia_dask.ipynb), en el cual se realizan una serie de tareas (establecer formato de fecha a una variable y creación de un conjunto de nuevas variables) sobre el mismo conjunto de datos (personas implicadas en accidentes desde 2010 a 2018) utilizando diferentes formas de ejecución (tanto con el formato Pandas como con Dask). De cada versión de ejecución, se ha realizado una medición del tiempo de cómputo:

- ***Versión 1:*** Uso de Dask dataframe con la función map para la creación de las nuevas variables.

```
%%time

dask['FECHA'] = dd.to_datetime(dask['FECHA'], format = '%d/%m/%Y')
dask['DIA'] = dask['FECHA'].map(lambda fecha: fecha.strftime('%d'))
dask['MES'] = dask['FECHA'].map(lambda fecha: fecha.strftime('%m'))
dask['ANO'] = dask['FECHA'].map(lambda fecha: fecha.strftime('%Y'))
dask['DIA SEMANA'] = dask['FECHA'].map(lambda fecha: fecha.weekday())
madrid_festivos = holidays.CountryHoliday('ES', prov='MD')
dask['FESTIVO'] = dask['FECHA'].map(lambda fecha: fecha in madrid_festivos)
dask[['FECHA', 'DIA', 'MES', 'ANO', 'DIA SEMANA', 'FESTIVO']].compute()
```

```
CPU times: user 10.6 s, sys: 310 ms, total: 10.9 s
Wall time: 12.7 s
```

- ***Versión 2:*** Uso de Pandas dataframe con la función map para la creación delas nuevas variables.

```
%%time

pandas['FECHA'] = dd.to_datetime(pandas['FECHA'], format = '%d/%m/%Y')
pandas['DIA'] = pandas['FECHA'].map(lambda fecha: fecha.strftime('%d'))
pandas['MES'] = pandas['FECHA'].map(lambda fecha: fecha.strftime('%m'))
pandas['ANO'] = pandas['FECHA'].map(lambda fecha: fecha.strftime('%Y'))
pandas['DIA SEMANA'] = pandas['FECHA'].map(lambda fecha: fecha.weekday())
madrid_festivos = holidays.CountryHoliday('ES', prov='MD')
pandas['FESTIVO'] = pandas['FECHA'].map(lambda fecha: fecha in madrid_festivos)
pandas[['FECHA', 'DIA', 'MES', 'ANO', 'DIA SEMANA', 'FESTIVO']]
```

```
CPU times: user 7.62 s, sys: 154 ms, total: 7.77 s
Wall time: 7.81 s
```

- ***Versión 3:*** Uso de Dask dataframe con la función map_partitions [27] y una funciónde mapeo por cada nueva variable que se crea.

```
%%time

dask['FECHA'] = dd.to_datetime(dask['FECHA'], format = '%d/%m/%Y')
dask['DIA'] = dask.map_partitions(lambda df_p: df_p['FECHA']
                                  .apply(lambda fecha: fecha.strftime('%d')))
dask['MES'] = dask.map_partitions(lambda df_p: df_p['FECHA']
                                  .apply(lambda fecha: fecha.strftime('%m')))
dask['ANO'] = dask.map_partitions(lambda df_p: df_p['FECHA']
                                  .apply(lambda fecha: fecha.strftime('%Y')))
dask['DIA SEMANA'] = dask.map_partitions(lambda df_p: df_p['FECHA']
                                         .apply(lambda fecha: fecha.weekday()))
madrid_festivos = holidays.CountryHoliday('ES', prov='MD')
dask['FESTIVO'] = dask.map_partitions(lambda df_p: df_p['FECHA']
                                      .apply(lambda fecha: fecha in madrid_festivos))
dask[['FECHA', 'DIA', 'MES', 'ANO', 'DIA SEMANA', 'FESTIVO']].compute()
```

```
CPU times: user 10.7 s, sys: 244 ms, total: 10.9 s
Wall time: 10.2 s
```

- ***Versión 4:*** Uso de Dask dataframe con la función map_partitions y una única función que mapeará con la función map por cada partición la creación de las nuevas variables.

```
madrid_festivos = holidays.CountryHoliday('ES', prov='MD')

def formatear_fecha(df):
  df['DIA'] = df['FECHA'].map(lambda fecha: fecha.strftime('%d'))
  df['MES'] = df['FECHA'].map(lambda fecha: fecha.strftime('%m'))
  df['ANO'] = df['FECHA'].map(lambda fecha: fecha.strftime('%Y'))
  df['DIA SEMANA'] = df['FECHA'].map(lambda fecha: fecha.weekday())
  df['FESTIVO'] = df['FECHA'].map(lambda fecha: fecha in madrid_festivos)

  return df
```

```
%%time

dask['FECHA'] = dd.to_datetime(dask['FECHA'], format = '%d/%m/%Y')
dask = dask.map_partitions(lambda df_p: formatear_fecha(df_p))
dask[['FECHA', 'DIA', 'MES', 'ANO', 'DIA SEMANA', 'FESTIVO']].compute()
```

```
CPU times: user 10.3 s, sys: 269 ms, total: 10.6 s
Wall time: 9.89 s
```

- ***Versión 5:*** Uso de Dask dataframe con la función map_partitions y una única función que mapeará con la función apply por cada partición la creación de las nuevas variables.

```
madrid_festivos = holidays.CountryHoliday('ES', prov='MD')

def formatear_fecha(df):
  df['DIA'] = df['FECHA'].apply(lambda fecha: fecha.strftime('%d'))
  df['MES'] = df['FECHA'].apply(lambda fecha: fecha.strftime('%m'))
  df['ANO'] = df['FECHA'].apply(lambda fecha: fecha.strftime('%Y'))
  df['DIA SEMANA'] = df['FECHA'].apply(lambda fecha: fecha.weekday())
  df['FESTIVO'] = df['FECHA'].apply(lambda fecha: fecha in madrid_festivos)

  return df
```

```
%%time

dask['FECHA'] = dd.to_datetime(dask['FECHA'], format = '%d/%m/%Y')
dask = dask.map_partitions(lambda df_p: formatear_fecha(df_p))
dask[['FECHA', 'DIA', 'MES', 'ANO', 'DIA SEMANA', 'FESTIVO']].compute()
```

```
CPU times: user 10.4 s, sys: 232 ms, total: 10.7 s
Wall time: 9.95 s
```

El resultado del experimento de eficiencia se puede resumir en la siguiente tabla:

Versión	*Descripción*	*Tiempo de ejecución*	*Orden de eficiencia*
1	Dask dataframe y map	12.7 s	5
2	**Pandas dataframe y map**	**7.81 s**	**1**
3	Dask dataframe y map_partitions con una función por cada nueva variable	10.2 s	4
4	Dask dataframe y map_partitions con una única función con map para la creación denuevas variables	9.89 s	2
5	Dask dataframe y map_partitions con una única función con apply para la creación denuevas variables	9.95 s	3

El resultado que obtenemos del presente experimento resulta ser que, para nuestro conjunto de datos, el cual no parece ser muy amplio, Dask no está mostrando todo supotencial y, por lo tanto, con una ejecución secuencial utilizando la librería Pandas obtenemos el tiempo de ejecución más rápido (versión 2 con 7.81 s). Sin embargo, aunque sería mejor ejecutar la limpieza de datos secuencialmente con Pandas, es necesario utilizar una herramienta de procesamiento en paralelo, aunque no sea necesario, por lo que se utilizará el formato de la versión 4 con el segundo tiempo de ejecución más rápido de 9.89 s, el cual utiliza Dask dataframe y la función map_partitions que realiza un mapeo de una tarea a todas las particiones de la matriz de datos.

5.2 Conjunto de Datos de Personas Involucradas en Accidentes en la Ciudad de Madrid (2010 - 2020)

En primer lugar, tendremos al conjunto de datos referente a los accidentes, aunque este conjunto de datos se encuentra dividido por año, podemos diferenciar dos grandes grupos de datos distintos. La información correspondiente a los años del 2010 hasta el 2018, está organizada de una forma distinta a la del 2019 al 2020. Por lo tanto, se deben procesar los dos grupos de datos de manera diferente para obtener un formato único, intentando perder el mínimo de información posible, para poder concatenarlos posteriormente y obtener una matriz de datos conjunta. Cada observación corresponde a un individuo distinto implicado en el accidente.

Este procesamiento de los datos se ha llevado a cabo con la implementación de un notebook en Google Colab (accidentes_data_cleaning_paralelo.ipynb), en el cual se desarrolla la limpieza de los datos, y la unión y creación del conjunto de datos de accidentes. En primer lugar, se ha realizado la conexión a la base de datos de MongoDB Atlas. A continuación, se han leído los conjuntos de datos de accidentes de las colecciones desde 2010 a 2018 y se han agrupado en un Dask dataframe con un total de 9 particiones (una por cada colección), se ha guardado en la variable DF1. Para los conjuntos de datos de accidentes de las colecciones desde 2019 a 2020, se han agrupado también en un Dask dataframe con un total de 2 particiones (una por cada colección), se ha guardado en la variable DF2.

A continuación, se explicará el tratamiento de los datos variable por variable, comentando como se ha llevado a cabo la limpieza de datos en ambos grupos de datos.

- ***FECHA***: para el posterior análisis de los datos va a ser de más utilidad extraer de la variable FECHA el día, el mes y el año. De esta forma, se crearán tres nuevas variables DIA, MES y ANO que harán referencia a los parámetros comentados anteriormente. Además, se incluirá una variable llamada FESTIVO que será un "booleano" que indique si el día correspondiente a la observación fue festivo en Madrid o no. La variable FECHA, a pesar de que no se usará en el tramo de creación de los modelos, será de máxima utilidad para unir el conjunto de datos del estado meteorológico posteriormente y por ello no será suprimida de la base de datos.
- ***RANGO HORARIO***: en cuanto al momento del día en el que se produjo el accidente, DF1 hace referencia con una variable llamada RANGO HORARIO en la que se identifica cada observación con un período de una hora en una hora. Por ejemplo, si un accidente se produjo a las 9:32, estará codificado como "DE 9:00 A 9:59". En cambio, en DF2 se especifica la hora exacta del siniestro en la variable HORA. Por lo tanto, se ha creado una variable nueva en DF2 llamada RANGO HORARIO, completada a partir de la hora exacta del accidente, en la que cada observación está incluida en un intervalo horario igual que en DF1.
- ***DIA SEMANA***: DF1 contiene una variable con este nombre que identifica cada observación con el día de la semana en la que se produjo. En DF2, debido a la no existencia de una variable que referencie esta información, se ha creado una nueva de nombre DIA SEMANA completada a partir de la fecha del incidente.

- ***DISTRITO***: existe una variable en ambos grupos de datos que informa acerca del distrito en el que se produjo el accidente. Existen un total de 21 distritos en Madrid: Centro, Arganzuela, Retiro, Salamanca, Chamartín, Tetuán, Chamberí, Fuencarral-El Pardo, Moncloa-Aravaca, Latina, Carabanchel, Usera, Puente de Vallecas, Moratalaz, Ciudad Lineal, Hortaleza, Villaverde, Villa de Vallecas, Vicálvaro, San Blas-Canillejas y Barajas.
- ***LUGAR ACCIDENTE***: la variable original en los dos grupos de datos contiene el nombre de la calle o carretera en la que se produjo el accidente. En caso de quefuera en un cruce, esta se completa con las dos calles que lo forman. El nombre de las calles no está normalizado en ambos conjuntos, las mismas calles pueden estar escritas de forma distinta, por lo que resulta imposible categorizaresta variable. No obstante, se puede extraer información relevante de esta. Se ha creado una nueva variable binaria llamada CRUCE que describe el hecho de que el lugar donde se haya producido el accidente sea un cruce o no.
- ***NUMERO***: Indica el número de la calle en la que se ha producido el accidente o en caso de ser una carretera el kilómetro. Dado que no se va a hacer uso de la variable que indica el nombre de la calle, carece de sentido hacer uso de esta variable.
- ***Nº PARTE***: el número del parte se ha codificado como "año/código_accidente". Aunque para la modelización no sea una variable de utilidad, resulta interesante mantenerla para hacer agrupaciones de accidentes. Ambos grupos de datos noposeen el mismo formato, ya que en el DF2 el símbolo "/" se sustituye por "S", por lo que se ha normalizado con el símbolo "/".

- ***ESTADO METEOROLÓGICO***: en DF2 existe una variable con este nombre que hace referencia al clima en el momento que tuvo lugar el accidente. Las categorías que lo conforman son: Granizando, Lluvia débil, Lluvia intensa, Nevando, Despejado, Nublado, Se desconoce y nan. Por otro lado, en DF1 existen variables binarias que identifican el estado meteorológico en el instante del accidente. Los nombres de estas son: CPFA Granizo, CPFA Hielo, CPFA Lluvia, CPFA Nieve, CPFA Niebla y CPFA Seco. Para poder combinar ambos grupos de datos y no perder información se ha decidido que la mejor manera de procesarla es agruparla en las variables binarias de DF1, añadiendo una llamada CPFA Desconocido, para las observaciones en las que se desconozca el estado. La conversión/mapeo de las variables de DF2 a DF1 ha sido la siguiente, de esta manera la variable con la que se mapea se pondrá a valor True y las demás se establecerán a False:

DF1	DF2
CPFA Granizo	Granizando
CPFA Lluvia	Lluvia débil, lluvia intensa
CPFA Nieve	Nevando
CPFA Seco	Despejado, nublado
CPFA Desconocido	Se desconoce, NaN

Cabe remarcar que DF1 incluye variables tituladas como “CPSV” que hacen referencia al estado de la vía. Esta información resulta imposible de obtener de DF2 por lo que serán eliminadas del conjunto de datos.

