UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Trabajo Fin de Grado en Ingeniería Informática

Análisis y creación de modelos del tráfico en la ciudad de Madrid (2019 - 2021) con R y Python

Analysis and modeling of traffic in the city of Madrid (2019 - 2021) with R and Python

Dirigido por:

Sonia Estévez Martín

Ana Álava Papí

Martín Rovira Barrionuevo

Curso académico 2021-22

Convocatoria junio

Resumen

El presente Trabajo de Fin de Grado abarca un estudio del tráfico de la ciudad de Madrid en los años 2019, 2020 y 2021, así como la creación de modelos predictivos de la carga de sus carreteras. El estudio ha sido realizado empleando los lenguajes *R* y *Python* a través del entorno de *RStudio*. Para ello, nos hemos valido de conjuntos de datos extraídos del *Portal de datos abiertos del Ayuntamiento de Madrid*[1] tales como: intensidad de tráfico, puntos de medida de tráfico, y distritos de la ciudad de Madrid.

Primero, se ha realizado una limpieza básica de datos anómalos que corrompían nuestro estudio, tales como valores negativos, NA y otros códigos de error. Seguidamente, se ha realizado una limpieza exhaustiva basándonos en incongruencias en los datos y eliminando variables que carecían de valor para nuestro estudio. Durante todo este proceso hemos estado en contacto con los responsables del portal de datos del Ayuntamiento de Madrid, los cuales nos han resuelto ciertas dudas que les planteamos.

Tras finalizar la limpieza de los datos hemos realizado un análisis descriptivo de las variables, en el que se ha tratado de reflejar visualmente mediante gráficos y mapas el impacto que ha tenido la pandemia en el tráfico de la ciudad de Madrid, haciendo una comparativa de resultados proporcionados por los tres años.

Por último, antes del modelado, hemos tratado todos los datos para darles el formato adecuado para emplear los algoritmos de forma eficaz.

En lo que al modelo respecta, se han empleado dos técnicas como son la regresión lineal simple y la regresión logística multi-clase. Para cada una de las técnicas empleadas se han realizado varias iteraciones variando diversos parámetros hasta encontrar la combinación óptima de predictores y variables para nuestros modelos. Finalmente hemos realizado un breve análisis de los resultados obtenidos y sus implicaciones.

Palabras Clave

- Carga
- Tráfico
- Regresión
- R
- Python

Abstract

This Final Degree Project covers a study of the traffic of the city of Madrid in the years 2019, 2020 and 2021, as well as the creation of predictive models of the load of its roads. The study has been carried out using the *R* and *Python* languages through the *RStudio* environment. For this purpose, we have used datasets extracted from *the open data Portal of the Madrid City Council*[1] such as: traffic intensity, traffic measurement points, and districts of the city of Madrid.

First, a basic cleaning of anomalous data that corrupted our study, such as negative values, NA and other error codes, was performed. Next, an exhaustive cleaning was carried out based on inconsistencies in the data and eliminating variables that had no value for our study. Throughout this process we have been in contact with the responsible for the data portal of the Madrid City Council, who have resolved certain doubts that we raised.

After finishing the data cleaning, we have carried out a descriptive analysis of the variables, in which we have tried to visually reflect through graphs and maps the impact that the pandemic has had on the traffic in the city of Madrid, making a comparison of the results provided by the three years.

Finally, before modeling, we have treated all the data to give them the appropriate format to use the algorithms effectively.

As far as the model is concerned, we have used two techniques such as simple linear regression and multi-class logistic regression. For each of the techniques used, several iterations have been carried out, varying various parameters until the optimal combination of predictors and variables for our models was found. Finally, we have made a brief analysis of the results obtained and their implications.

Keywords

- Load
- Traffic
- Regression
- R
- Python

Contents

List of Figures

List of Tables

0.1 Introducción

0.1.1 Motivación de la investigación

Lo primero que decidimos fue realizar un TFG sobre análisis de datos puesto que estamos muy interesados en este campo y además ya habíamos cursado asignaturas en las que habíamos trabajado de forma similar, como Minería de datos, Gestión de información en la web y Análisis de redes sociales. El principal atractivo que le vemos a este campo es como pasas de tener un conjunto inicial de datos el cual a priori no te aporta información alguna, a extraer información útil, a la cual le puedes dar la aplicación que tú necesites.

Tras tener claro la orientación del TFG estuvimos buscando la temática, y finalmente nos decantamos por realizar un análisis de datos de tráfico del Ayuntamiento de Madrid. Los motivos que nos llevaron a elegir esta temática, son que nos resulta interesante conocer más de cerca algo que nos afecta a diario como es el tráfico de nuestra ciudad, además, el tráfico es un elemento muy interesante de estudio, puesto que a veces se comporta de forma inesperada y cuesta mucho extraer patrones acerca de él.

0.1.2 Objetivos

Los objetivos de la investigación son poder llegar a conocer algunos patrones o tendencias en el tráfico de la ciudad de Madrid, sin entrar en grandes detalles, puesto que como ya hemos mencionado el tráfico es un elemento relativamente impredecible, y por tanto, muy difícil de estudiar. Además, aprovechando la anómala situación que hemos vivido estos últimos años con la aparición de la COVID-19 y todas las implicaciones que ha llevado eso en nuestras vidas, creemos que es interesante estudiar el impacto de la pandemia en el tráfico, es por ello que hemos elegido unos datos que comprenden desde inicios del año 2019 hasta finales del año 2021. Tras ese estudio inicial de la evolución y el comportamiento del tráfico, hemos empleado diferentes modelos predictivos con el fin de poder predecir la carga de las carreteras de acuerdo a diversas variables.

0.1.3 Antecedentes

Tras tener definida la temática y de forma paralela a la definición de objetivos de nuestro estudio, nos centramos en la búsqueda de información sobre estudios previos. Los estudios más relevantes que hemos encontrado relacionados con nuestra temática, son los propios estudios publicados en la página de la Comunidad de Madrid [2]. En estos estudios se realiza un análisis del tráfico en la Comunidad de Madrid, sobretodo orientado al aforo de las vías, teniendo en cuenta el tipo de vía y vehículos que la frecuentan. Los estudios publicados en esta plataforma son de carácter anual, los cuales se iniciaron con la publicación de un primer informe en el año 2015. Actualmente (Abril

2022) están disponibles los informes del año 2015, 2016, 2017 y 2018, por lo que el periodo de tiempo 2019-2021 no está cubierto, lo que nos ha dado un motivo más para seleccionar esta franja de tiempo para trabajar.

También encontramos un estudio[3] en el Portal de transparencia del Ayuntamiento de Madrid, donde se analiza, diferenciando entre los distintos distritos y tipos de vía, la velocidad e intensidad de las vías, esta última desglosando las zonas interurbanas y analizándolas más detalladamente. Además, se estima la matriz de desplazamientos en coche, donde se analizan las zonas (urbana o interurbana) donde tienen lugar el origen y destino de estos viajes.

En este artículo [4] de *El País*, se recoge un estudio que realiza una comparativa de los datos de tráfico antes y después de la pandemia, más concretamente entre los años 2019 y 2021, haciendo alusión a la influencia del teletrabajo y al posible incremento del vehículo privado por miedo ante el riesgo de contagio. En este artículo trabajan con los mismos datos que los empleados en este proyecto, pero dándole un enfoque centrado más en la movilidad y las distintas alternativas de transporte como son las bicicletas y el transporte público de los cuales se extraen patrones y tendencias durante el periodo de la pandemia.

0.1.4 Plan de trabajo

Al comenzar el desarrollo de este proyecto planteamos dos posibles formas de orientarlo. La primera era realizar dos estudios sobre conjuntos de datos distintos (cada uno realizado por un integrante distinto), y posteriormente realizar una comparativa de las conclusiones obtenidas. Las ventajas de este planteamiento era que íbamos a tener diferentes perspectivas y diferentes formas de enfocar una misma problemática, lo cual era muy interesante para el análisis final comparando ambos trabajos. Por otro lado, la problemática que surgía, era que era muy ambicioso, ya que prácticamente implicaría dos TFGs en uno, y además omitíamos el factor del trabajo en grupo el cual es muy interesante, sobre todo hoy en día en un mundo en el que el trabajo grupal es imprescindible para realizar prácticamente cualquier proyecto de manera exitosa. Por lo que, finalmente, optamos por la opción de realizar un trabajo de forma conjunta, trabajando sobre el mismo conjunto de datos y de manera sincronizada.

Para la distribución de tareas, al inicio optamos por realizar reuniones vía Discord de forma que trabajábamos de forma conjunta, lo que nos permitía tener absoluto conocimiento de todas las partes del proyecto en todo momento. Esta forma de trabajo la mantuvimos durante los primeros meses, ya que el inicio del proyecto nos parecía lo más importante para ir familiarizándonos con los datos y el proyecto en general. Posteriormente, cambiamos esa forma inicial de trabajo y optamos por realizar una reunión a la semana en la que simplemente comentábamos el trabajo que habíamos estado realizando durante esos días y los problemas encontrados, y en caso de haberlos podido resolver, como lo habíamos hecho. En esa reunión también compartíamos propuestas sobre como enfocar el trabajo, y se acordaban nuevas vías de trabajo si era necesario,

por último, concretábamos las tareas a realizar antes de la siguiente reunión.

Para ayudarnos a coordinarnos y organizarnos mejor, hemos empleado herramientas con las que ya habíamos trabajado previamente, como los repositorios de Github, Google Drive... Además, investigamos acerca de nuevas herramientas que nos pudiesen ser útiles y descubrimos la herramienta de Trello que permite tener como un bloc de notas común, en el cual apuntábamos las dudas, las tareas completadas hasta la fecha, las tareas en proceso, futuros pasos del proyecto, enlaces de interés etc. Además, nos ayudamos de diagramas con las tareas a realizar y la fecha estimada para la que tenían que estar completadas, para de esta forma, llevar un mejor control de los plazos e intentar minimizar los imprevistos dentro de los posible.

En la figura 1 se muestra un ejemplo de uno de los diagramas de Gantt que hemos empleado.

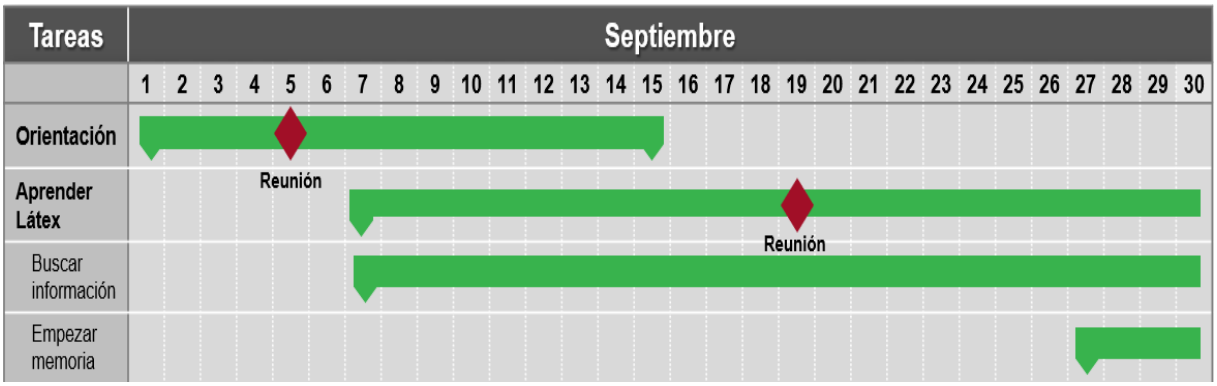


Figure 1: Diagrama de planificación del mes de Septiembre 2021

0.1.5 Estructura de la memoria

La memoria se ha estructurado de la siguiente forma:

En el punto 1 se exponen la motivación, objetivos y planteamiento del desarrollo de este proyecto, así como estudios previos relacionados con nuestro proyecto y la estructura de la memoria.

En el punto 2 se describen tanto la metodología elegida para el desarrollo del proyecto, como las distintas herramientas empleadas para ello.

En el punto 3 se explica más a fondo la metodología elegida, así como los algoritmos empleados para nuestro modelo predictivo.

En el punto 4 se realiza una exploración de los datos, un proceso de tratamiento en el que se limpian los datos, y se formatean y crean variables. Posteriormente se realiza un análisis descriptivo de las variables más significativas.

En el punto 5 se explica la integración de R con Python, tanto en el desarrollo del

mapa de calor como en el modelo de regresión logística multi-clase.

En el punto 6 se detallan las distintas fases del modelado de los datos.

En el punto 7 se evalúan los resultados obtenidos del modelo predictivo.

En el punto 8 se exponen las conclusiones y trabajo futuro de este proyecto.

En el punto 9 se recogen las aportaciones individuales de cada miembro.

0.1 Introduction

0.1.1 Motivation

The first thing we decided was to carry out a TFG on data analysis since we are very interested in this field and we had already studied subjects in which we had worked in a similar way, such as Data mining and the Big Data paradigm, Web Information Management and Social Network Analysis. The main attraction that we see to this field is how you go from having an initial set of data, which a priori does not give you any information, to extract useful information, to which you can give the application you need. In this way, for example, companies have been able to improve manufacturing processes, marketing campaigns, identify fraud and any another application that we imagine.

After being clear about the orientation of the TFG, we were looking for the theme, and finally we opted to carry out a traffic data analysis from the Madrid City Council. The reasons that led us to choose this theme are that we find it interesting to learn more about something that affects us daily such as the traffic in our city, In addition, traffic is a very interesting element of study, since sometimes it behaves unexpectedly and it is very difficult to extract patterns about it.

0.1.2 Goals

The goals of the research are to be able to get to know some patterns or tendencies in the traffic of the city of Madrid, without going into detail, as we have already mentioned, traffic is a relatively unpredictable element, and therefore hard to study.

In addition, taking advantage of the anomalous situation that we have experienced in this recent years with the appearance of COVID-19 and the impact it has had on our lives, we believe it is interesting to study the impact of the pandemic in traffic, that is why we have chosen the data from beginning of the year 2019 until the end of the year 2021.

After this initial study of traffic evolution and behavior, we have used different

predictive models in order to be able to predict the road load according to different variables.

0.1.3 Background

After having defined the theme alongside the goals of our study, we focused on the search of information on previous studies. The most relevant studies that we have found related to our subject, are the studies themselves published on Community of Madrid's website. These studies carry out an analysis of traffic in the community of Madrid[2], especially oriented to the roads' capacity, considering the road's type and vehicles that frequent it. The studies published on this platform are annual, starting in 2015. Currently (April 2022) the reports for the years 2015, 2016, 2017 and 2018 are available, so the 2019-2021 period is not covered, which has given us one more reason to select this time slot to work.

We also found a study[3] in the Transparency Portal of the Madrid City Council, where it is analyzed, differentiating between the different districts and types of roads, and intensity of the roads, the latter breaking down the interurban areas and analyzing them in more detail. In addition, the matrix of car trips is estimated, in which areas (urban or interurban) where the origin and destination of these trips take place.

In this article[4] from *El País*, there is a study that compares the traffic data before and after the pandemic, more specifically between the years 2019 and 2021, referring to the influence of teleworking and the possible increase in the use of private vehicles out of fear of the risk of contagion. In this article they work with the same data as those used in this project, but with a more focused approach on mobility and the different transport alternatives such as bicycles and public transport, from which patterns and trends are extracted during the period of the pandemic.

0.1.4 Work dynamics

At the beginning of the development of this project we posed two possible ways of guiding it. The first one was to conduct two studies on different data sets (each one conducted by a different member), and then compare the conclusions obtained. The advantages of this approach were that we were going to have differents perspectives and ways of approaching the same problem, which was very interesting for the final analysis comparing both studies. At the same time, the problem that arose was that it was very ambitious, given the fact that it would probably involve two TFGs in one, and we would also omit the team work factor, which is very interesting, especially nowadays where team work is essential to develop a project successfuly. And thats why we opted for the option of working together, synchronously, on the same set of data.

For the distribution of tasks, at the beginning we chose to hold meetings via Discord so we could work together, which allowed us to have absolute knowledge of all parts of

the project at all times. We kept this work method during the first months, because at the beginning of the project, it seemed to us it was the most important way to familiarize with the data and the whole project. Later, we changed that initial way of working and opted to hold a meeting per week, in which we simply commented on the work that we had been doing during those days and the problems encountered, and in case of having been able to solve them, explained the solution. At these meetings we also shared suggestions on how to approach the work, and new ways of working were agreed upon if necessary. Finally, we specified the tasks to be carried out before the next meeting.

To help us coordinate and organize ourselves better, we have used tools we had worked previously with such as Github repositories, Google Drive... In addition, we investigated about new tools that could help us be useful and we discovered *Trello*, that allows us to have a shared notebook, in which we wrote down the doubts, the tasks completed so far, the tasks in process, future steps of the project, links of interest, etc. In addition, we help ourselves with diagrams with the tasks to be carried out and the estimated due date, in order to have a better control of the deadlines and try to minimize unforeseen events as much as possible.

Figure 1 shows an example of one of the Gantt charts that we have used.

0.1.5 Document's structure

This document has been structured as follows:

Section 1 describes the motivation, objectives and approach to the development of this project, as well as previous studies related to our project and the structure of the document.

Section 2 describes both the methodology chosen for the development of the project and the different tools used for this purpose.

Section 3 explains in more detail the chosen methodology and the algorithms used for our predictive model.

In section 4, a data exploration, a treatment process in which the data are cleaned, and variables are formatted and created, is carried out. Subsequently, a descriptive analysis of the most significant variables is performed.

Section 5 explains the integration of R with Python, both in the development of the heat map and in the multi-class logistic regression model.

Section 6 details the different phases of data modeling.

Section 7 evaluates the results obtained from the predictive model.

Section 8 presents the conclusions and future work of this project.

Section 9 contains the individual contributions of each member.

0.2 Metodología y herramientas

En esta sección explicamos que metodologías hemos seguido a lo largo de la realización del proyecto, así como las herramientas que hemos empleado durante todo el proceso de desarrollo, además de los principales motivos que nos han llevado a elegirlas frente a las otras opciones de las que disponíamos.

0.2.1 Selección de nuestra metodología de trabajo

Las metodologías de desarrollo para minería de datos más conocidas son SEMMA, CRISP-DM y KDD. Cada una de ellas se diferencia por sus fases y planteamientos para desarrollar un proyecto:

Tras investigar sobre todas ellas y realizar una comparativa tal y como se muestra en la tabla 1, pensamos que la que más se ajusta a nuestro tipo de proyecto es CRISP-DM (Cross-industry Standard Process for Data Mining) ya que nos permite cubrir todas las fases del proyecto, sus tareas respectivas, y poder hacer relaciones entre estas tareas.

Pensamos que es una metodología muy adecuada para nuestro proyecto, ya que la secuencia de las fases no es rígida, pudiendo volver a la etapa anterior para realizar otra iteración si fuese necesario, lo que nos permite mucha flexibilidad a la hora de trabajar en un proyecto como este, en el cual estamos en constante descubrimiento de nuevas necesidades y problemáticas.

	KDD	**SEMMA**	**CRISP**
Flexibilidad	Esquema en cascada, secuencia de fases más rígida	Alta	Alta
Complejidad	Más complejo de implementar que los otros dos, tiene una cantidad considerable de fases a desarrollar	Simple y bastante ágil, sus fases están más implementadas a desarrollo ágil	Es el menos complejo de entender y aplicar, cuenta con una curva de adaptabilidad muy amplia para cualquier desarrollador
Número de fases	9	6	6
Siglas	Knowledge Discovery in Databases	Sample, Explore, Modify, Model and Access	Cross-industry Standard Process
Relevancia actual	Baja	Media	Alta

Table 1: Comparativa de las principales metodologías para el Data Science

0.2.2 Herramientas empleadas

En este apartado explicaremos las herramientas y tecnologías empleadas durante el desarrollo de este proyecto, así como las decisiones que nos han llevado a elegirlas sobre otras alternativas similares.

0.2.2.1 Lenguajes de programación empleados

Los principales lenguajes para el tratamiento de datos son Python y R, ya que tienen un amplio soporte de librerías y disponen de múltiples funciones orientadas a realizar Data Science. Además, son herramientas gratuitas y con una amplía aceptación en la comunidad, por lo que hay infinidad de proyectos disponibles en internet, que aunque no se centren en la misma temática, podemos estudiar para coger ideas, solventar dudas y tener en definitiva más facilidad a la hora de trabajar. Además, de esta forma, el proyecto es escalable, permitiendo nuevas orientaciones o incluso ampliaciones del estudio ya realizado.

En la tabla 2 mostramos una comparativa entre estos dos lenguajes, centrándonos en las ventajas e inconvenientes que nos aportan al emplearlos en el desarrollo de nuestro proyecto. De esta forma podemos tener una visión más clara de las necesidades que tenemos y las facilidades o inconvenientes de emplear cada uno de los lenguajes mencionados.

	R	**Python**
Pros	1) Muchos paquetes estadísticos 2) Más potente para visualizar datos	1) Más intuitivo 2) Mucha documentación
Contras	1) Mayor complejidad 2) Ejecuciones más lentas	1) Menos potente

Table 2: Comparativa de los principales lenguajes para Data Science

Tras analizar que lenguaje se adaptaba mejor a los requerimientos del proyecto, optamos por emplear R como lenguaje principal y dejamos *Phyton* relegado a un segundo plano para realizar algunas funciones concretas de acuerdo a necesidades puntuales.

Por otro lado, aunque no sea un lenguaje de programación como tal, hemos empleado Latex como herramienta para documentar todo nuestro trabajo. A pesar de no haber trabajado con Latex previamente, hemos considerado que aporta un toque más profesional y adecuado para este trabajo que otras herramientas como Word.