- ***TIPO ACCIDENTE***: gracias a esta variable se conoce qué clase de accidente sucedió. Ambos grupos de datos contienen categorías diferentes para esta variable, por lo que se hará una conversión de cada categoría de cada uno de los grupos de datos a las categorías definitivas de la variable:

Categoría TIPO ACCIDENTE	DF1	DF2
COLISION DOBLE	COLISIÓN DOBLE	Colisión fronto-lateral, Colisión lateral, Colisiónfrontal
COLISION MULTIPLE	COLISIÓN MÚLTIPLE	Colisión múltiple
CHOQUE OBSTACULO FIJO	CHOQUE CON OBJETO FIJO	Choque contra obstáculo fijo
CAIDA	CAÍDA MOTOCICLETA, CAÍDA CICLOMOTOR, CAÍDA VIAJEROBUS, CAÍDA BICICLETA, CAÍDA VEHÍCULO 3 RUEDAS	Caída
ATROPELLO	ATROPELLO	Atropello a persona
VUELCO	VUELCO	Vuelco
OTRO	OTRAS CAUSAS	Alcance, Atropello a animal, Otro, Solo salida de la vía, Despeñamiento
DESCONOCIDO	-	NaN

- ***TIPO VEHÍCULO***: contiene información acerca del tipo de automóvil partícipe enel accidente. De la misma forma que con la anterior variable, se ha categorizado la información para tener una normalización de los datos. Se ha seguido el modelo de DF1 en que los datos están agrupados en las siguientes categorías: TURISMO, NO ASIGNADO, FURGONETA, MOTOCICLETA, AUTO-TAXI, VARIOS, CAMION, CICLOMOTOR, AUTOBUS-AUTOCAR, BICICLETA, AMBULANCIA y VEH.3 RUEDAS. Por lo tanto, la conversión para DF2 es la siguiente:

DF1	DF2
TURISMO	Turismo, Todo terreno, Cuadriciclo no ligero, Cuadriciclo ligero
NO ASIGNADO	NaN, Sin especificar
FURGONETA	Furgoneta
MOTOCICLETA	Motocicleta hasta 125cc, Motocicleta > 125cc
AUTO-TAXI	-
VARIOS	Maquinaria de obras, Vehículo articulado, Otros vehículos con motor, Semirremolque, Remolque, Tranvía, Otros vehículos sinmotor, Tren/metro, Maquinaria agrícola
CAMION	Camión rígido, Tractocamión, Camión de bomberos
CICLOMOTOR	Ciclomotor, Ciclo, Ciclomotor de dos ruedas L1e-B
AUTOBUS	Autobús, Autobús articulado, Autocaravana, Microbús <= 17
BICICLETA	Bicicleta, VMU eléctrico, Patinete, Bicicleta EPAC (pedaleo asistido)
AMBULANCIA	Ambulancia SAMUR
VEH.3 RUEDAS	Moto de tres ruedas hasta 125cc, Moto de tres ruedas > 125cc, Ciclomotor de tres ruedas

- ***TIPO PERSONA***: esta variable indica si el implicado en el accidente se trata delconductor, un pasajero, un peatón o un testigo. DF2 no contiene ningún testigo.
- ***SEXO***: en esta columna se hace referencia al sexo de las personas implicadasen el accidente. Existen las categorías HOMBRE, MUJER y NO ASIGNADO.
- ***RANGO EDAD***: esta variable indica la edad de cada persona implicada en el accidente. Esta categorizada de la misma forma en ambos grupos de datos, salvo para una de las categorías del DF1, la cual se ha convertido de DE MAS DE 74 AÑOS a MAYOR DE 74 AÑOS. Los intervalos de edad son los siguientes: De 0 a 5 años, De 6 a 9 años, De 10 a 14 años, De 15 a 17 años, De 18 a 20 años, De21 a 24 años, De 25 a 29 años, De 30 a 34 años, De 35 a 39 años, De 40 a 44 años, De 45 a 49 años, De 50 a 54 años, De 55 a 59 años, De 60 a 64 años, De 65 a 69 años, De 70 a 74 años, Mayor de 74 años y Desconocida.

- ***LESIVIDAD***: se indica la gravedad de la lesión de cada individuo. En DF1 se encuentra agrupada en las cinco siguientes categorías: MT (muerto), HG (herido grave), HL (herido leve), IL (ileso) o NO ASIGNADA. Por otro lado, en DF2 existe una agrupación distinta basada en unos códigos que se muestran acontinuación:
    - 1: Atención en urgencias sin posterior ingreso
    - 2: Ingreso inferior o igual a 24 horas
    - 3: Ingreso superior a 24 horas
    - 4: Fallecido 24 horas
    - 5: Asistencia sanitaria ambulatoria con posterioridad
    - 6: Asistencia sanitaria inmediata en centro de salud o mutua
    - 7: Asistencia sanitaria sólo en el lugar del accidente
    - 14: Sin asistencia sanitaria
    - 77: Se desconoce
    - En blanco: Sin asistencia sanitaria

  Se ha decidido modificar estas últimas categorías para obtener un formato igual al de DF1 siguiendo las indicaciones de la documentación del Ayuntamiento de Madrid sobre los datos. Se han producido los siguientes cambios:

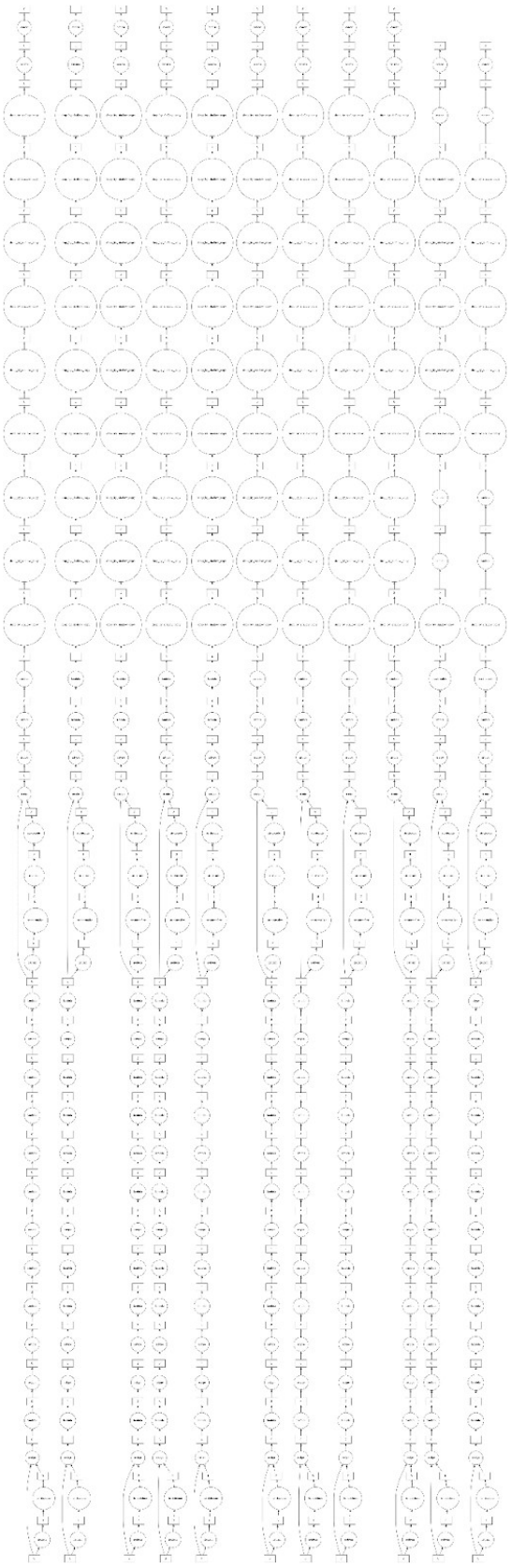
DF1	DF2
MT (muerto)	4
HG (herido grave)	3
HL (herido leve)	1, 2, 5, 6, 7
IL (ileso)	14, en blanco
NO ASIGNADA	77

- ***Nº VÍCTIMAS***: esta variable únicamente existe en DF1 e indica el número de víctimas que hubo en cada accidente. Como víctimas se cuentan a heridos leves, herido graves y muertos; los ilesos o no asignados no se computan para esta variable. Dado la relevancia de su información, la variable ha sido creada en DF2 realizando en primer lugar un agrupamiento con la función groupby y utilizando la variable Nº PARTE, a continuación, se ha realizado un conteo de las observaciones que contenían en la variable LESIVIDAD los valores HL, HG o MT. En el segundo grupo de datos aparecen accidentes sin ningún lesionado a diferencia del primero en el que en todos los accidentes había al menos una persona herida.
- ***ESTACION METEOROLOGICA***: de forma adicional, se creará una nueva columna que contenga un indicador con la estación meteorológica más cercana al distrito en el que se ha producido el accidente. De esta forma se podrán añadir con mayor facilidad la información correspondiente al estado meteorológico obtenida de los servicios de AEMET. Las asignaciones propuestas para cada estación han sido las siguientes:

ESTACIÓN METEOROLÓGICA CERCANA	DISTRITO
MADRID, RETIRO	ARGANZUELA, CARABANCHEL, CENTRO, CIUDAD LINEAL, HORTALEZA, MORATALAZ, PUENTE DE VALLECAS, RETIRO, SALAMANCA, USERA y VILLAVERDE
MADRID AEROPUERTO	BARAJAS, SAN BLAS-CANILLEJAS, VICALVARO y VILLA DE VALLECAS
MADRID, CIUDAD UNIVERSITARIA	CHAMARTIN, CHAMBERI, FUENCARRAL-EL PARDO, MONCLOA-ARAVACA y TETUAN
MADRID, CUATRO VIENTOS	LATINA
DESCONOCIDO	DESCONOCIDO

Después de todo este proceso de limpieza de datos, procedemos a eliminar un conjunto de variables innecesarias para cada uno de los conjuntos de datos y a renombrar otras variables para que coincidan en el nombre. De esta forma, tendremos en ambos grupos de datos la misma estructura de variables, por lo que podremos realizar una concatenación con la función concat y obtendremos el Dask dataframe definitivo con la unión de todas las observaciones.

Como hemos indicado anteriormente, Dask utiliza la llamada ejecución perezosa, por lo que aún no hemos llevado a cabo ningún procesamiento sobre los datos, sino que se han quedado almacenados el conjunto de procesos que queremos realizar. Para ejecutarlo paralelamente sobre todos los datos, utilizaremos la función compute sobre el Dask dataframe unificado, y obtendremos como resultado un dataframe de Pandas con la unificación. El task graph del procesamiento ha sido el siguiente, en el podemos diferenciar las 9 particiones del primer grupo de datos y las 2 últimas particiones del segundo grupo, y empezando desde abajo todas las tareas realizadas:



Finalmente, el conjunto de datos obtenido se ha guardado en la base de datos de MongoDB Atlas, dentro de una colección llamada accidentes_madrid. El conjunto de datos resultante cuenta con un total de 337224 observaciones y 25 variables diferentes. Como información añadida, se incorpora en los Anexos los documentos que informan el conjunto de variables para cada conjunto de datos proporcionado por el ayuntamiento de Madrid.

5.3 Conjunto de Datos del Estado Meteorológico Diario de la Ciudad de Madrid (2010 – 2020)

En segundo lugar, tendremos el conjunto de datos referente a la información meteorológica diaria en la ciudad de Madrid dividido por estación meteorológica. Dicha información deberá ser procesada seleccionando únicamente aquellas variables que aporten valor y sean conjuntas a todas las estaciones meteorológicas. Posteriormente, se concatenarán los 4 conjuntos de datos referentes a las 4 estaciones meteorológicas de Madrid para obtener un único conjunto de datos. Cada observación se corresponde a un día-estación.