0.2.2.2 Entornos de trabajo

Para el uso de R hemos empleado el entorno de RStudio, el cuál nos ha ayudado en las labores de visualización y carga de datos, así como una interfaz más intuitiva con la que trabajar. El desarrollo de la totalidad del código lo guardábamos en un script el cúal teníamos asociado a un repositorio de Github, de forma que podíamos trabajar de forma dinámica sin contratiempos con los backups y ediciones simultaneas. Y en caso de ser necesario realizar pruebas concretas, las ejecutábamos a través de la consola de forma local, con lo que no alterábamos el código, eliminando los riesgos de dejarlo modificado de forma no intencionada.

Además, para el empleo puntual de Pyhton fuera del entorno de Rstudio hemos empleado la herramienta Pycharm la cuál nos brinda ciertas facilidades a la hora de trabajar. Otras alternativas similares son Visual Studio Code y editores de texto similares.

Para realizar el diagrama de flujo de nuestros datos hemos utilizado Microsoft Visio,

ya que nos pareció una aplicación muy intuitiva y fácil de personalizar.

0.2.2.3 Herramientas para la comunicación y sincronización

- **Discord:** Esta aplicación la usamos como principal medio de comunicación, en ella realizábamos nuestras reuniones mediante llamadas y usábamos el chat para ir anotando las dudas que surgían y las tareas a realizar.
- **Google Meet:** Esta aplicación la usamos para realizar reuniones con nuestra tutora, de manera que estuviese al día con nuestros avances, poder resolver nuestras dudas y realizar correcciones.
- **Trello:** En esta aplicación organizamos nuestras tareas por bloques, de manera que podíamos gestionar fácilmente el estado de las tareas moviéndolas entre los distintos bloques según fuéramos avanzando en el desarrollo de nuestro proyecto.
- **Github:** Esta aplicación la usamos para llevar un control de versiones de nuestro código.
- **Drive:** Esta aplicación la usamos para almacenar nuestro proyecto entero, tanto el proyecto de R, como la memoria. También guardábamos documentación útil para el desarrollo del proyecto.

0.2.2.4 Repositorio de trabajo

Con el objetivo de que este proyecto sea escalable y reproducible, hemos tratado de documentar todos los pasos seguidos en su realización, tanto a nivel de fundamentos teóricos y estudio, como de desarrollo de código el cual cuenta con comentarios explicativos también.

La estructura de nuestros datos consta de tres carpetas:

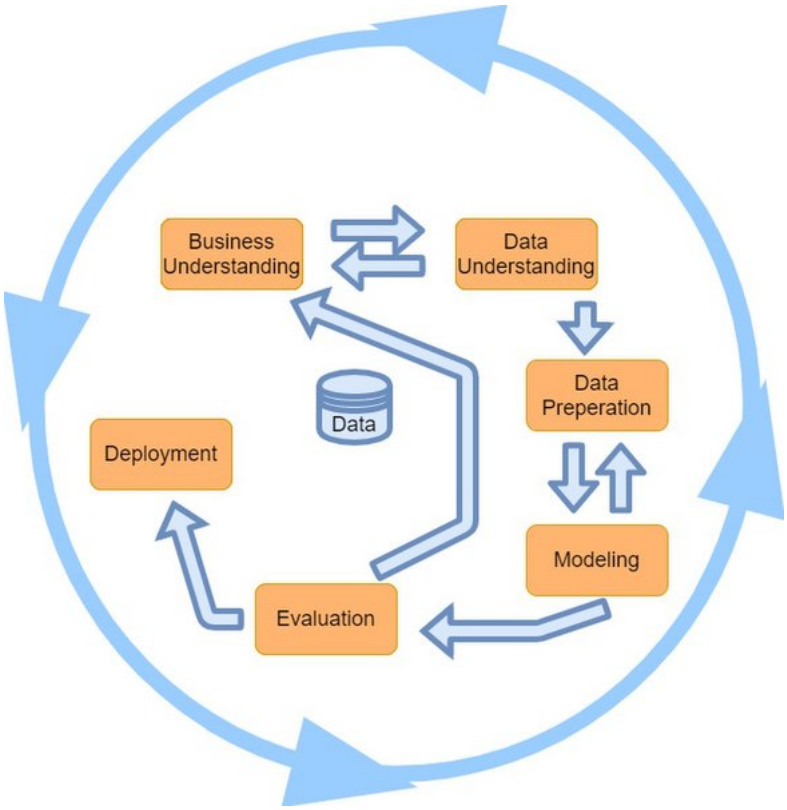
- **Distritos**: En esta carpeta guardamos los archivos *.shp* correspondiente a los distritos de Madrid.
- **Tráfico**: En esta carpeta se guardan los archivos *.csv* de tráfico divididos en tres carpetas, una por año, y en cada una de ellas doce archivos correspondientes a los meses.
- **Ubicacion_de_los_puntos_de_medida_del_trafico**: En esta carpeta se guardan los archivos *.shp* correspondientes a los puntos de medida de tráfico, divididos en tres carpetas, una por año, y en cada una de ellas los archivos correspondientes a los meses.

0.3 Fundamento teórico

Como parte del trabajo, a continuación desarrollamos las bases teóricas que respaldan a todos los procesos, algoritmos y metodologías que hemos empleado.

0.3.1 Metodología CRISP-DM

Tras comentar los motivos de la elección de esta metodología, a continuación explicaremos en que consiste, así como de las fases que está formada. En la figura 2 podemos observar un esquema a grandes rasgos de esta metodología.



Fuente: Alexander Schröder (2021), CC BY-SA 4.0 via Wikimedia Commons

Figure 2: Esquema de la metodología CRISP-DM

A continuación explicaremos en que consiste cada una de estas 6 fases:

1. **Comprensión del negocio**

   Comprensión de los objetivos y requerimientos

   Valoración de la situación

Objetivos de Minería de Datos

2. **Estudio de los datos**

   Obtención del conjunto inicial de datos

   Exploración del conjunto de datos

   Identificar la de calidad de los datos

3. **Preparación de datos**

   Selección de datos

   Limpieza de datos

   Construcción de los datos

   Formateo de los datos

   Definición del datawarehouse

4. **Modelado**

   Implementación en herramientas de Minería de Datos.

5. **Evaluación**

   Determinar si los resultados coinciden con los objetivos

   Identificar las temas de negocio que deberían haberse abordado

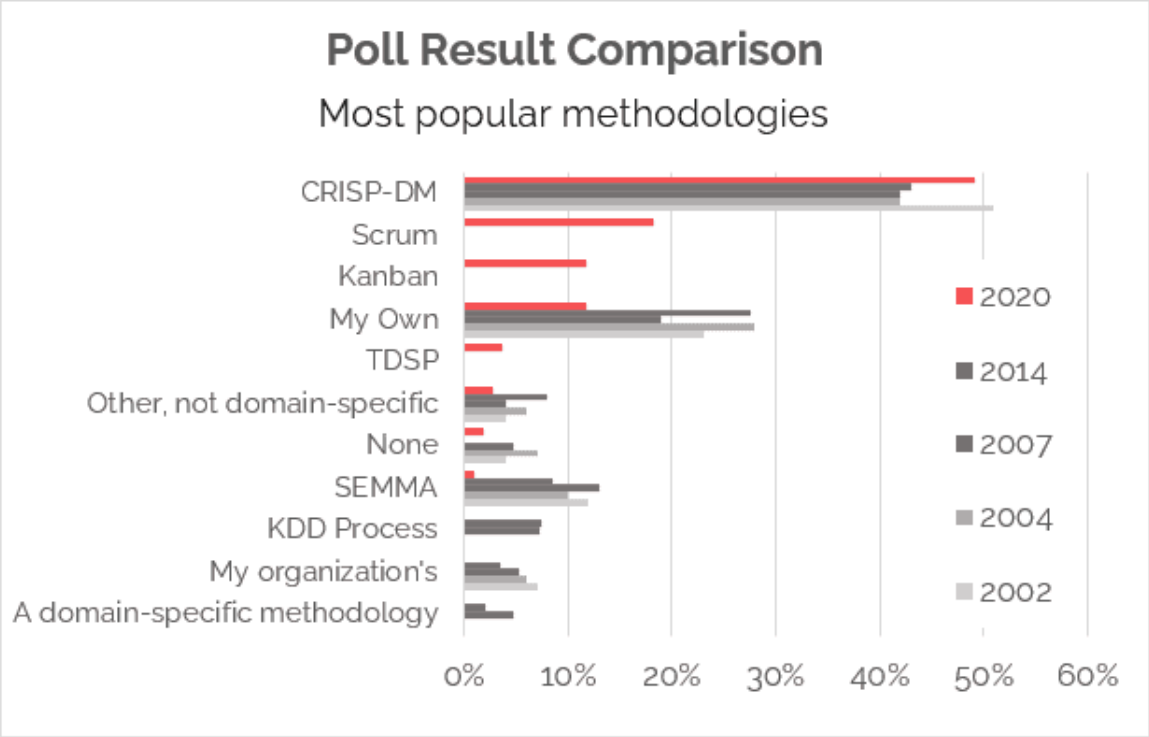
6. **Despliegue**

   Instalar los modelos resultantes en la práctica

   Configuración para minería de datos de forma repetida

Como podemos observar, la secuencia de fases se ajusta muy bien y es muy coherente con nuestro proyecto, por lo que con ligeros ajustes eliminando la orientación a negocio y obviando el paso del datawarehouse (ya que no necesitamos almacenar los datos de tal forma), podemos implementarla y seguirla de forma muy realista.

A pesar de la fuerza que ha cobrado la metodología Scrum en los últimos años, CRISP-DM se mantiene al frente como principal elección para los proyectos de minería de datos, tal y como refleja la encuesta realizada por *Data Science Process Alliance* en 2020.



Fuente: Data Science Process Alliance (2020). Poll result comparision.

Figure 3: Encuesta realizada por Data Science Process Alliance en 2020

0.3.2 Métricas de error

Las siguiente métricas han sido empleadas como indicador de calidad de las iteraciones realizadas mediante regresión lineal múltiple.

Para la comparación del modelo final obtenido de la regresión lineal con el modelo final de la regresión logística multi-clase, hemos empleado otra métrica distinta (el % de acierto del modelo), por problemas de compatibilidad como explicamos más adelante.

- **MSE**: El **error cuadrático medio** calcula la media de la diferencia cuadrada entre el valor real y el valor estimado, de manera que un error 0 sería un modelo perfecto, y cuanto mayor sea este error peor será el modelo.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

  donde y_i es el valor real y $\hat{y}_i$ el valor estimado.

- **MAE**: El **error absoluto medio** calcula la media de la diferencia absoluta entre el valor real y el valor estimado. Es una puntuación lineal, por lo que cada una

de sus diferencias se ponderan por igual en el promedio.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

donde y_i es el valor real y $\hat{y}_i$ el valor estimado.

0.3.3 Algoritmos empleados

En esta sección explicaremos como funcionan los diferentes algoritmos que hemos valorado emplear en nuestros modelos, y cuales han sido finalmente aquellos que hemos empleado.

0.3.3.1 Regresión Lineal Múltiple

"La regresión lineal múltiple permite generar un modelo lineal en el que el valor de la variable dependiente o respuesta (Y) se determina a partir de un conjunto de variables independientes llamadas predictores (X1, X2, X3...). Es una extensión de la regresión lineal simple. Los modelos de regresión múltiple pueden emplearse para predecir el valor de la variable dependiente o para evaluar la influencia que tienen los predictores sobre ella"[5] .

0.3.3.2 Regresión Logística

La regresión logística es una técnica de clasificación binaria ampliamente usaba en Machine Learning. Al igual que la regresión lineal simple emplea uno o varios predictores para predecir la variable dependiente. Sin embargo, su principal diferencia respecto a la regresión lineal simple, es que que realiza una predicción retornando un valor en el intervalo [0,1], el cual indica la probabilidad de pertenecer a la clase que estamos estudiando. Un ejemplo sencillo sería si por ejemplo estamos prediciendo la probabilidad de tener COVID-19, un valor de 0.9 significa que el modelo ha estimado que tenemos un 90% de tener COVID-19.

En nuestro caso, puesto que lo que deseamos predecir es la carga, que es una variable continua (valores en el intervalo [0-100]), lo que hemos hecho es discretizar esa carga de tal forma que la hemos transformado en una variable discreta, agrupándola en 4 rangos: [0-24],[25-49],[50-74] y [75-100], por tanto, la información que va a recibir el modelo, es para cada fila (registro de tráfico) si pertenece a cada una de las clases o no, por lo que contamos con 4 columnas (cada una representando un rango de carga) de las cuales solo una de ellas va a tener el valor 1 (indicando que pertenece a dicha clase) y las otras tres contendrán el valor 0. En las figuras 4 y 5 se muestra una comparativa de antes y después de la transformación.

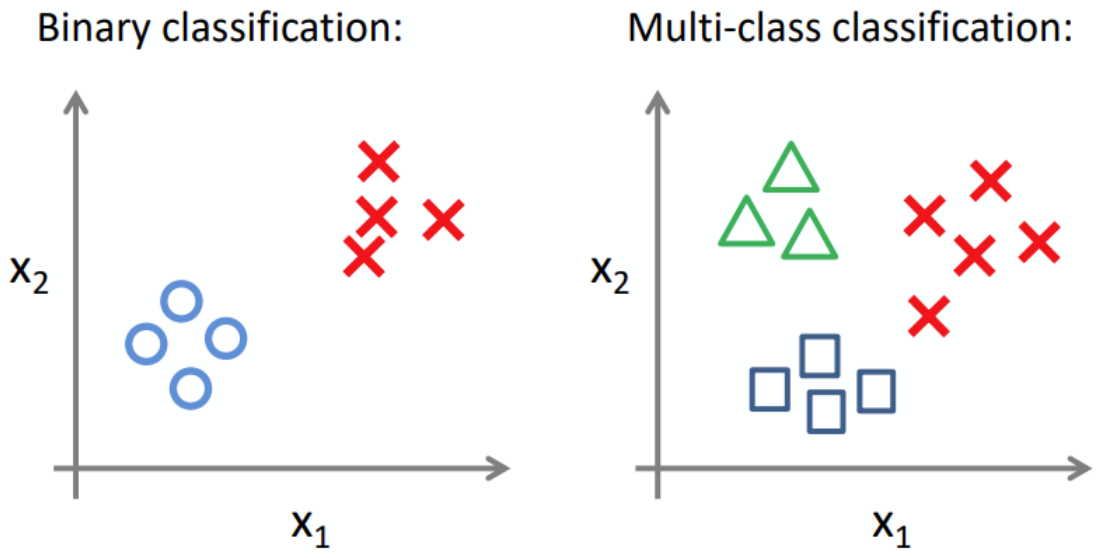
carga
6
37
31
18
92
33
9
68
92
48
24
16

Figure 4: Carga sin categorizar

carga_0_24	carga_25_49	carga_50_74	carga_75_100
1	0	0	0
0	1	0	0
0	1	0	0
1	0	0	0
0	0	0	1
0	1	0	0
1	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0
1	0	0	0
1	0	0	0

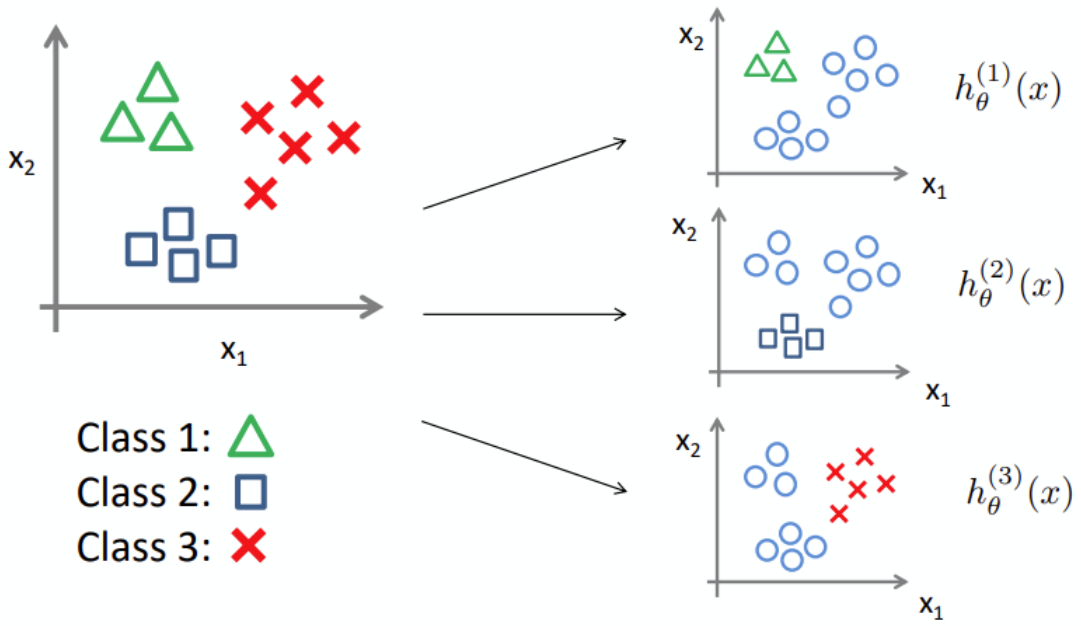
Figure 5: Carga categorizada

Como ya hemos mencionado, la regresión logística permite identificar si nuestra variable carga pertenece o no a la clase que estamos estudiando, pero, como nosotros contamos con 4 clases, es necesario aplicar la técnica conocida como *One vs All*. Esta técnica consiste en realizar el mismo proceso pero repetido tantas veces como clases tengamos. Para ello, en cada iteración entrenamos al modelo con nuestros datos de entrenamiento, pero, con una ligera modificación en la estructura de datos, de tal forma que por ejemplo, para una primera iteración en la que deseamos entrenar al modelo para la clase [0-24], mantenemos con valor 1 todas las filas que pertenecen a esa clase [0-24], pero el resto de filas que no pertenecen a esa clase [25-49][50-74][75-100] se marcan con 0. En la figura 7 se muestra un ejemplo visual para aclarar este proceso de transformación.



Fuente: Andrew Ng. Aprendizaje Automático(Universidad de Stanford).

Figure 6: Regresión logística binaria vs Regresión logística multi-clase



Fuente: Andrew Ng. Aprendizaje Automático(Universidad de Stanford).

Figure 7: Método One vs All

Una vez aplicada esta técnica, nuestro modelo está entrenado para todas las clases,

es decir al igual que la regresión lineal simple, el modelo ha establecido unos pesos a cada uno de los predictores, además de una variable de sesgo. El siguiente paso es probar el modelo con el conjunto de validación, de tal forma que el modelo nos va a retornar 4 valores para cada fila o muestra. Cada uno de esos 4 valores es la probabilidad de pertenecer a cada una de las 4 clases con las que contamos.

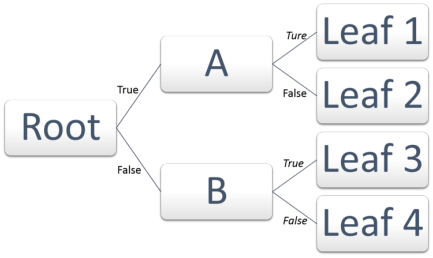
Puesto que, cuanto más cerca de 1 esté cada valor, más probable es que pertenezca a esa clase en cuestión, concluimos la predicción quedándonos con el valor más alto de estas 4 predicciones, ya que es la clase más probable según nuestro modelo.

Una vez concluida la predicción solo falta realizar la comprobación, para ello comparamos la clase real de cada una de nuestras muestras o registros, con la clase que nuestro modelo ha predicho, y realizando una regla de tres, obtenemos el porcentaje de acierto de nuestro modelo.

0.3.3.3 Algoritmos descartados

Estos son algunos algoritmos que barajamos emplear en nuestro proyecto, pero que finalmente fueron descartados. Los motivos que nos llevaron a prescindir de ellos fueron que era imposible emplear todas las técnicas existentes ya que no disponíamos de tanto tiempo, por lo que tuvimos que centrarnos en las escogidas para este trabajo.

- **Árboles de decisión**: Es un algoritmo muy popular en tareas de regresión o clasificación, ya que es fácil de usar y muy potente. Consiste en la representación gráfica en forma de estructura de árbol, en el que se representan, siguiendo unas condiciones, la solución a una decisión. De manera que queda una estructura que comienza con un nodo inicial (raíz) y se va bifurcando en nuevas ramas, que representan una decisión, hasta llegar a los nodos finales (hojas) que serían la solución a la decisión planteada.

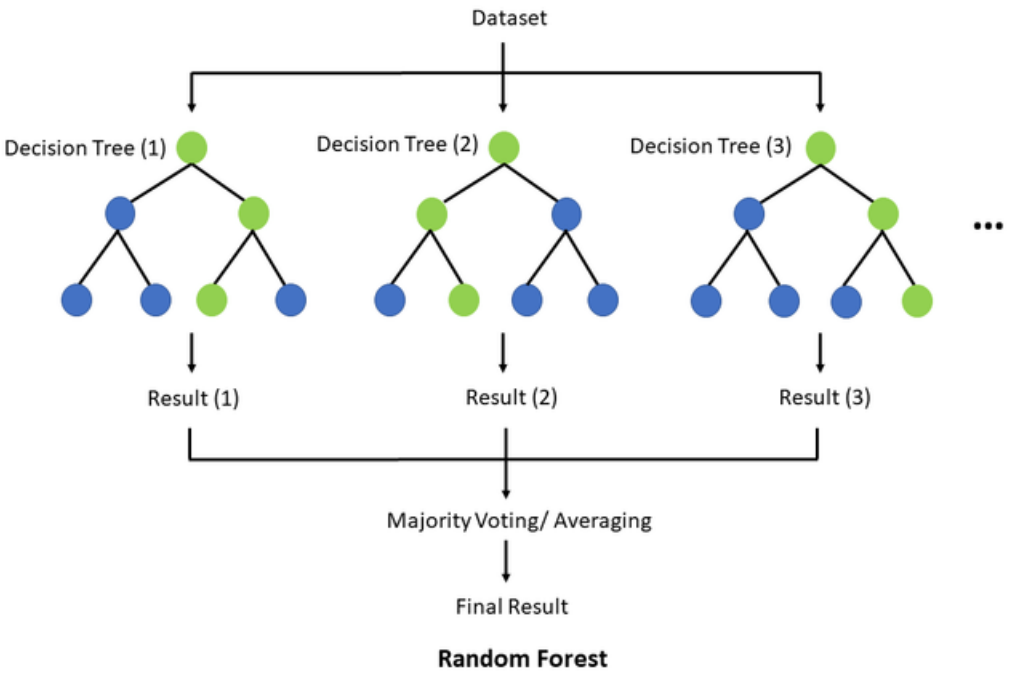


Fuente: Pkuwangyan06 (2015), CC BY-SA 4.0 via Wikimedia Commons

Figure 8: Árbol de decisión

- **Random Forest**: Combinación de conjuntos de árboles de decisión en la que distintos conjuntos de datos son vistos por distintos árboles, de manera que, para

un mismo problema, cada árbol se entrena con datos distintos. Finalmente se combinan todos sus resultados y, por ejemplo, mediante un sistema de votación, se escoge el resultado más votado.