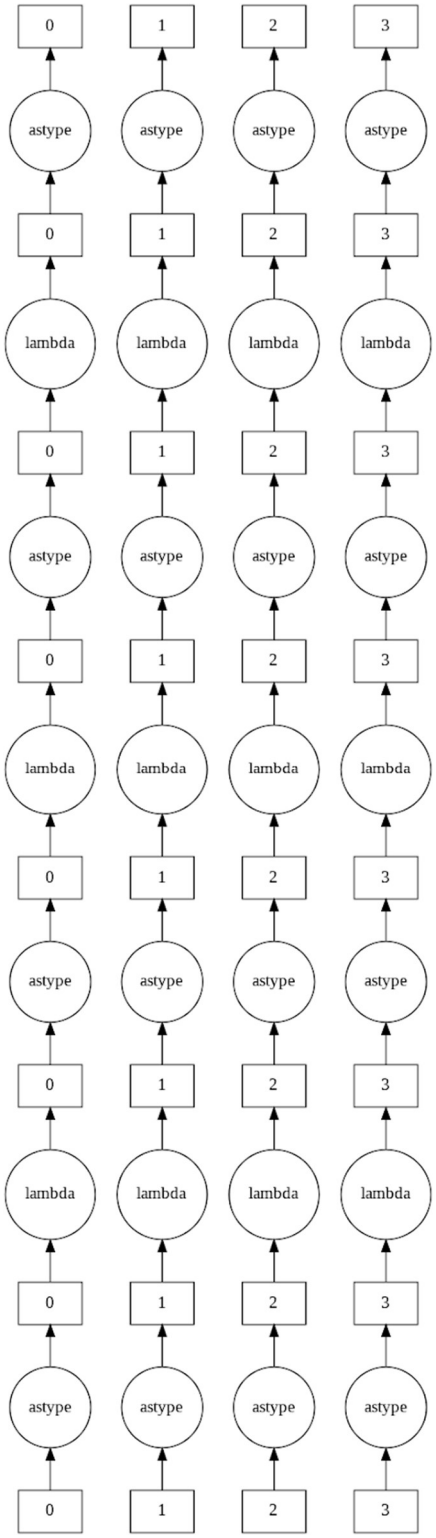
Este procesamiento de los datos se ha llevado a cabo con la implementación de un notebook en Google Colab (estado_meteorologico_data_cleaning_paralelo.ipynb), en el cual se desarrolla el procesamiento de los datos, y la unión y creación del conjunto de datos de estado meteorológico. En primer lugar, se ha realizado la conexión a la base de datos de MongoDB Atlas. A continuación, se han leído los conjuntos de datos de información meteorológica de las colecciones de las 4 diferentes estaciones meteorológicas de la ciudad de Madrid, y se han agrupado en un Dask dataframe con un total de 4 particiones (una por cada colección), se ha guardado en la variable df.

Estos conjuntos de datos contienen varias variables como pueden ser altitud, tmin (temperatura mínima del día), sol, presMax (presión máxima del día), etc. Sin embargo, como se ha indicado anteriormente solo se escogerán aquellas variables que veamos que aportarán valor a nuestro modelo sobre el estado meteorológico en los días que se produjeron los accidentes. Estas variables son las siguientes:

- ***fecha***: fecha del día de la medición de la información, se utilizará para hacer la unión con el conjunto de datos de accidentes.
- ***nombre***: nombre de la estación meteorológica donde se produjo la medición dela información, se utilizará para hacer la unión con el conjunto de datos de accidentes.
- ***tmed*** (Temperatura media): indica la temperatura media en grados Celsius del día especifico en la estación determinada.
- ***prec*** (Precipitación): cantidad de agua caída a lo largo del día calculada en milímetros (o litros caídos por unidad de superficie).
- ***velmedia*** (Velocidad media del viento): velocidad media del viento a lo largo deldía en m/s.

Una de las transformaciones que se ha aplicado sobre estas es cambiar el valor “Ip” de “prec”, que indica que es inferior a 0.1 mm, por el número 0 para obtener una variable numérica en su completitud. Además, las tres últimas variables consisten en valores decimales, sin embargo, estos datos utilizaban una coma para representar el valor decimal, por lo que se ha sustituido en todos sus valores la coma por el punto decimal,para no tener problemas en un futuro.

Al igual que en el proceso de limpiado del conjunto de datos de accidentes, aquí también tenemos que computar la ejecución de todas las tareas realizadas, debido a la ejecución perezosa. El task graph del procesamiento ha sido el siguiente, en el cual podemos diferenciar las 4 particiones diferentes (una por cada estación):


0
1
2
3
astype
lambda

Finalmente, el conjunto de datos obtenido se ha guardado en la base de datos de MongoDB Atlas, dentro de una colección llamada estado_meteorologico. El conjunto de datos resultante cuenta con un total de 15905 observaciones y 5 variables diferentes.

5.4 Conjunto de Datos de Personas Involucradas en Accidentes en la Ciudad de Madrid junto con el Estado Meteorológico (2010 – 2020)

El último paso para obtener la matriz de datos definitiva con la que se trabajará es unirlos conjuntos de datos anteriormente creados, correspondientes a la información de los accidentes y el estado meteorológico.

Este procesamiento de unificación de los datos se ha llevado a cabo con la implementación de un notebook en Google Colab (union_accidentes_estado_meteorologico.ipynb), en el cual se desarrolla la unión de losdos conjuntos de datos uno único. En primer lugar, se ha realizado la conexión a la base de datos MongoDB Atlas. A continuación, se ha leído el conjunto de datos referente a los accidentes de la colección accidentes_madrid y se ha almacenado en un Dask dataframe en 11 particiones, llamada dd_accidentes. Por otro lado, se ha leído el conjunto de datos de la información meteorológica de la colección estado_meteorologico y se ha almacenado en un Dask dataframe en 4 particiones, llamada dd_meteo.

Para llevar a cabo el proceso de unificación se ha realizado utilizando la función mergede Dask, a través de la siguiente instrucción:

```
dd_accidentes_meteo = dd.merge(dd_accidentes, dd_meteo, how='left', left_on=['FECHA','ESTACION METEOROLOGICA CERCANA'], right_on = ['fecha','nombre'])
```

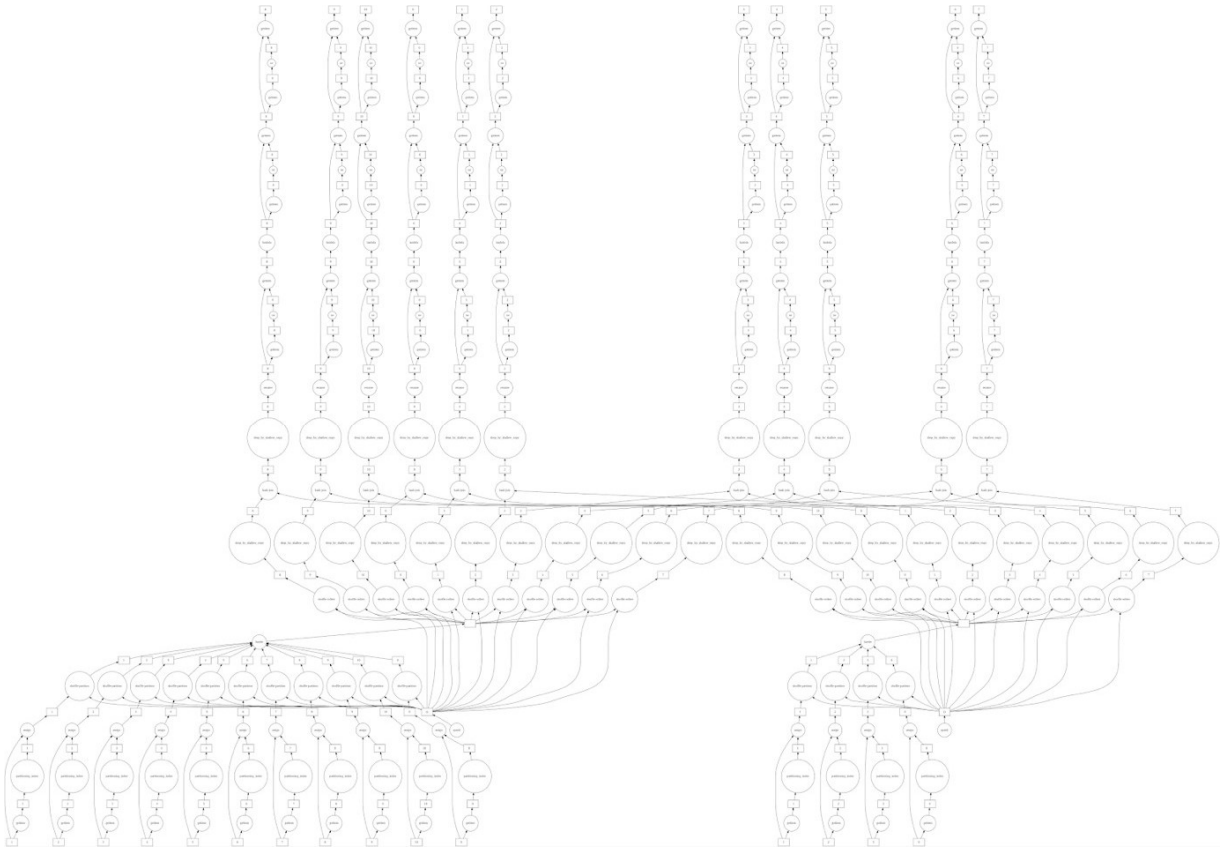
Como podemos comprobar se realiza una unión por la parte izquierda, es decir por el conjunto de datos de accidentes, utilizando para ello las variables FECHA – fecha y ESTACION METEOROLOGICA CERCANA – nombre. En dd_accidentes_meteo tendremos almacenado el conjunto de datos unificado con accidentes y estado meteorológico.

De esta forma, se han añadido al conjunto de datos de los accidentes las columnas “TEMPERATURA MEDIA”, “PRECIPITACION” y “VELOCIDAD MEDIA VIENTO” que harán
referencia a las variables “tmed”, “prec” y “velmedia” del segundo “dataset” respectivamente. En caso de que el valor de este sea DESCONOCIDO, se ha optado poreliminar estas observaciones debido a que tan solo eran 7 en total.

Otro procesamiento que se ha llevado a cabo es en las observaciones en las que alguna de las variables tmed, prec o velmedia contengan el valor NaN. Esto es debido a que, como se indicó anteriormente, la estación de Ciudad Universitaria hubo días que no realizado mediciones. Para crear artificialmente estos valores desconocidos se ha agrupado el conjunto de datos de estado meteorológica por la variable fecha y se ha realizado la media de las variables anteriormente indicadas. A continuación, se ha establecido el valor medio de las demás estaciones para aquellos valores que se encuentran nulos.

En última instancia se ha optado por eliminar las observaciones que hacen referencia a accidentes sin víctimas, ya que estas solo están recogidas en los años 2019 y 2020 y suponen un cambio drástico en la distribución de la variable objetivo LESIVIDAD. Lógicamente suponen un aumento masivo en la categoría ILESO. Además, todas las observaciones con valor NO ASIGNADA en LESIVIDAD han sido a su vez eliminadas porque no tendría ningún sentido predecir cuando el valor de la variable respuesta es desconocido.

Como en los procesamientos anteriores, esta unificación también se ha realizado en paralelo, por lo que finalmente ejecutaremos todos los procesos establecidos anteriormente. El task graph de la ejecución ha sido el siguiente:



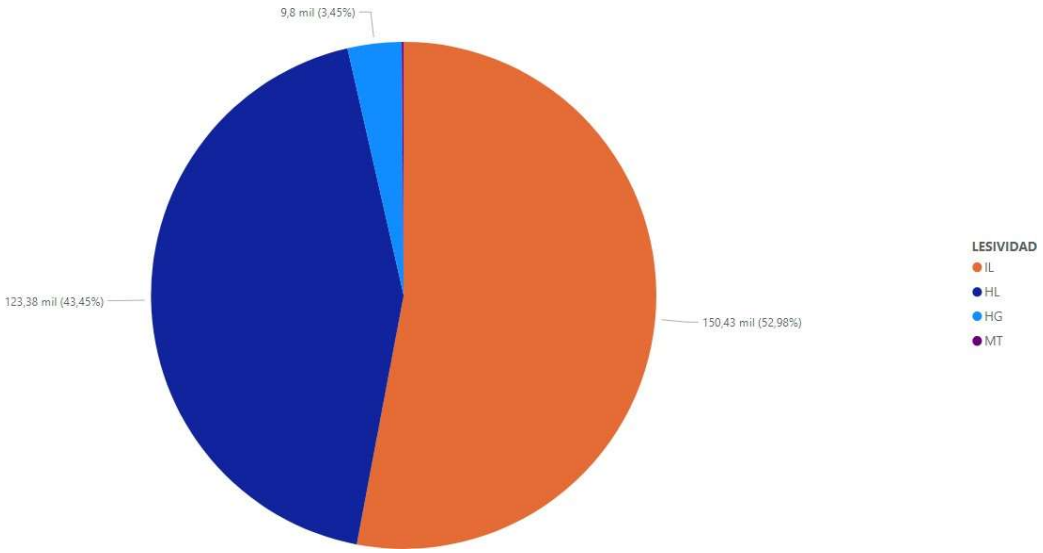
Finalmente, el conjunto de datos obtenido se ha guardado en la base de datos de MongoDB Atlas, dentro de una colección llamada accidentes_estado_meteorologico. El conjunto de datos resultante cuenta con un total de 283947 observaciones y 27 variables diferentes.

6. **Visualización** (Análisis exploratorio)

En este apartado, se llevará a cabo un análisis exploratorio para desgranar la morfología de la base de datos. Este girará en torno a la variable respuesta LESIVIDAD y en él se especificará en comportamiento de cada una de las variables. La herramienta que se usará para analizar las variables categóricas será PowerBI y todos los gráficos mostrados a continuación serán adjuntados de forma complementaria en la entrega del proyecto. Para las variables numéricas se utilizará el R.