Fuente: TseKiChun (2021), CC BY-SA 4.0 via Wikimedia Commons

Figure 9: Random Forest

0.4 Exploración y procesamiento de los datos

Posiblemente es la parte más importante del proyecto, ya que el trabajo realizado aquí afecta al desarrollo de la totalidad del trabajo. Es por ello, que ha sido uno de los puntos donde hemos centrado nuestra atención y al que más tiempo hemos dedicado.

Primero, comenzamos con un análisis básico de nuestros datos, empleando diversas funciones para conocer más acerca de nuestros datos. Nos centramos en identificar máximos, mínimos, valores anómalos, tendencias en los datos... También, exploramos cualquier valor que implicase un código de error en los datos, así como su localización y distribución en el conjunto de datos. Realizamos funciones para observar la correlación entre nuestras variables y ver si era factible continuar con el planteamiento original de predecir la carga. Toda esta fase nos permitió tener una visión más profunda de nuestros datos y poder enfocar el proyecto de una forma adecuada.

Tras todo el análisis inicial seguido de un paso a gráficas y esquemas, realizamos una

segunda fase de limpieza más exhaustiva en la cual identificamos errores de coherencia en los datos basándonos en nuestro conocimiento adquirido hasta el momento. Tras la detección de múltiples incoherencias y algunas de ellas sin explicación alguna, nos pusimos en contacto con los responsables del Portal de Datos Abierto del Ayuntamiento de Madrid[1], los cuales nos resolvieron las dudas que les formulamos vía correo electrónico.

Posteriormente tras estas primeras fases de limpieza, tomamos la decisión de realizar un agrupamiento de los registros que teníamos, ya que las muestras eran en periodos de 15 minutos, y en lo referente a datos de circulación, nos parecía una franja demasiado precisa, por lo que agrupamos todos nuestros datos en periodos de 1 hora, lo que implicó recalcular todas las variables, pero, que a su vez nos permitió reducir el conjunto de datos a 1/4 de su tamaño original.

El último paso, fue la preparación de todo el conjunto de datos para el entrenamiento de los algoritmos, para ello fue necesario realizar adaptaciones concretas para el correcto funcionamiento en cada uno de ellos. Este paso requirió mucho trabajo debido a que teníamos que ser extremadamente cuidadosos de mantener el mismo conjunto de datos (y los mismos subconjuntos de entrenamiento y validación) para todos los algoritmos, ya que, sino las comparativas entre ellos no serían relevantes. Durante el desarrollo de esta etapa fue donde surgieron la mayoría de problemáticas asociadas a este trabajo. El primer problema que surgió fue que a pesar de todas las limpiezas aplicadas, contábamos con un volumen de datos demasiado grande para poder trabajar con los algoritmos, lo que implicaba fallos de memoria y de tiempo de ejecución en nuestros equipos, y dado que la idea de este proyecto es que sea escalable y que cualquier particular pueda trabajar con él, optamos por crearnos un subconjunto de datos de entorno a 1.5 millones de filas, las cuales fueron escogidas de forma pseudo-aleatoria, para evitar el sesgo y mantener el balance de los datos. Aclarar que, a pesar de trabajar con subconjuntos, hemos guardado todos y cada uno de los datasets para que en caso de poder y querer reproducir el estudio con la totalidad de datos, sea posible. Además, como ya hemos mencionado que hemos empleado una función pseudo-aleatoria para la creación de subconjuntos de una forma adecuada, en caso de querer obtener exactamente los mismos resultados que los mostrados en el presente estudio, es necesario importarse todos los datasets contenidos en la carpeta *final_data* o de lo contrario trabajareis con conjuntos distintos a los nuestros, lo que implicará una ligera variación de los resultados.

0.4.1 Estudio y comprensión de los datos

En este apartado se expone toda la fase inicial de preparación y estudio de nuestros datos, explicando en detalle los pasos seguidos.

0.4.1.1 Obtención del conjunto inicial de datos

Los datos los hemos obtenido del Portal de Datos Abiertos del Ayuntamiento de Madrid [1], concretamente trabajamos con datos referentes al tráfico en el periodo enero 2019

hasta diciembre 2021. El motivo de elegir esa franja de tiempo es que contamos con un gran volumen de datos para nuestro estudio (ver tabla 3). Además, podemos estudiar el periodo que comprende a la COVID-19 ya que partimos de un periodo anterior a su aparición en España y más concretamente en Madrid (entorno a febrero-marzo de 2020).

0.4.1.2 Descripción de los datos

Para realizar nuestro estudio hemos empleado los siguientes conjuntos de datos: *tráfico diario, ubicación de los puntos de medida del tráfico y distritos* La estructura de los archivos que contienen dichos conjuntos de datos está esquematizada en la Figura 10. Además, en la Tabla 3 se ofrece una breve descripción de los datos de partida.

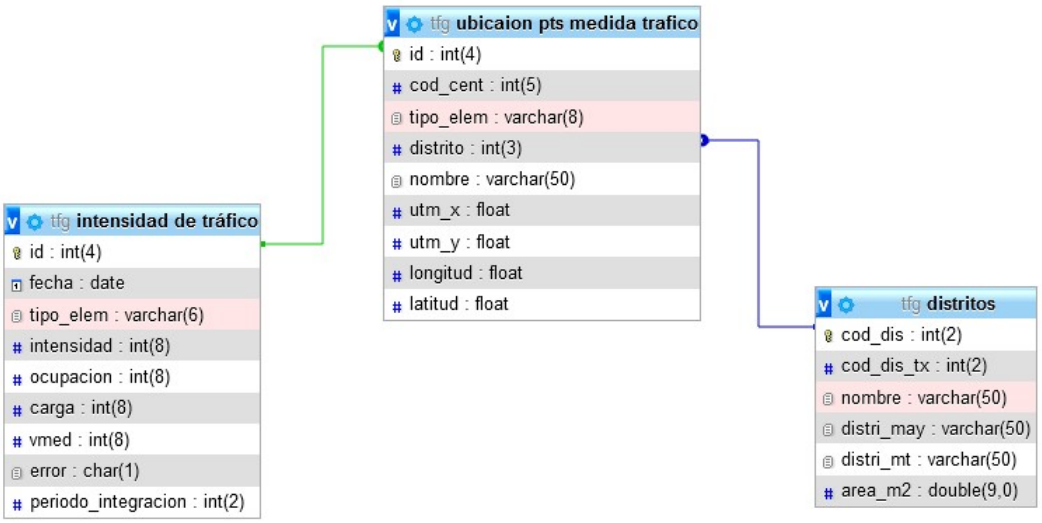


Figure 10: Representación de las tablas realizada con la herramienta phpMyAdmin

Archivo	Registros	Campos
distritos	21	6
pmed_trafico_MM_YYYY	≈ 4 500 por mes	9
trafico_MM_YYYY	≈ 1 100 000 por mes	9

Table 3: Descripción de la estructura de los archivos descargados

Descripción de Archivos

1) Archivos denominados `trafico_MM_YYYY`

Estos archivos muestran las medidas de tráfico de la ciudad de Madrid en la M30 y en la zona urbana en periodos de 15 minutos.

La medición se realiza mediante un sistema de espiras colocadas debajo del asfalto separadas por 2.5 metros. Las espiras son cuadrados de 2 metros de lado que, entre otros, se encargan de medir la velocidad de los vehículos, su tamaño, peso y el tiempo que permanecen ocupadas durante esa medición.

En la tabla 4 explicamos en detalle cada variable y los valores que toman. Para dicha tabla, la variable error, resaltada con el símbolo *, toma los siguientes valores:

N: no ha habido errores ni sustituciones
E: los parámetros de calidad de alguna de las muestras integradas no son óptimos
S: alguna de las muestras recibidas era totalmente errónea y no se ha integrado

Variable	**Descripción**	**Valores**
id	Identificación única del Punto de Medida en los sistemas de control del Ayuntamiento de Madrid	4372
fecha	Fecha y hora oficiales de Madrid	[01-01-2019 00:00, 31-12-2021 23:59]
tipo_elem	Tipo de vía	M30 y URB
intensidad	Número de vehículos que han pasado por el Punto de Medida en el periodo de 15 minutos (se hace una estimación de vehículos/hora)	[0, 9 420]
ocupacion	Porcentaje de tiempo que el Punto de Medida permanece ocupado en el periodo de 15 minutos	[0%, 100%]
carga	Parámetro que establece el grado de uso de la vía durante el periodo de 15 minutos. Se calcula de acuerdo a la intensidad, ocupación y capacidad de la vía.	[0, 100]
vmed	Velocidad media de los vehículos en el periodo de 15 minutos (Km/h). Sólo para puntos de medida interurbanos (M30)	[0, 246]
error*	Indicación de si ha habido al menos una muestra errónea o sustituida en el periodo de 15 minutos.	N, E, S
periodo integracion	Número de muestras recibidas y consideradas para el periodo de integración.	[1, 30]

Table 4: trafico_MM_YYYY

2) pmed_trafico_MM_YYYY

Contiene los datos de la ubicación de cada uno de los puntos de medida que recogen datos del tráfico de la ciudad de Madrid (calle, coordenadas, barrio...). Este archivo nos ha permitido enlazar nuestros puntos de medida con su localización geográfica, con lo que hemos podido crear ciertas representaciones tales como mapas, asociándolos a su vez al archivo de los distritos.

Variable	Descripción	Valores
tipo_elem	Tipo vía	M30 URB
distrito	Identificador del distrito de Madrid donde se encuentra el medidor	[1, 21]
id	Identificador único y permanente del punto de medida	4571
cod_cent	Código de centralización en los sistemas	[1001, 98416] [PM10013, PM43223]
nombre	Denominación del punto de medida. Para los puntos de medida de tráfico urbano se identifica con la calle. Para los puntos de vías rápida y accesos a Madrid se identifica con el punto kilométrico, la calzada y si se trata de la vía central, vía de servicio o un enlace	4227
utm_x	Coordenada X_UTM del centroide de la representación del polígono del punto de medida	[429 055.9, 450 772.3]
utm_y	Coordenada Y_UTM del centroide de la representación del polígono del punto de medida	[4 464 902, 4 485 213]
longitud	Proporciona la localización del punto de medida, en dirección Este u Oeste desde el meridiano de Greenwich	[-3.836943, -3.580713]
latitud	Proporciona la localización del punto de medida, en dirección Norte o Sur desde el ecuador	[40.33245, 40.51561]

Table 5: pmed_trafico_MM_YYYY

3) Distritos

En este archivo se recogen los datos de los distintos distritos de Madrid, contiene información sobre sus coordenadas, tamaño...

Variable	Descripción	Valores
cod_dis	Identificador único del distrito	[1, 21]
cod_dis_tx	Identificador único del distrito (...)	[01, 21]
nombre	Nombre del distrito	1. Centro 2. Arganzuela 3. Retiro 4. Salamanca 5. Chamartín 6. Tetuán 7. Chamberí 8. Fuencarral - El Pardo 9. Moncloa - Aravaca 10. Latina 11. Carabanchel 12. Usera 13. Puente de Vallecas 14. Moratalaz 15. Ciudad Lineal 16. Hortaleza 17. Villaverde 18. Villa de Vallecas 19. Vicálvaro 20. San Blas - Canillejas 21. Barajas
distri_may	Nombre del distrito (formato mayúsculas)	1. CENTRO 2. ARGANZUELA 3. RETIRO 4. SALAMANCA 5. CHAMARTIN 6. TETUAN 7. CHAMBERI 8. FUENCARRAL - EL PARDO 9. MONCLOA - ARAVACA 10. LATINA 11. CARABANCHEL 12. USERA 13. PUENTE DE VALLECAS 14. MORATALAZ 15. CIUDAD LINEAL 16. HORTALEZA 17. VILLAVERDE 18. VILLA DE VALLECAS 19. VICALVARO 20. SAN BLAS - CANILLEJAS 21. BARAJAS
distri_mt	Nombre del distrito (formato mayúsculas y codificación UTF-8)	1. CENTRO 2. ARGANZUELA 3. RETIRO 4. SALAMANCA 5. CHAMARTÍN 6. TETUÁN 7. CHAMBERÍ 8. FUENCARRAL - EL PARDO 9. MONCLOA - ARAVACA 10. LATINA 11. CARABANCHEL 12. USERA 13. PUENTE DE VALLECAS 14. MORATALAZ 15. CIUDAD LINEAL 16. HORTALEZA 17. VILLAVERDE 18. VILLA DE VALLECAS 19. VICÁLVARO 20. SAN BLAS - CANILLEJAS 21. BARAJAS
area_m2	m^2 de la superficie del distrito	[4 679 185 , 237 838 370]

Table 6: Distritos

0.4.1.3 Exploración del conjunto de datos

La etapa de exploración de los datos es necesaria y en ella se realizan ciertas comprobaciones que nos permiten verificar la consistencia de dichos datos. A menudo estos datos deben ser transformados para facilitar su futuro tratamiento. A continuación indicamos archivo por archivo dichas comprobaciones:

1) Archivos Trafico_MM_YYYY

- Si tenemos valores de *intensidad* = 0 en la M30, la *vmed* debería ser = 0 ya que no hay flujo de vehículos y por tanto no puede haber *vmed*.
- Para un mismo punto de medida, si observamos dos registros de *intensidad* iguales, deberíamos obtener *ocupaciones* prácticamente idénticas, ya que el comportamiento es casi idéntico. Efectivamente los datos se comportaron de la forma prevista.
- Para valores de *intensidad* = 0, la *ocupación* debería ser = 0, ya que no hay flujo de vehículos. En este caso hemos hallado incongruencias que posteriormente estudiaremos.
- Con *intensidad* > 0, esperamos *ocupación* > 0, ya que si entran vehículos, mínimo tienen que ocupar un periodo de tiempo las espiras. Al realizar esta comprobación detectamos ciertas anomalías que se estudiarán posteriormente.
- Con *intensidad* > 0 y *ocupacion* < 30 (valores bajos de ocupación, para evitar los casos de atascos en los que obviamente *vmed* tiende a 0), esperamos *vmed* > 0 en la M30, ya que los vehículos deben pasar por las espiras a cierta velocidad, por pequeña que sea. Hallamos casos que se comportaban de manera inesperada los cuales serán objeto de análisis.
- En la M30, cuando la *intensidad* = 0 y la *vmed* = 0, nos fijamos en la *ocupacion* puesto que debería ser elevada reflejando que existe un atasco, ya que hay flujo de vehículos pero están parados. Hay ciertos casos que no cumplen esta hipótesis y reflejan valores de *ocupacion* muy bajos, por lo que los estudiaremos posteriormente.
- La variable *error* indica si se ha detectado un fallo en la medición, hemos eliminado todas las entradas con estas características.

2) Archivos pmed_trafico_MM_YYYY

- La variable *cod_cent*, que se define como *código de centralización en los sistemas*, presenta una nomenclatura diferente para los puntos de medida interurbanos, formado por dos letras y 5 dígitos "PMXYYYZ":

- Dígito 'X', corresponde al tipo de calzada con un rango de valores del 1 al 5, pudiendo ser *1 (calzada interior)*, *2 (calzada exterior)*, *3 (carretera de salida de Madrid)* y *4 (carretera de entrada de Madrid)*
- Dígitos 'Y', si se encuentra en la M30, los dos primeros dígitos es el kilómetro y el tercer dígito el hectómetro, si se encuentra en una carretera de entrada o salida, los dos primeros dígitos son el kilómetro del enlace de la carretera con la M30 y el tercer dígito es el número ordinal que indica la proximidad a la M30.
- Dígito 'Z', corresponde a la situación en la calzada con un rango de valores del 1 al 8, pudiendo ser *1 (tronco)*, *2 (Vía de servicio (o lateral))*, *3 (Transfer de vía de servicio a tronco)*, *4 (Transfer de tronco a vía de servicio)*, *5 (Acceso al tronco)*, *6 (Acceso a la vía de servicio)*, *7 (Salida del tronco)*, *8 (Salida de la vía de servicio).*

0.4.1.4 Calidad de los datos

- **Calidad a priori:** Tras realizar el estudio preliminar, podemos afirmar que contamos con un gran volumen de datos, los cuales tendremos que limpiar exhaustivamente puesto que hay numerosas incongruencias pero que afectan a muy pocos datos, por lo que tras la limpieza seguiremos contando con un gran volumen de datos para nuestro estudio.
- **Calidad a posteriori:** Tras la realización de todo el estudio, y viéndolo con perspectiva, la calidad de los datos efectivamente es buena puesto que se han podido desarrollar ciertos modelos con unos porcentajes de acierto alto, pero, la cantidad de datos seleccionado ha sido excesiva ya que hemos tenido muchos problemas a la hora de trabajar con ellos por temas de memoria y velocidad de procesamiento.

0.4.1.5 Problemas en la fase de preparación de los datos

Durante el estudio y preparación de los datos, nos encontramos con ciertos obstáculos, tanto de comprensión del comportamiento u obtención de algunas de las variables, como de interpretación de resultados en los que hallamos anomalías que dificultaban la comprensión de los resultados, ya que se presentaban casos opuestos que contaminaban el estudio. Al no tratarse de casos aislados decidimos ponernos en contacto con los responsables del Portal de Datos del Ayuntamiento de Madrid[1], realizando un vídeo, acompañado de una presentación PowerPoint, en el que exponíamos nuestras dudas, tanto las variables que no terminábamos de comprender, como los casos anómalos y nuestra interpretación del posible error.

En su respuesta nos explicaron con detalle tanto las variables que no comprendíamos como el porqué de las situaciones anómalas. Esto nos permitió continuar con nuestro estudio y limpiar y procesar los datos de forma más precisa.

Nuestras principales dudas fueron:

- Para tramos con *intensidad* igual a 0 y *ocupacion* elevada (60% o superior), no entendíamos porqué las variables *ocupacion* y *carga* eran iguales (si la *ocupacion* es de un punto en concreto y la *carga* es de toda la vía), y porqué, para estos casos, la variable *vmed* tenía valores NA, y si esos valores afectaban al cálculo del resto de variables.

  Nos explicaron que esas situaciones pueden darse en atascos o semáforos cuando el censo inicia el minuto ocupado y, poco antes de terminar el minuto, el vehículo que lo estaba activando se ha ido, sin llegar a ser ocupado por el siguiente, de forma que no hay ningún tránsito completo y no se puede calcular la *vmed*.

- Para tramos con *intensidad* igual a 0 y *ocupacion* baja (30% o inferior), no sabíamos si esto se debía a que la *ocupacion* fuese de vehículos que ya se encontraban en el tramo, y para tramos con *intensidad* mayor que 0 y *ocupacion* y *vmed* igual a 0, quisimos saber porqué la *ocupacion* se comportaba igual tanto para valores altos como bajos.

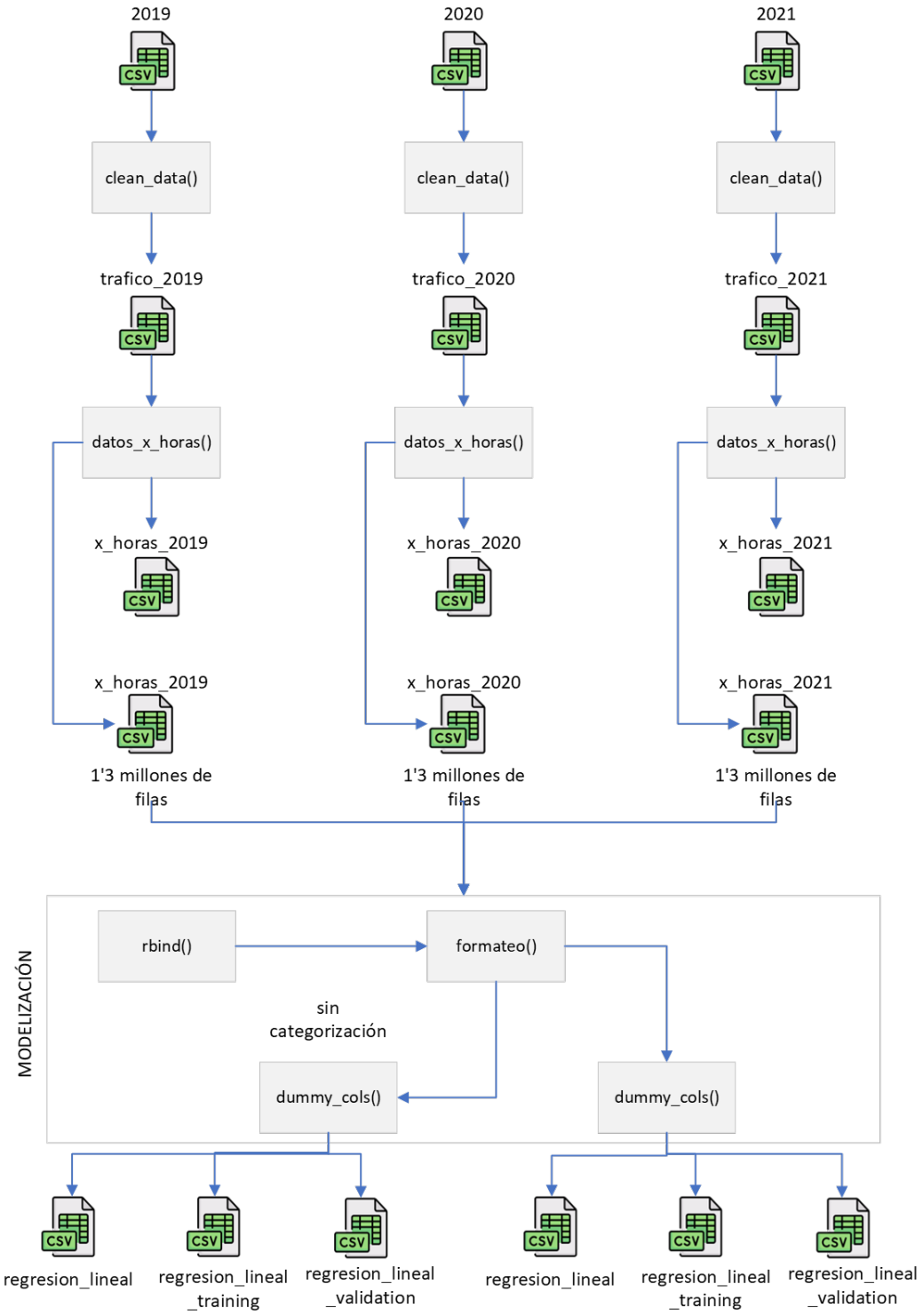
  Nos explicaron que, en ambos casos, se debe a un mal funcionamiento del sistema, ya que, por ejemplo, el periodo de ocupación de cada sensor a una velocidad de unos 70 Km/h es de unas 250 milésimas de segundo, y esto hace que el sistema no tenga una fiabilidad del 100%, dejando un margen de error del 5%.