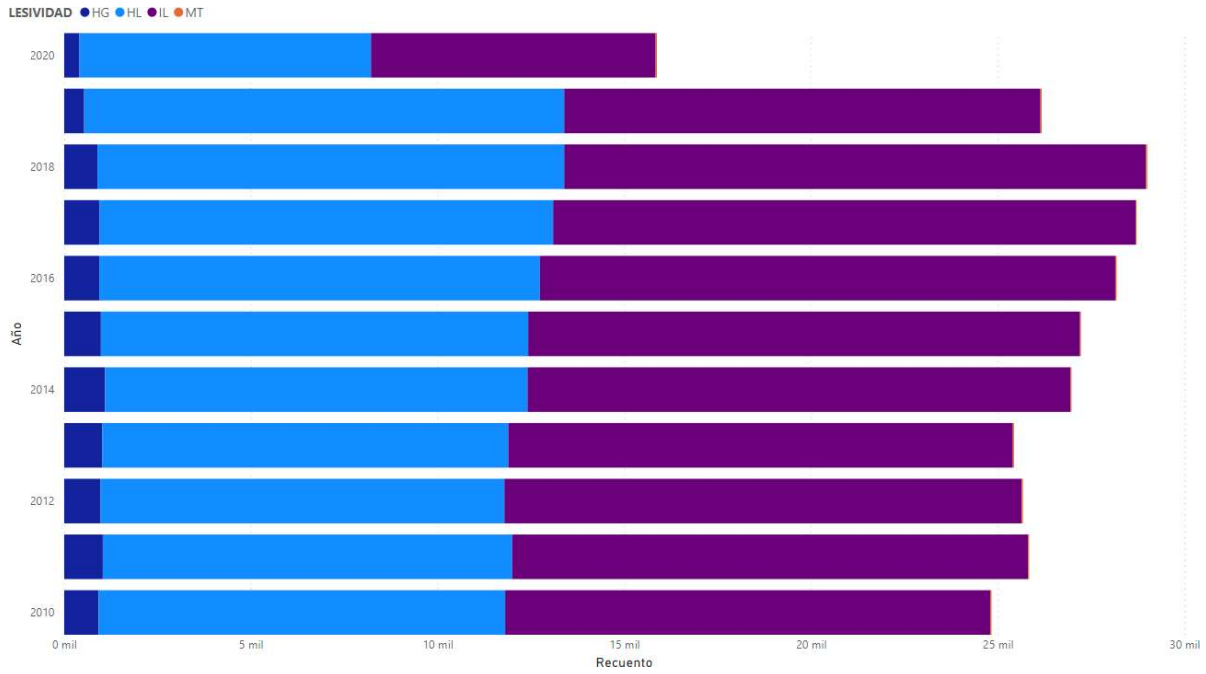
En primer lugar, se tienen un total de 283947 registros, es decir, 283947 personas implicadas en accidentes de los cuales se ha dado parte y ha habido al menos una persona lesionada. Además, se tiene un total de 104313 accidentes registrados con un número de víctimas de 133506. Cabe recordar que se computan como víctimas los heridos leves, heridos graves y fallecidos, en ningún caso los que han salido ilesos.

La variable respuesta LESIVIDAD sigue la siguiente distribución:



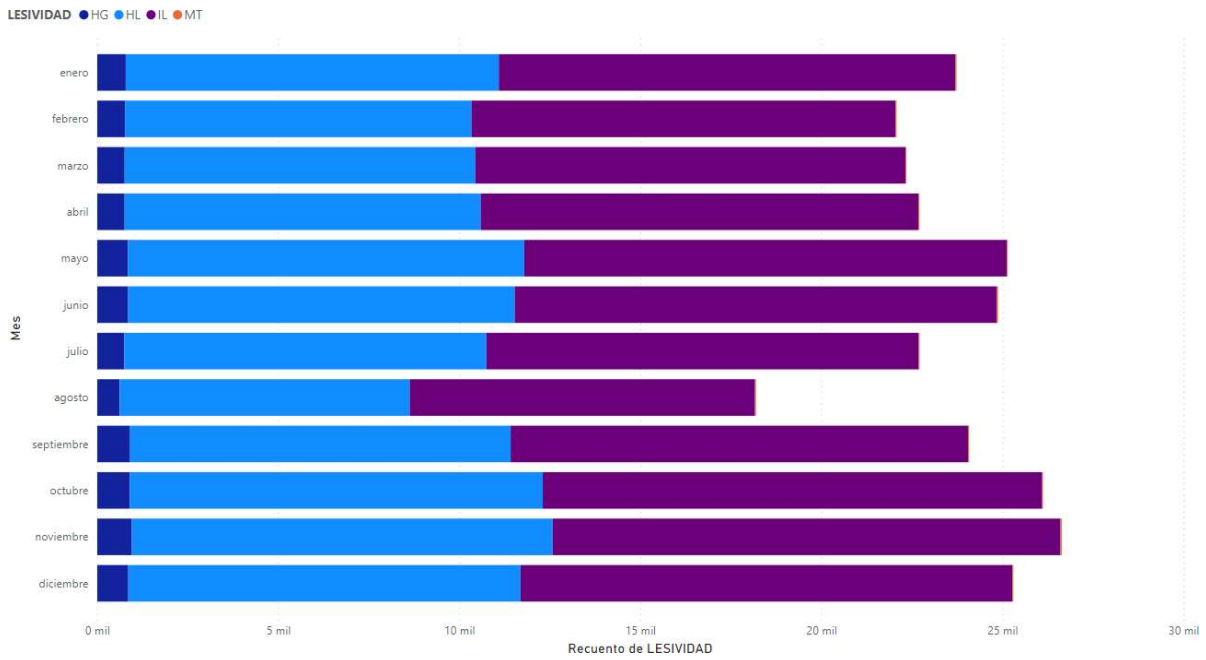
En la gran mayoría de accidentes los implicados salen ilesos (150.43 mil, 52.98%). Por otra parte, los que sufren una lesión suelen padecer lesiones leves (123.38 mil, 43.45%), solo un 3.45% de las personas (9.8 mil) sufren una lesión grave. El porcentaje de muertos resulta ínfimo, de hecho, solo representa el 0.26% de los individuos lesionados.

Será de máxima importancia conocer cómo han evolucionado los datos a lo largo del tiempo. Por lo tanto, se va a evaluar la evolución de los accidentes sufridos del 2010 hasta el 2020.



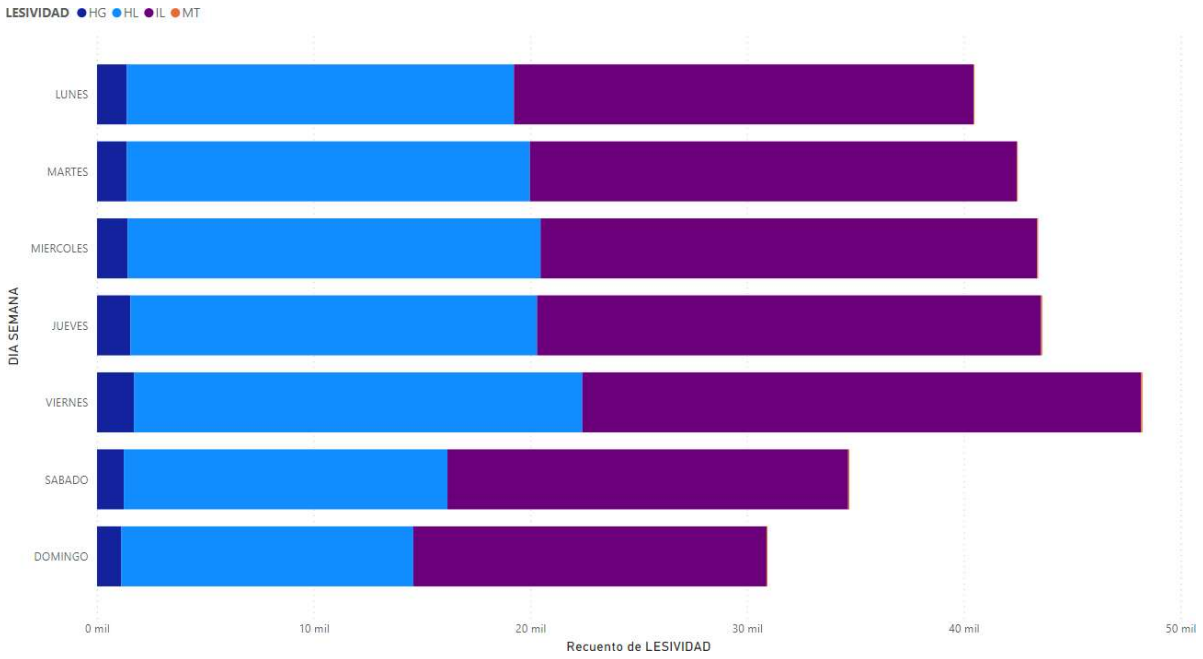
Los accidentes tuvieron una tendencia creciente siendo el año 2018 el año con un mayor volumen (casi 30 mil). En el año 2019 estos se redujeron. Cabe decir que este cambio podría ser debido al cambio de la forma de recopilar la información por parte del ayuntamiento de Madrid a partir del 2019. Además, el impacto del COVID-19, modifica drásticamente los resultados del año 2020, reduciéndolos en gran medida.

Siguiendo con la información respecto al tiempo, resulta interesante conocer la evolución de los accidentes de tráfico a lo largo de los meses del año. Por ello, se ha diseñado este gráfico:



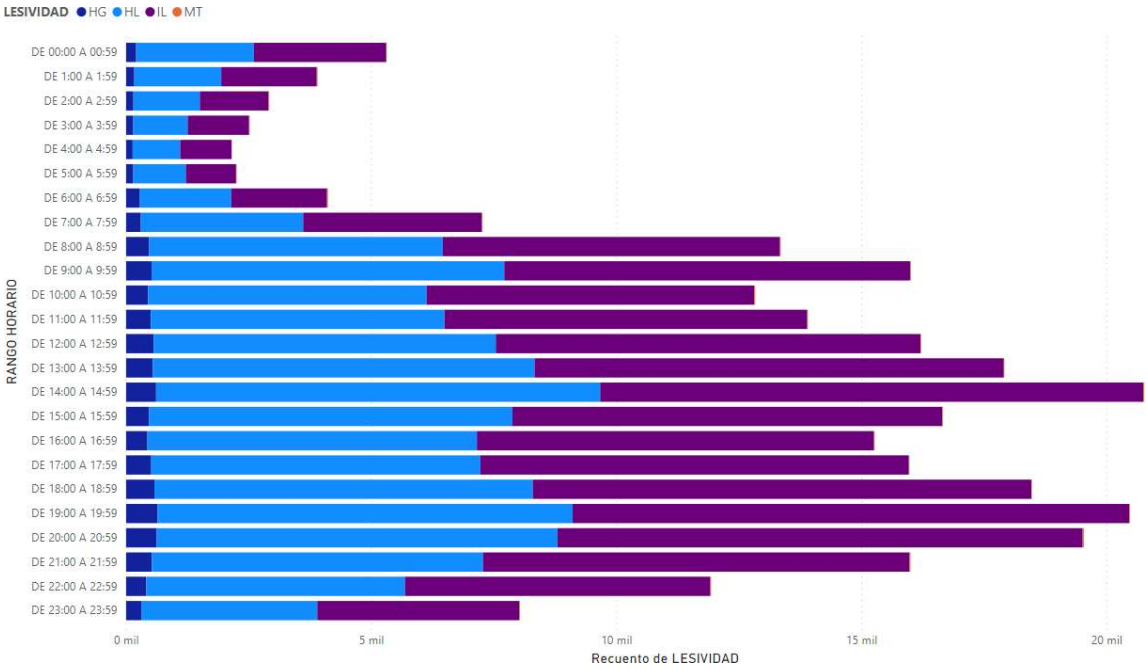
Claramente se puede apreciar como en el mes de agosto se producen un número de accidentes significativamente menor debido a la marcha de gran parte de la ciudadanía de la capital con motivo de sus vacaciones. Los meses que más destacan serían octubre y noviembre por un lado y mayo y junio por el otro, oscilando alrededor de los 25 mil accidentes.

Otro apartado de máximo interés es el que hace referencia a los días de la semana. Aquí un desglose:



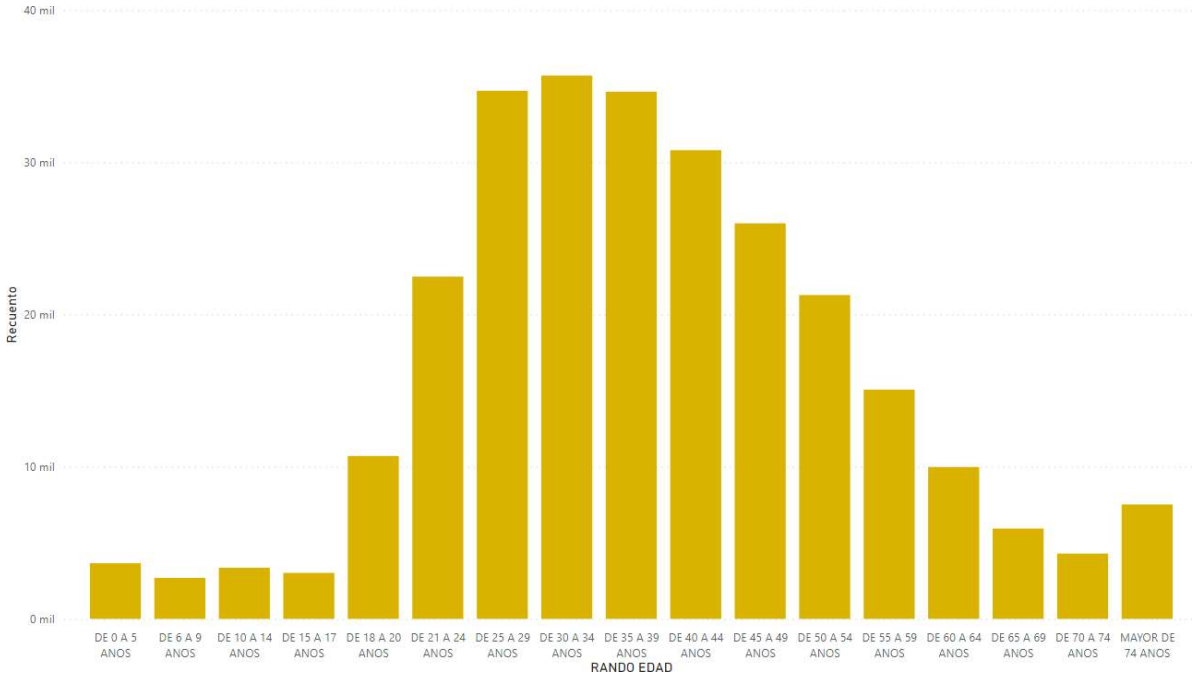
Se aprecia como claramente los días de lunes a viernes se producen un mayor número de accidentes, el uso del transporte para ir a trabajar puede ser una causa vital en este aspecto. Por tanto, los sábados y domingos se producen muchos menos. Los viernes se destacan como el día que se producen un mayor número de siniestros.

Siendo más precisos todavía, si uno se centra en las horas cuando se producen más accidentes, se puede hacer el siguiente desglose:



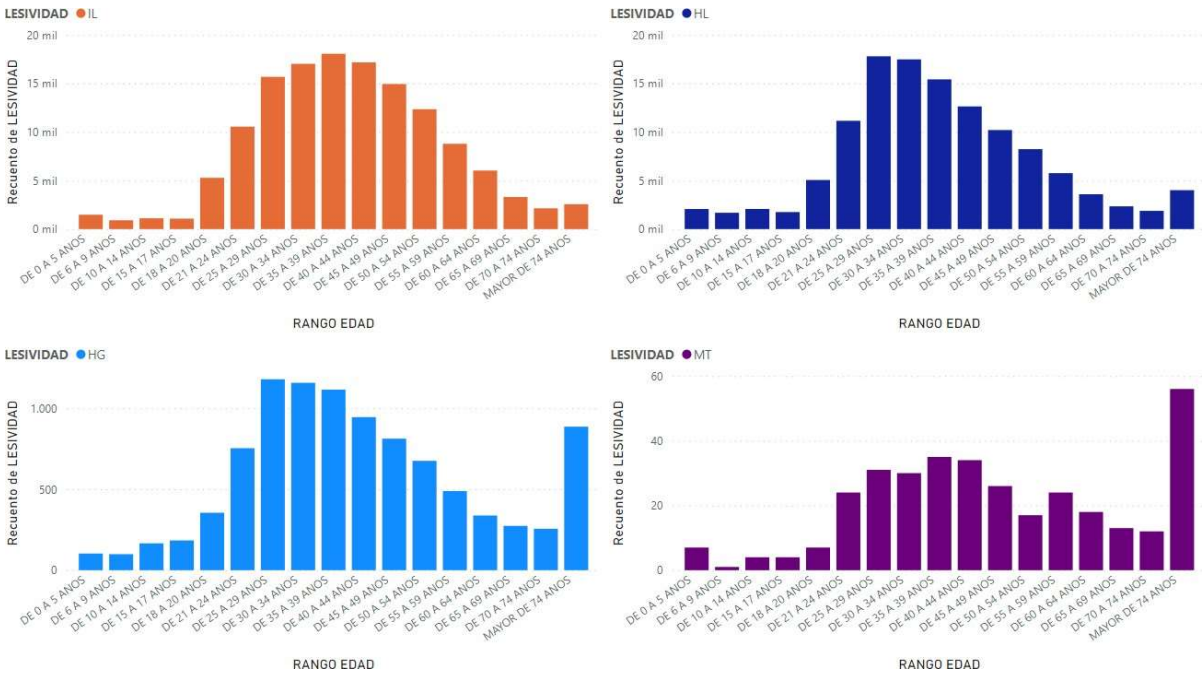
Donde se puede ver que en las primeras horas del día debido a la poca circulación se producen menos accidentes y en las franjas horarias de 18:00 a 21:00 y de 13:00 a 15:00 se producen un número muy elevado.

La edad es un factor clave a tener en cuenta a la hora de sufrir una lesión. Por ello es interesante ver que distribución sigue la variable RANGO EDAD.



Se aprecia que los accidentes mayoritariamente se producen en personas alrededor de los 25-40 años. Estos individuos son los que acostumbran a usar más el coche por lo que resulta lógico. Los accidentes en menores de 18 años no son muy frecuentes ya que estos no pueden hacer uso del vehículo como conductores.

Por otra parte, es interesante estudiar la variable según la lesividad de los accidentes.



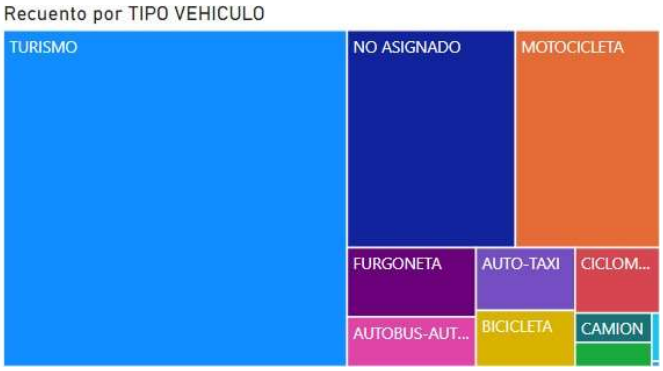
En este caso, hay una distribución similar de los datos, pero cabe destacar que los mayores de 74 años tienden a tener una mayor facilidad para sufrir una lesión grave omorir de una forma muy significativa debido a que su estado de salud es más débil.

También resulta relevante estudiar qué aspectos son los más recurrentes y provocan una mayor lesividad en los accidentes según algunos parámetros como el tipo de accidente, de persona o de vehículo. A continuación, se comentarán algunas tablas queexplican esto:

TIPO ACCIDENTE	HG	HL	IL	MT	Total
COLISION DOBLE	4086	67520	84991	79	**156676**
ATROPELLO	3295	14650	27145	171	**45261**
COLISION MULTIPLE	419	12381	18607	15	**31422**
CAIDA	1114	15729	9017	32	**25892**
OTRO	132	5504	6254	5	**11895**
CHOQUE OBSTACULO FIJO	676	6784	4064	40	**11564**
VUELCO	76	806	339	2	**1223**
DESCONOCIDO	2	4	8		**14**
Total	**9800**	**123378**	**150425**	**344**	**283947**

El tipo de accidente que sucede mayoritariamente es una colisión doble (156676). Otros accidentes habituales serían el atropello (45261), la colisión múltiple (31422) o la caída (25892), en este orden. Centrando la atención en las muertes, la mayoría se producen por atropello (171).

TIPO VEHICULO	HG	HL	IL	MT	Total
TURISMO	1657	54301	92764	56	**148778**
NO ASIGNADO	2966	11832	32125	143	**47066**
MOTOCICLETA	3583	33790	2835	117	**40325**
FURGONETA	139	3146	8182	3	**11470**
AUTOBUS-AUTOCAR	136	3755	4343	5	**8239**
AUTO-TAXI	102	2822	5080		**8004**
BICICLETA	568	5915	656	11	**7150**
CICLOMOTOR	527	6055	519	3	**7104**
CAMION	44	482	2362	3	**2891**
VARIOS	62	1041	1277	2	**2382**
AMBULANCIA	10	200	275		**485**
VEH.3 RUEDAS	6	39	7	1	**53**
Total	**9800**	**123378**	**150425**	**344**	**283947**



En cuanto al tipo de vehículo, la mayoría de los accidentes se producen al volante de un turismo (148778). Esto resulta obvio ya que es el vehículo más usado en una ciudad en la actualidad. Por otra parte, las motocicletas también tienen una gran representación en los accidentes en la ciudad (47066) y resulta sorprendente la cuantiosa cantidad de vehículos desconocidos con los que se producen accidentes (47066). Esto supone una pérdida de información terrible, ya que conocer el tipo de vehículo partícipe en el accidente podría ser muy relevante. Respecto a los fallecimientos, hay un número muy amplio producido por conducir una motocicleta (117) demostrando así su peligrosidad.

TIPO PERSONA	HG	HL	IL	MT	Total
CONDUCTOR	5500	80542	92860	153	**179055**
VIAJERO	1018	28923	25423	18	**55382**
TESTIGO			31666		**31666**
PEATON	3282	13908	462	173	**17825**
DESCONOCIDO		5	14		**19**
Total	**9800**	**123378**	**150425**	**344**	**283947**

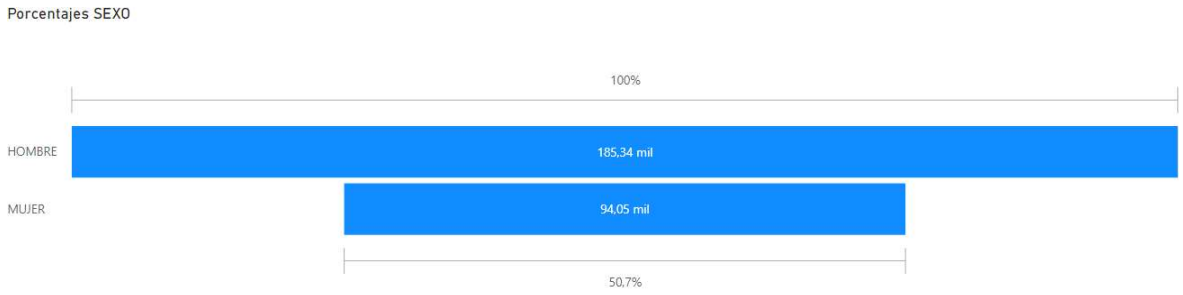
El tipo de persona más repetido es, como era de esperar, el conductor (179055), ya que este necesariamente ha de participar en el accidente y, por lo tanto, todos ellos deben al menos tener uno. El siguiente tipo de persona más repetido es el viajero (55382). Hay que poner el asterisco en la cifra de testigos (31666) ya que desde el año 2019, como se ha comentado previamente, no se recoge información al respecto. Es evidente que todos han salido ilesos del siniestro. Por último, los peatones son la categoría minoritaria (17825), pero por contra, son las que tienen un índice de mortalidad mayor (173 muertos).

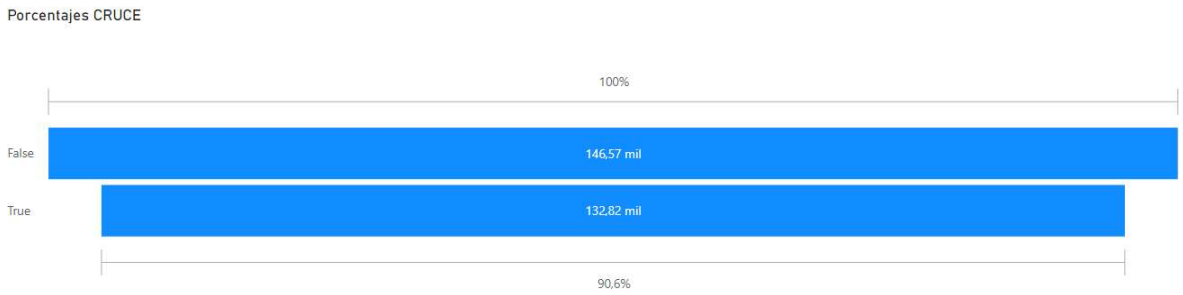
Para terminar con este análisis de tablas, se mostrará una tabla conjunta del tipo de accidente, de vehículo y de persona en la que se visualizará el recuento de fallecidos para estudiar qué tipo de accidentes son más mortíferos:

TIPO VEHICULO	TIPO ACCIDENTE	TIPO PERSONA	Recuento de LESIVIDAD
NO ASIGNADO	ATROPELLO	PEATON	142
MOTOCICLETA	COLISION DOBLE	CONDUCTOR	48
MOTOCICLETA	CAIDA	CONDUCTOR	28
MOTOCICLETA	CHOQUE OBSTACULO FIJO	CONDUCTOR	19
TURISMO	ATROPELLO	PEATON	17
TURISMO	CHOQUE OBSTACULO FIJO	CONDUCTOR	12
BICICLETA	COLISION DOBLE	CONDUCTOR	10
MOTOCICLETA	COLISION MULTIPLE	CONDUCTOR	9
TURISMO	COLISION DOBLE	CONDUCTOR	8
TURISMO	COLISION DOBLE	VIAJERO	8
MOTOCICLETA	ATROPELLO	PEATON	5
AUTOBUS-AUTOCAR	ATROPELLO	PEATON	4
MOTOCICLETA	OTRO	CONDUCTOR	4
TURISMO	CHOQUE OBSTACULO FIJO	VIAJERO	4
CICLOMOTOR	CAIDA	CONDUCTOR	2
FURGONETA	CHOQUE OBSTACULO FIJO	CONDUCTOR	2
MOTOCICLETA	COLISION DOBLE	VIAJERO	2
TURISMO	COLISION MULTIPLE	CONDUCTOR	2
TURISMO	VUELCO	CONDUCTOR	2
AUTOBUS-AUTOCAR	CAIDA	VIAJERO	1
BICICLETA	CHOQUE OBSTACULO FIJO	CONDUCTOR	1
CAMION	ATROPELLO	PEATON	1
CAMION	COLISION DOBLE	CONDUCTOR	1
CAMION	COLISION MULTIPLE	CONDUCTOR	1
CICLOMOTOR	COLISION DOBLE	CONDUCTOR	1
FURGONETA	ATROPELLO	PEATON	1
MOTOCICLETA	CAIDA	VIAJERO	1
MOTOCICLETA	COLISION MULTIPLE	VIAJERO	1
NO ASIGNADO	CHOQUE OBSTACULO FIJO	PEATON	1
TURISMO	COLISION DOBLE	PEATON	1
TURISMO	COLISION MULTIPLE	VIAJERO	1
TURISMO	OTRO	CONDUCTOR	1
VARIOS	ATROPELLO	PEATON	1
Total			**344**