- Para tramos con *intensidad* alta y *vmed* igual a 0, observamos que había casos en los que la *ocupacion* era muy baja, y quisimos saber si se debía a un error de la variable *vmed*, ya que la interpretación que hicimos de *ocupacion* e *intensidad* no sugería que hubiese retenciones, y si, como comentamos antes, ese error estaba afectando al cálculo del resto de variables. Nos explicaron que, para tramos *urbanos*, los sensores son dobles, para poder calcular la velocidad y el tamaño de los vehículos, y cuando un vehículo pasa solo por uno de los dos sensores o uno de ellos no funciona, únicamente se pueden contabilizar los tránsitos y la ocupación del sensor que se ha activado, pero no es posible calcular ni la velocidad ni el tamaño.

0.4.2 Tratamiento de los datos

Al tratarse de ficheros de gran volumen de datos, hemos utilizado la librería de R *data.table*, para mejorar el tiempo de ejecución y facilitar la manipulación y exploración de los datos, concretamente hemos usado las funciones *fread* y *fwrite* para optimizar la lectura y escritura de estos.

Para poder empezar a trabajar con los datos lo primero es detectar aquellos que nos van a ser de utilidad y prescindir de los que no aportan ninguna información. Para ello hemos realizado una serie de procesos que se explicarán a continuación.



Fuente: Iconos csv creados por Freepik - Flaticon

Figure 11: Diagrama de flujo de nuestros datos

0.4.2.1 Selección de datos y variables

Puesto que se trata de datos históricos, el primer paso que hemos realizado es guardar toda la información que teníamos en nuestros datasets. Sin embargo, hemos tenido que tratar los archivos de partida y eliminar algunas columnas, ya que al unir algunos archivos implicaría redundancia de datos.

0.4.2.2 Dar formato a los datos

- En nuestros archivos de tráfico tenemos registros de dos tipos de vía: Interurbano (M30) y Urbano (URB), las cuales tienen ciertas diferencias, como por ejemplo que los tramos URB no tienen valores para la velocidad media. Por lo que hemos separado esos datos en dos archivos diferentes de acuerdo a ese criterio.

  Además, tras un estudio más intensivo hemos decidido agrupar nuestros datos en periodos de 1 hora (originalmente todas las mediciones venían en períodos de 15 minutos). De esta forma hemos pasado a trabajar con 1/4 del volumen de datos, agilizando mucho los periodos de ejecución de nuestros algoritmos. Además, esto no afecta a la calidad de los datos ya que mediciones cada 15 minutos son excesivamente precisas para trabajar con datos de tráfico en un periodo de 3 años.

  Para realizar este cambio hemos tenido que juntar las mediciones agrupándolas por el *id* (identificador de punto de medición) y la *fecha* (la cual pusimos en formato YYYY/MM/DD/HH eliminando minutos y segundos para poder agrupar en periodos de 1 hora). Además, variables como son la *carga*, la *vmed* o la *ocupación* han sido recalculadas aplicando un promedio de los 4 periodos de 15 minutos.

  Posteriormente, tras la agrupación por horas, hemos creado 4 nuevas variables: año, mes, día y hora a partir de la variable *fecha*, de esta forma podemos realizar agrupaciones más sencillas basándonos en estos criterios para la realización de gráficas para nuestro estudio.

  También hemos recodificado la variable *día*, la cual originalmente indicaba el día del mes (valores desde 1 hasta máximo 31). Hemos aplicado la función *weekday()* de tal forma que ahora la variable *día* representa el día de la semana que es (Lunes-Domingo). Este cambio se debe a que nos interesa mucho más esta codificación para el enfoque que queremos darle a los datos, ya que por ejemplo resulta muy interesante observar tendencias o patrones en los días de la semana, pero no tanto en los días del mes porque es más trivial.

  Otro de los principales cambios que hemos realizado en este dataset es la aplicación de *One Hot Encoding* sobre las variables `año, mes, día` y `hora`. La estrategia que implementa esta codificación es crear una columna para cada valor distinto que exista en la variable que estamos codificando, y marcar con un 1 la columna a la que pertenezca dicho registro y dejar las demás con 0. Por ejemplo, en vez de tener una variable *mes* que va puede tomar los valores desde 1 a 12, creamos

12 columnas de las cuales solo la columna a la que pertenezca esta medición se marcara con un 1 y el resto de las once columnas se marcaran con un 0.

La aplicación de *One Hot Encoding* es muy importante e imprescindible para que el entrenamiento de nuestros modelos funcione correctamente, ya que si entrenásemos al modelo con un atributo mes que contiene del 1 al 12, el algoritmo daría más peso ciertos meses frente a otros por el mero hecho de que hay una jerarquía numérica y el algoritmo no entiende que es una codificación y que esos valores no implica que un mes sea más o menos relevante que otro.

Tras todos los cambios tenemos 24 columnas (una para cada hora del día), 7 columnas para los días, 12 para los meses, 3 para los años, 3 para el tipo de vía (M30, URB y other), 5 para la fase de la pandemia (prepandemia, fase 0,1,2 y postpandemia), 4 para la carga agrupada en rangos, además de la ocupación, intensidad, el id y la velocidad media. Por lo que contamos con un total de 62 variables para nuestros modelos.

0.4.2.3 Construcción de nuevas variables

Al aplicar la codificación One Hot hemos creado un gran número de nuevas columnas, pero realmente son derivadas de las columnas originales, además de por ejemplo decodificar la fecha en varias columas con la hora, mes, día... En cuanto a creación de nuevas variables desde cero, recalcar la creación de la variable *fase* la cual basándonos en la fecha del registro en cuestión y comparándola con las fases de la pandemia en la Comunidad de Madrid le hemos otorgado un valor u otro (prepandemia, fase 0, fase 1, fase 2, postpandemia)

0.4.3 Análisis descriptivo de las variables

En este punto analizamos las variables más relevantes para nuestro estudio. Para hacer un análisis más preciso de los datos, en este punto dividimos los archivos en dos, uno con datos interurbanos y otro con datos urbanos. Todos los gráficos de este punto han sido generados con lenguaje R en RStudio, haciendo uso de la librería *ggplot2*[6].

En el análisis de la variable *intensidad*, en la Figura 12, se puede observar que el comienzo de la pandemia en el año 2020 supuso un descenso en la media de la intensidad por tramo a lo largo del año, bajando un 28% en los tramos urbanos, representados de color azul, y un 24% en los tramos interurbanos, representados de color rojo. Aunque si analizamos más en detalle, como podemos ver en la Figura 13, el mes de abril de 2020, correspondiente al *Estado de alarma*, supuso una caída de más del 70% de la media de intensidad por tramo a lo largo del mes de abril, superando con diferencia a la caída de porcentaje a nivel anual.

La información correspondiente al *Estado de alarma* la obtuvimos del *BOE* del Sábado 14 de marzo de 2020.