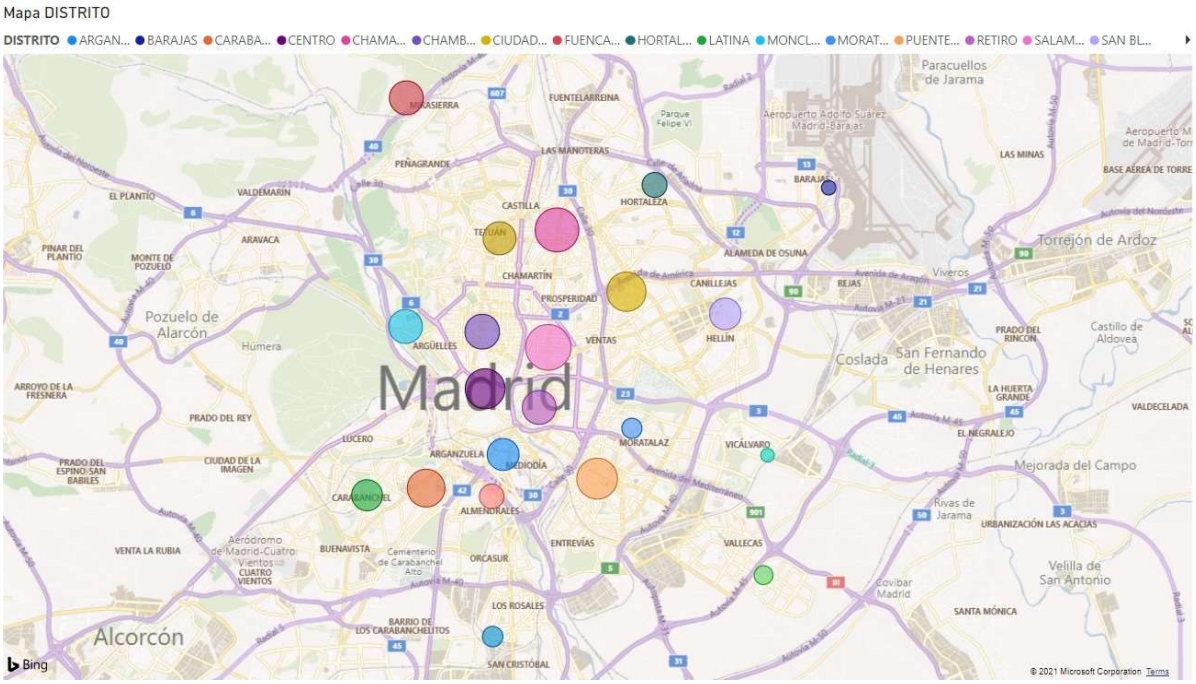
El accidente mortal más repetido es el atropello a un peatón (142). Estos tienen un índice de mortalidad elevadísimo. Otros accidentes con mucha peligrosidad de muerte son en los que interviene una motocicleta y muere el conductor de esta, más específicamente las colisiones dobles (48), las caídas (28) y los choques con un obstáculo fijo (19).

Resulta productivo estudiar otras variables como el hecho de que el accidente se haya producido en un cruce o el género del conductor. En estas dos gráficas se puede apreciar con cierta facilidad que el número de hombres implicados dobla al de mujeres y que aproximadamente el mismo número de accidentes se han producido en cruces que en carreteras o calles (solo un 9.4% más).





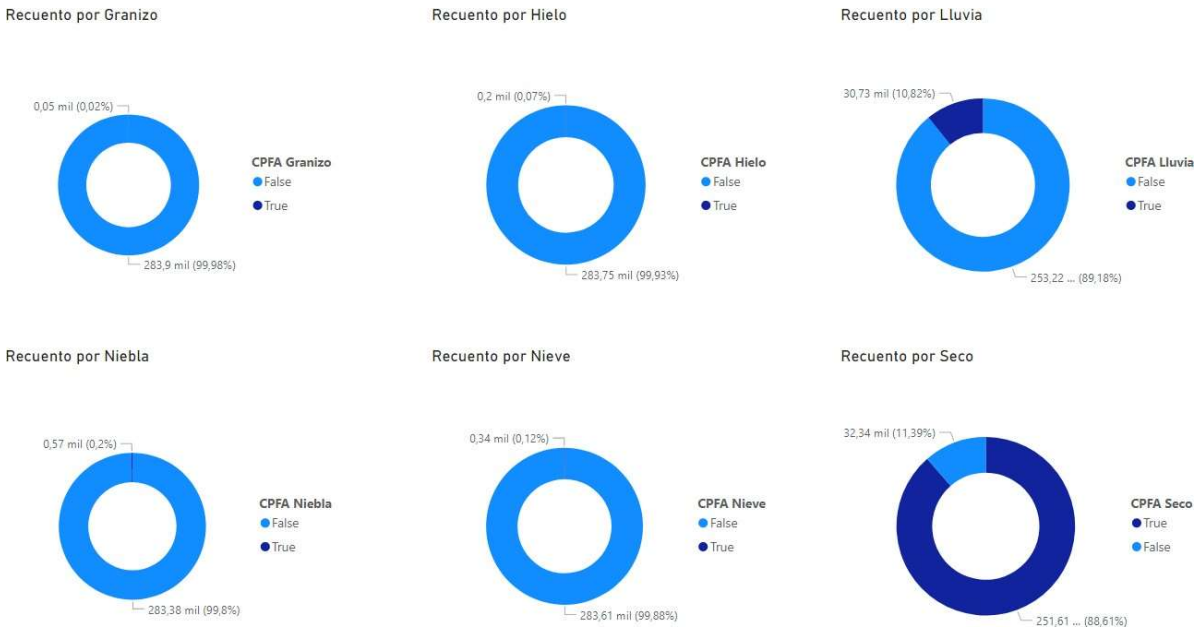
Para contemplar la distribución geográfica se ha diseñado un mapa y se han ubicado los distintos distritos con círculos de mayor o menor tamaño dependiendo de la cantidad de accidentes producidos en cada uno de ellos. El resultado es el siguiente:



Para elaborar unas conclusiones satisfactorias respecto a la ubicación de los accidentes también se han elaborado estos gráficos de barras con el recuento de accidentes totales y accidentes mortales distribuidos en distritos:

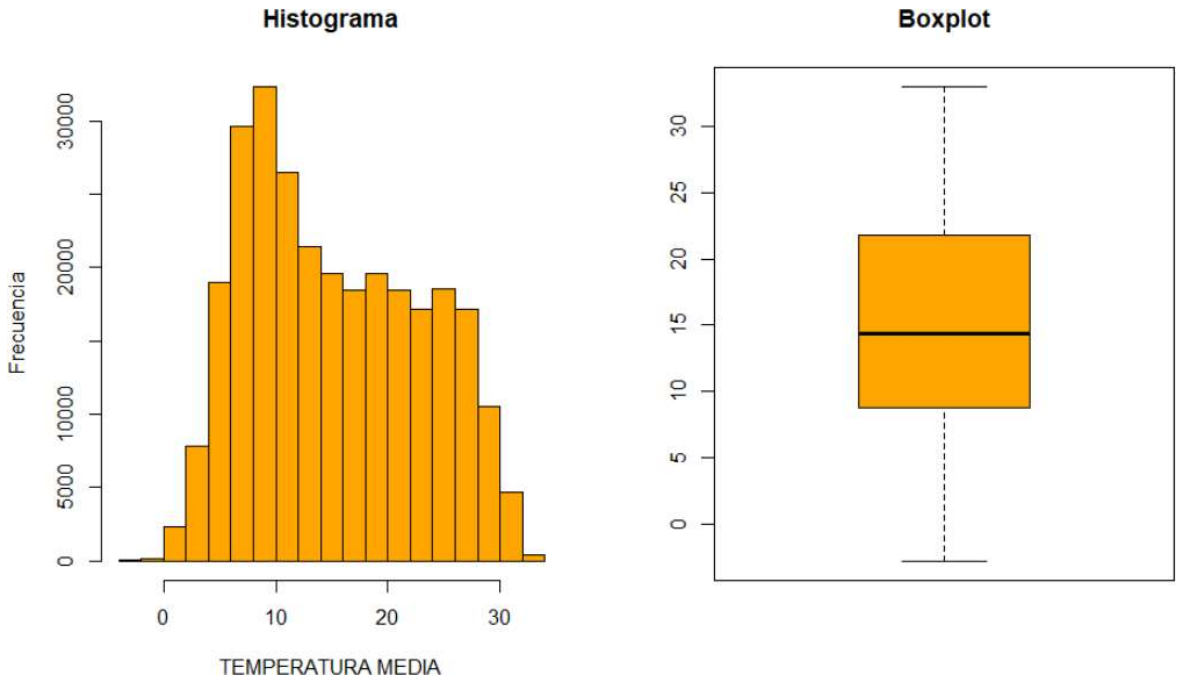
Se puede apreciar que la mayoría de los accidentes se producen en los lugares más céntricos de la ciudad como, por ejemplo, Salamanca o Madrid Centro. En distritos de la periferia como Barajas, Villaverde, Hortaleza, Villa de Vallecas o Vicálvaro se acumula un menor número de accidentes. Adicionalmente, cabe destacar la alta mortalidad en la zona de Fuencarral- El Pardo posiblemente causada por el paso de la autovía M-40.

En el aspecto meteorológico se obtienen los siguientes resultados:

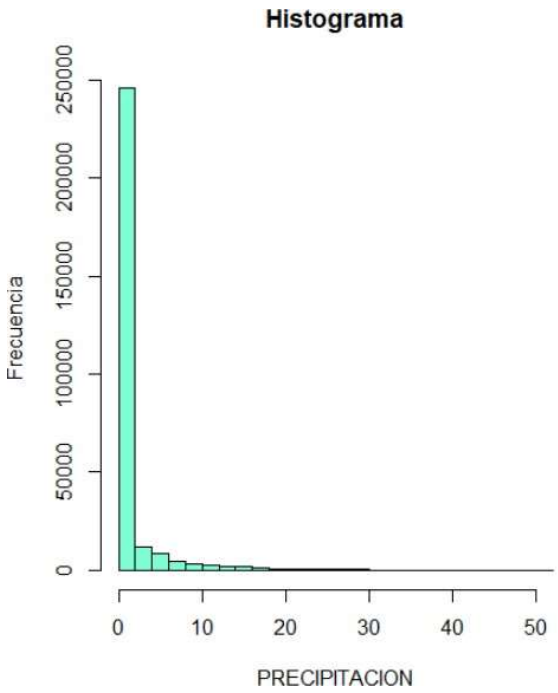


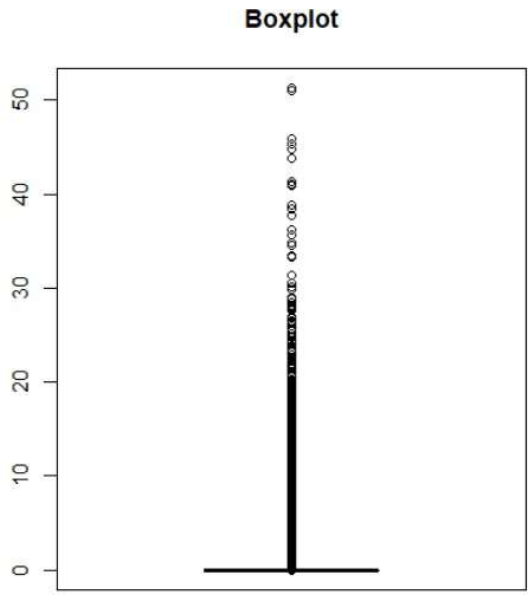
Algunos aspectos como el granizo, el hielo, la niebla o la nieve son realmente inusuales, ya que no se producen ni en un 1% de los días desde 2010 hasta 2020. Por ello, no parece que tengan excesiva relevancia sobre nuestras observaciones. Por otra parte, se llega a la conclusión que alrededor de un 10% de los días ha llovido.

En última instancia, se hará un análisis detallado de las variables numéricas proporcionadas por AEMET y se evaluará si existe algún tipo de correlación entre ellas. En primer lugar, se mostrará un histograma y un boxplot para cada una de ellas.

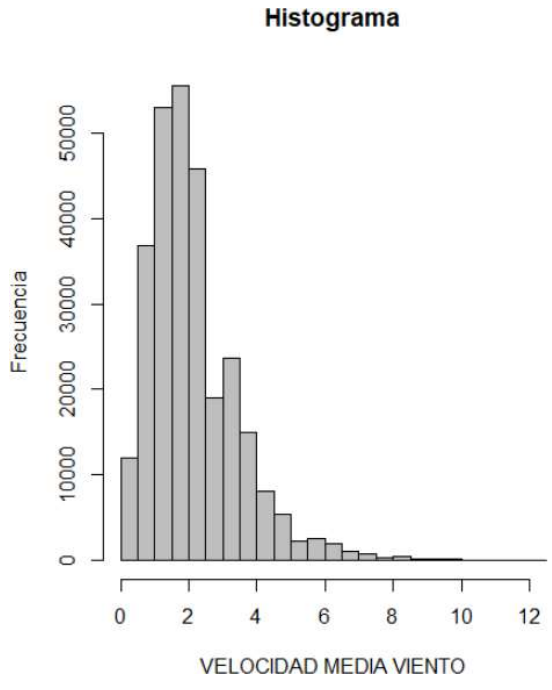


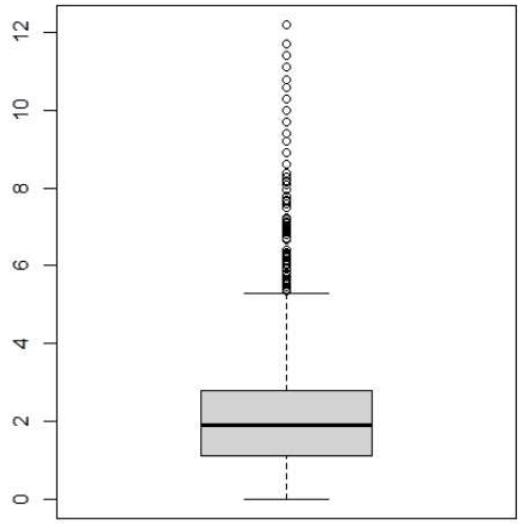
La temperatura media se encuentra en grados centígrados y tiene una media de 15.39. Se puede apreciar como en Madrid las temperaturas a lo largo del año suelen oscilar entre los 9 y 21 grados. Aun así, en verano se ha llegado a tener temperaturas máximas de 33 grados y en invierno se han llegado a registrar -2.8 grados.





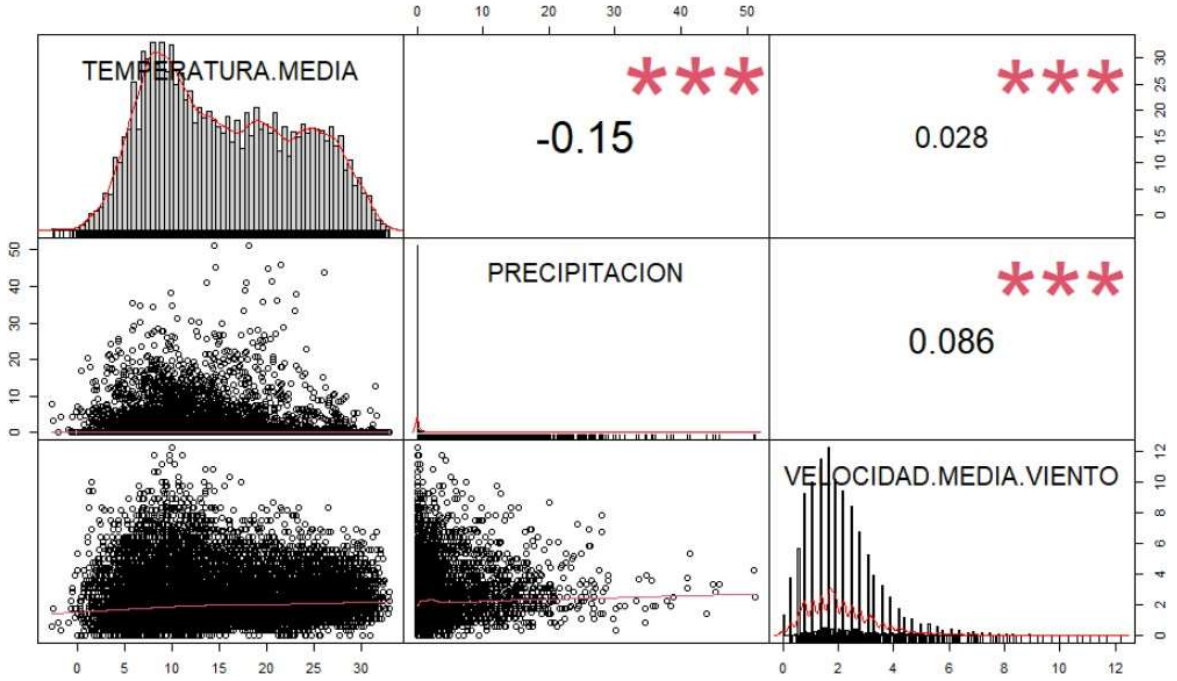
En cuanto a la precipitación, se puede observar que Madrid no es una ciudad especialmente lluviosa. La mayoría de los registros indican que no ha llovido durante todo el día ya que los primer y tercer cuartiles tienen valor 0 mm. Por lo tanto, se podría considerar casi como una excepción los días de lluvia durante el año ya que son “outliers”. La máxima precipitación registrada es de 51.3 mm.





Como se puede apreciar en los gráficos, la media de la velocidad del viento en Madrid es de 2.139 m/s. A lo largo del año se obtienen resultados entre los 0 y 4 m/s como se ve en el histograma. Aun así, se han llegado a reportar registros de hasta más de 6 m/s,alcanzando el pico de 12.3 m/s.

Por último, el siguiente esquema muestra la correlación entre estas variables. En él, aparecen representados los histogramas de cada variable en la diagonal, los gráficos de dispersión en la parte inferior y los coeficientes de covariancia de Pearson en la partesuperior.



Se puede apreciar que la correlación es bajísima, siendo -0.15 el coeficiente de correlación de Pearson mayor en valor absoluto, por lo que cada una de ellas aportaráinformación distinta al conjunto de los datos.

7. Modelos Analíticos

Para la predicción de la variable objetivo, ***Lesividad***, se utilizarán dos modelos analíticos concretos: ***Random Forest, XGBoost.*** Por otro lado, se ha intentado predecir la gravedad de los accidentes reduciendo la variable Lesividad en Accidentes Graves (AG) y Accidentes Leves (AL), utilizando para ello el módelo

Como antes hemos podido observar en la muestra analizada, las variables que la comprenden son, en su mayoría, categóricas. La información almacenada en este tipo de muestras es distinta a una muestra con variables cuantitativas y por tanto su estudio y análisis también. Son por tanto ambos los modelos antes citados, los seleccionados y que finalmente se utilizarán, teniendo en cuenta la longitud de la muestra y el tipo de información contenida en la misma.

7.1 XGBoost

El primero modelo seleccionado es el basado en el algoritmo conocido como Extreme Gradient Boosting o XGBoost [28]. Un algoritmo supervisado de machine learning muy utilizado últimamente en modelos de aprendizaje automático.

Una de las claves de este algoritmo es la eficiencia computacional, es decir, podemos obtener resultados realmente contundentes con un esfuerzo computacional menor que el de otros modelos más complejos. Este modelo está compuesto por multitud de modelos secuenciales que realizan una predicción no muy compleja, avanzando de modelo a modelo, tomando el resultado del anterior y adquiriendo en cada secuencia un grado superior de robustez y por tanto mayor precisión.

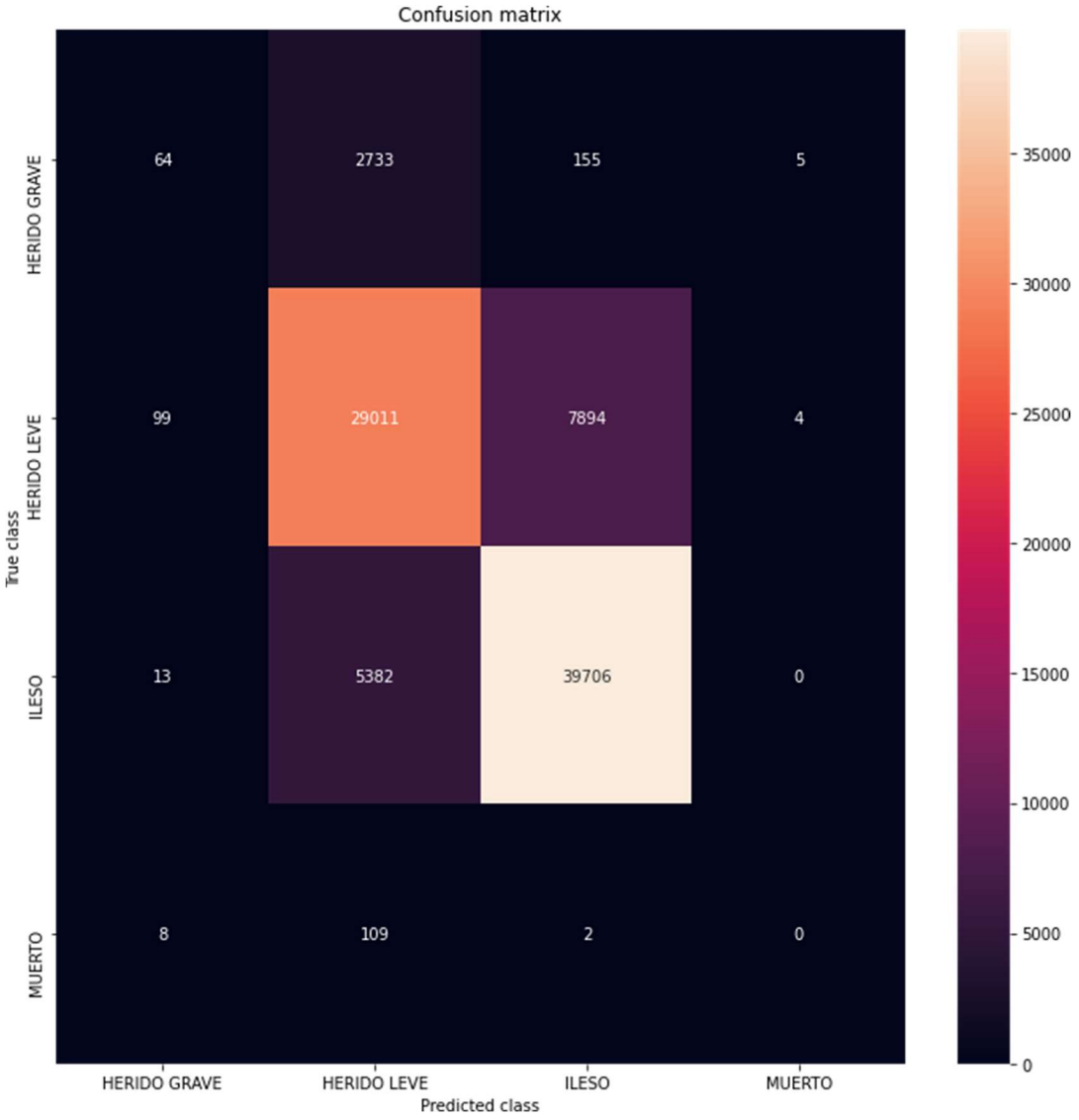
Para la formación de estos modelos simples o secuenciales que conforman el algoritmo XGBoost, se utilizan arboles de decisión, de manera progresiva y por tanto cada vez más profundos.

El algoritmo será construido en lenguaje Python apoyados principalmente en las librerías: sklearn y xgboost

Para explicar la construcción del modelo XGBoost, dividiremos por etapas las fases de elaboración y destacaremos los resultados obtenidos:

- Selección de variables e iteración.
- Para esta librería en concreto es necesario realizar el Encoding de variables categóricas, un proceso obligatorio para que el modelo pueda computarse adecuadamente. Para este proceso utilizaremos la codificación One Hot Encoding, más concretamente el proceso get dummies o dummificacion de variables de la librería Pandas, creando una nueva columna por categoría para cada variable.
- El tratamiento de las variables booleanas será su conversión a números enteros, dónde el valor booleano True será 1, mientras que el valor False será 0.
- Para nuestra variable objetivo utilizaremos un proceso de codificación distinto, LabelEncoding, dándole a cada categoría de esta variable un numero único y ordenado.
- Una vez codificada el total de la muestra, separamos el Dataframe en dos partes, la primera, denominada 'X', con las variables para el entrenamiento y la segunda, 'Y', únicamente compuesta por la variable objetivo (Lesividad).
- Se produce la separación de ambos subsets 'X' e 'Y' en train y test para ambas. En una proporción del 70% para train y 30% para test, todo esto con la aplicación de una semilla de randomización.
- Entrenamiento del modelo, el cual consta de los siguientes parámetros:
    - *Número de estimadores:* 80
    - *Ratio de aprendizaje:* 0.5
    - *Porcentaje de columnas que se utilizan por árbol:* 0.5
    - *Profundidad máxima del árbol:* 8
    - *Peso de la hoja L1:* 10
- Test del modelo

7.1.1 Matriz de confusión



	Precision	*Recall*	*f1- score*	*Support*
0	0.35	0.02	0.04	2957
1	0.78	0.78	0.78	37008
2	0.83	0.88	0.86	45101
3	0.00	0.00	0.00	119
Accuracy			0.81	85185
Macro avg	0.49	0.42	0.42	85185
Weighted avg	0.79	0.81	0.79	85185

- ***Accuracy***: Número total de predicciones correctas dividido por el número totalde predicciones
- ***Precision***: Define cuan confiable es un modelo en responder si un punto pertenece a esa clase.
- ***Recall***: Expresa cuan bien puede el modelo detectar a esa clase.
- ***f1- score***: Media harmónica de precisión y recall (2 x precision x recall / (precision+recall)) digamos que combina precisión y recall en una sola métrica.

7.1.2 Precisión del modelo

Para este conjunto de datos el algoritmo predice la variable ***'Lesividad'*** con una precisión del ***80,74%.***

```
[ ]  print('XGBoost model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

     XGBoost model accuracy score: 0.8074
```

Además, se ha realizado un calculo del error utilizando la métrica ***Mean Absolute Error (MAE)*** [29], el resultado ha sido del ***0.196***, el cual resulta ser un valor bastante bueno, ya que se encuentra del 0 absoluto.

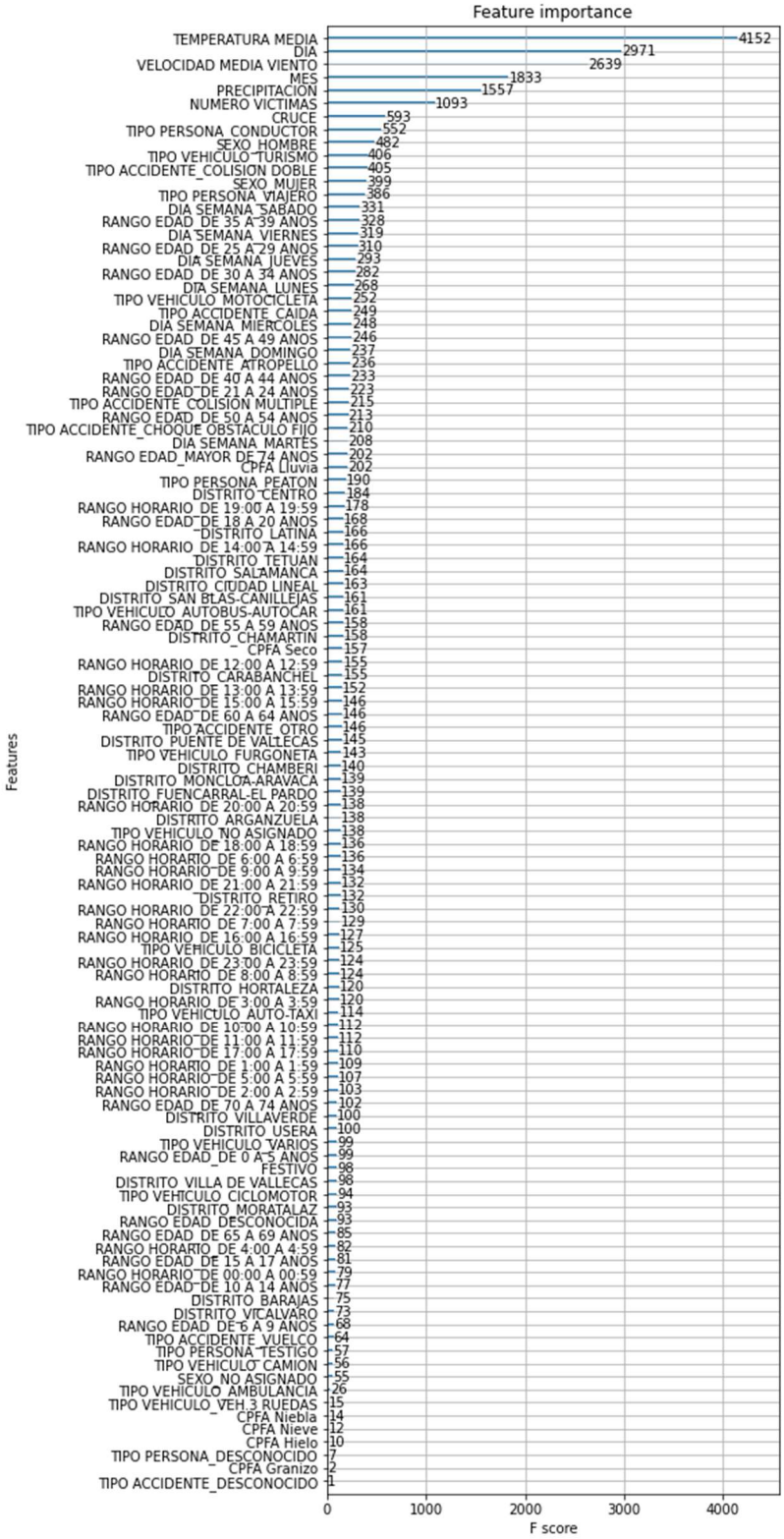
```
[ ]  score = mean_absolute_error(y_test, y_pred)
     print('MAE:', score)

     MAE: 0.19617303515877207
```

Las categorías de las que se disponían menos datos: ***'MUERTO'*** y ***'HERIDO GRAVE'***, son dos valores cualitativos difíciles de aprender para el algoritmo por la falta de información y no aportan un resultado muy preciso. No son malos resultados y este tipo de algoritmos trabaja bien con matrices sparse, con valores 0, cercanos al cero o 'sucesos extraños' ('MUERTO' y'HERIDO GRAVE'). Con más información resultará un modelo sólido.

7.1.3 Importancia de variables

Por último, se ha llevado a cabo un último análisis de cuales resultan ser las variables más importantes utilizadas por el modelo XGBoost implementado. El resultado ha sido presentado a través de la siguiente tabla:



En ella, aparece como filas todas y cada una de las variables utilizadas para la creación del modelo predictivo, y a modo de barra se muestra a través de un valor la importancia de la variable en la decisión del resultado. Analizando esta gráfica podemos deducir que hay un conjunto de variables que realmente no son relevantes para la predicción, estas son:

- CPFA Granizo
- CPFA Hielo
- CPFA Nieve
- CPFA Niebla
- TIPO ACCIDENTE (DESCONOCIDO)
- TIPO PERSONA (DESCONOCIDO)

7.2 Random Forest

Se seleccionó un modelo predictivo con Random Forest [30] debido a la facilidad que presenta este algoritmo a la hora de elaborar arboles de decisión, lo que facilitaencontrar puntos de corte en las variables independientes. Al establecer estos puntos de corte podemos aproximar aquellas variables que tengan mayor correlación con la variable dependiente. Por lo tanto, el algoritmo tendrá más fácil promediar varios árboles de decisión entrenados con distintas características. Para poder testear el funcionamiento del modelo debemos de dividir la base de datos entorno a un porcentaje (realizamos el proceso mediante el método split), en este caso lo hemos hecho entorno al 80% para train (datos que utilizaremos para entrenar el modelo) y un 20% para test (que utilizaremos para realizar test sobre el modelo y la información que puede aportar). Tanto en la parte del Split como en la aplicación del algoritmo tenemos que ponerle una seed para hacer que los procedimientos anteriores tengan un cierto azar, pero que pueda ser reproducible y obtenga los mismos valores en los que se ha basado el modelo para establecer las predicciones.

Matriz de confusión: nos da la información necesaria sobre los resultados del modelo y sus futuras predicciones que nosotros utilizaremos para establecer si hemos elaborado un buen o mal modelo de predicción. A continuación, hablaremos de ello:

- ***Accuracy:*** podemos averiguar cuál es la precisión del modelo elaborado, en este casohemos conseguido una predicción del 78%.

- ***Recall o sensibilidad:*** lo utilizamos para averiguar la sensibilidad del modelo para identificar elementos correctamente como positivos del total de positivos, en este caso hemos conseguido:
    - *Heridas graves (HG):* 48% de sensibilidad
    - *Heridas leves (HL):* 77% de sensibilidad
    - *Ingresos leves (IL):* 78% de sensibilidad
    - *Fallecidos (MT):* 1% de sensibilidad
- ***Especificidad:*** lo utilizamos para averiguar la sensibilidad del modelo para identificar elementos correctamente como negativos del total de negativos, en este caso hemos conseguido:
    - *Heridas graves (HG):* 96% de especificidad
    - *Heridas leves (HL):* 79% de especificidad
    - *Ingresos leves (IL):* 85% de especificidad
    - *Fallecidos (MT):* 4% de especificidad

7.3 Regresión Logística

Por último, se ha llevado a cabo la creación de un modelo predictivo utilizando el algoritmo Regresión Logística [31], este tipo de modelos es utilizado para predecir el resultado de una variable categórica en función de un conjunto de variables independientes. Para realizar este modelo, también hemos realizado una limpieza de datos y hemos quitado las FECHA y CPFA Desconocido debido a su irrelevancia. Además, en este caso hemos optado por otro tipo de solución en el cual, el resultado será binomial , por lo que la variable LESIVDAD, se ha reducido en Accidentes Graves (AG) con muertos y heridos graves y Accidentes Leves (AG) con heridos leves e ilesos, de esta forma intentamos predecir la probabilidad de que una persona involucrada en un accidente en Madrid sea de alta gravedad o leve.

A diferencia del modelo anterior, se ha procedido a dividir la base de datos en un 70% para el conjunto de entrenamiento y en un 30% para las pruebas. En este caso, los resultados que hemos obtenido en la matriz de confusión han sido los siguientes:

	FALSE	TRUE
AG	23	3020
AL	83	82058

El resultado que hemos obtenido con este modelo en el conjunto de pruebas ha sido el siguiente:

- ***Accuracy:*** podemos averiguar cuál es la precisión del modelo elaborado, en este casohemos conseguido una predicción del 96%.
- ***Recall o sensibilidad:*** lo utilizamos para averiguar la sensibilidad del modelo para identificar elementos correctamente como positivos del total de positivos, en este caso hemos conseguido un 99% de sensibilidad.
- ***Especificidad:*** lo utilizamos para averiguar la sensibilidad del modelo para identificar elementos correctamente como negativos del total de negativos, en este caso hemos conseguido un 7% de especificidad.

8. Glosario

Big Data: Macrodatos o datos masivos. Es un concepto que hace referencia al almacenamiento de grandes cantidades de datos y a los procedimientos usados para encontrar patrones repetitivos dentro de esos datos.

Análisis de Datos: Proceso que consiste en inspeccionar, limpiar y transformar datos con el objetivo de resaltar la información útil para sugerir conclusiones y apoyo en la toma de decisiones.

Modelo de Análisis de Datos: Modelo mediante el cual se van a realizar todas las etapas del análisis de datos. Que se divide en 6 etapas: el problema, la recolección, la limpieza, la exploración, el análisis y la conclusión.

Mongo DB: Base de datos no relacional y de código abierto orientada a documentos.

Python: Lenguaje de programación interpretado que soporta la programación orientada a objetos y que hace hincapié en la legibilidad de su código.

R: Lenguaje de programación multiparadigma orientado al análisis estadístico.

Start Up: Empresa de nueva creación sobre una base tecnológica innovadora que tiene una elevada tasa de crecimiento.

Agile: Es una filosofía que supone una forma de trabajar y organizarse, en la que cada proyecto se trocea en pequeñas partes que tienen que completarse y entregarse en pocas semanas.

Dataframe: Estructura de datos organizada para realizar el estudio estadístico de una muestra.

9. Referencias

[1] PyCharm (https://www.jetbrains.com/pycharm/)

[2] Spyder (https://www.spyder-ide.org/)

[3] Jupyter Notebook (https://jupyter.org/)

[4] Google Colaboratory (https://colab.research.google.com/notebooks/intro.ipynb#recent=true)

[5] R Studio (https://www.rstudio.com/)

[6] MongoDB (https://www.mongodb.com/es)

[7] Pandas (https://pandas.pydata.org/)

[8] Dask (https://dask.org/)

[9] Scikit-Learn (https://scikit-learn.org/stable/index.html)

[10] Selenium (https://www.selenium.dev/)

[11] PyMongo (https://pymongo.readthedocs.io/en/stable/index.html)

[12] PowerBI (https://powerbi.microsoft.com/es-es/)

[13] Datos Madrid (https://datos.madrid.es/portal/site/egob)

[14] Datos Abiertos Madrid ¿Qué son? (https://datos.madrid.es/portal/site/egob/menuitem.400a817358ce98c34e937436a8a409a0/?vgnextoid=eba412b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextchannel=eba412b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default)

[15] Web Scraping (https://es.wikipedia.org/wiki/Web_scraping)

[16] MongoDB Atlas (https://www.mongodb.com/cloud/atlas/register)

[17] AEMET (http://www.aemet.es/es/portada)

[18] API (https://en.wikipedia.org/wiki/API)

[19] requests (https://docs.python-requests.org/es/latest/)

[20] mongolite (https://jeroen.github.io/mongolite/)

[21] PyMongo, función insertMany (https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/)

[22] PyMongo, función find (https://docs.mongodb.com/manual/reference/method/db.collection.find/)

[23] Dask setup (https://docs.dask.org/en/latest/setup.html)

[24] Dask virtudes (https://docs.dask.org/en/latest/)

[25] Lazy execution (https://tutorial.dask.org/01x_lazy.html)

[26] Dask vs Apache Spark (https://docs.dask.org/en/latest/spark.html?highlight=spark)

[27] Dask, conjunto de funciones para dataframes (https://docs.dask.org/en/latest/dataframe-api.html)

[28] XGBoost (https://xgboost.readthedocs.io/en/latest/)

[29] Mean Absolute Error (https://es.wikipedia.org/wiki/Error_absoluto_medio)

[30] Random Forest (https://es.wikipedia.org/wiki/Random_forest)

[31] Regresión Logística (https://es.wikipedia.org/wiki/Regresión_logística)