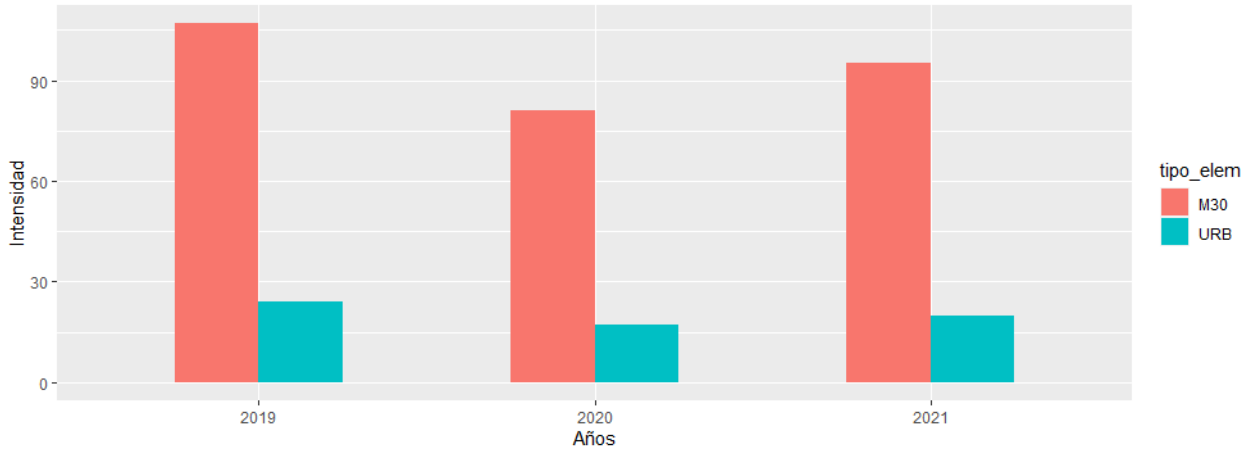


Figure 12: Intensidad media por tramo en un año

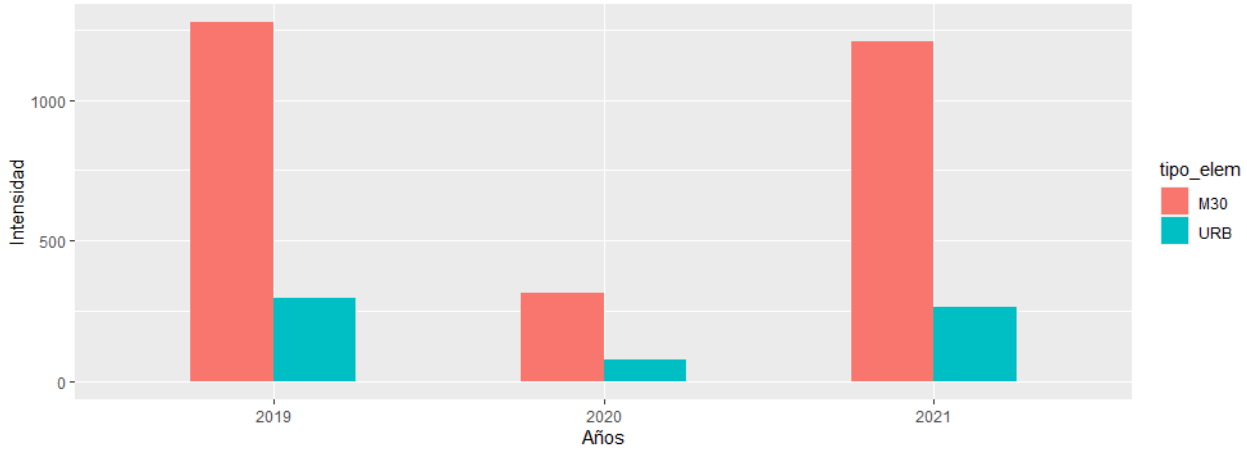


Figure 13: Intensidad media por tramo en el mes de abril

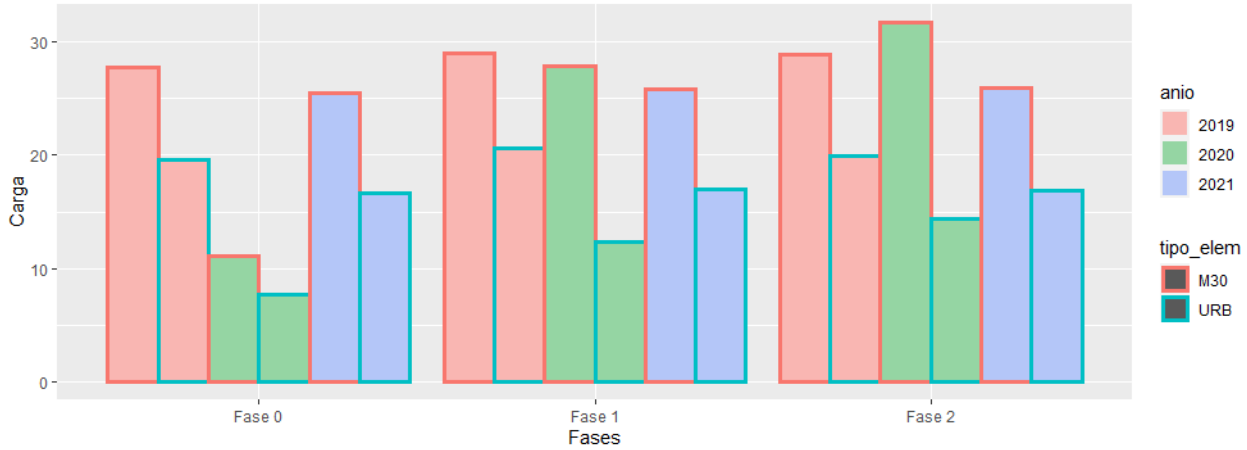


Figure 14: Carga media por tramo en una fase

Para el análisis de la variable *carga*, hemos tomado los datos de los meses correspondientes a las *fases de confinamiento*, que son:

- *Fase 0*: 3 de mayo – 24 de mayo
- *Fase 1*: 25 de mayo – 7 de junio
- *Fase 2*: 8 de junio – 21 de junio

Para poder estudiar la variación de la *carga*, hemos analizado el mismo periodo en los demás años, representando de color rojo el año 2019, de color verde el año 2020 y de color azul el 2021.

En el año 2020, se puede observar, en la Figura 14, un descenso significativo de la media de la *carga* por tramo a lo largo de la *Fase 0*, motivado por las duras medidas preventivas aplicadas en esa fase que limitaban la libertad de circulación, bajando la media de *carga* por tramo en más de un 60% tanto en tramos urbanos como interurbanos.

En cuanto a la *Fase 1* y *Fase 2* de este mismo año, se flexibilizaron las medidas de libertad de circulación, y puede verse reflejado con el incremento de la media de la *carga* por tramo a lo largo de estas dos fases.

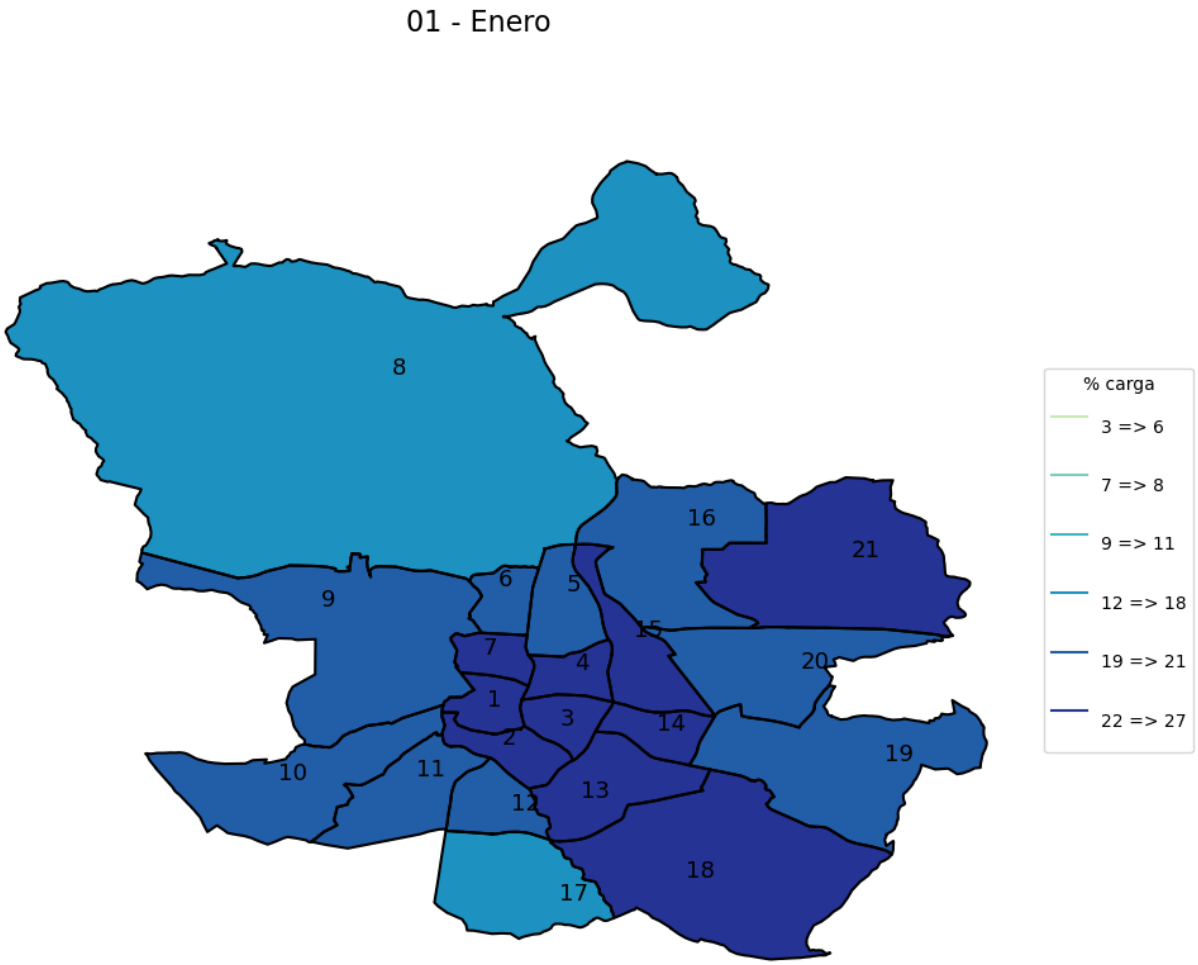


Figure 15: Carga media para cada mes y por distrito del año 2020

Realizamos un mapa para cada mes del año 2020 en forma de GIF, en el que representamos la *carga* media con datos urbanos para cada uno de los distritos de Madrid. Se puede observar en la Figura 15 que durante enero y febrero el nivel de *carga* media predominante supera el 19%, disminuye en el periodo de confinamiento (14/03/2020 - 02/05/2020) donde podemos ver que en el mes de abril la *carga* media predominante no supera el 9%, y a partir de junio empieza a recuperar los niveles pre-confinamiento, disminuyendo ligeramente en el mes de agosto debido a las vacaciones de verano.

La información correspondiente a las *Fases de desescalada* la obtuvimos del *BOE* de las siguientes fechas: Domingo 3 de mayo de 2020 y Sábado 9 de mayo de 2020.

Para analizar la variable *ocupacion* hemos analizado los años 2019 y 2021 para realizar una comparativa entre la antigua y la nueva normalidad, representado de color rojo los tramos interurbanos y de color azul los tramos urbanos.

Tanto en la Figura 16 como en la Figura 17 se puede observar que los meses de verano

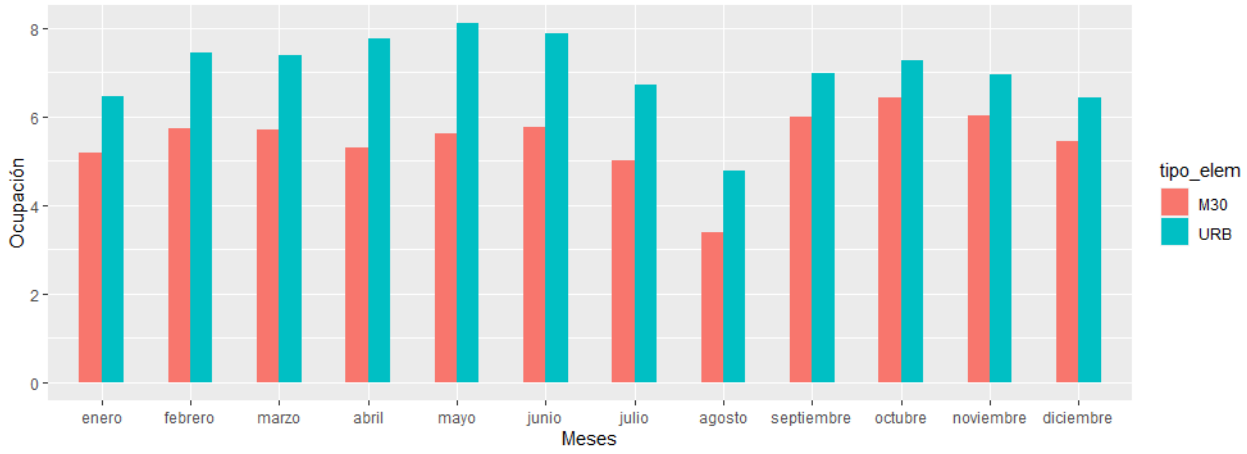


Figure 16: Ocupación media por tramo en el año 2019

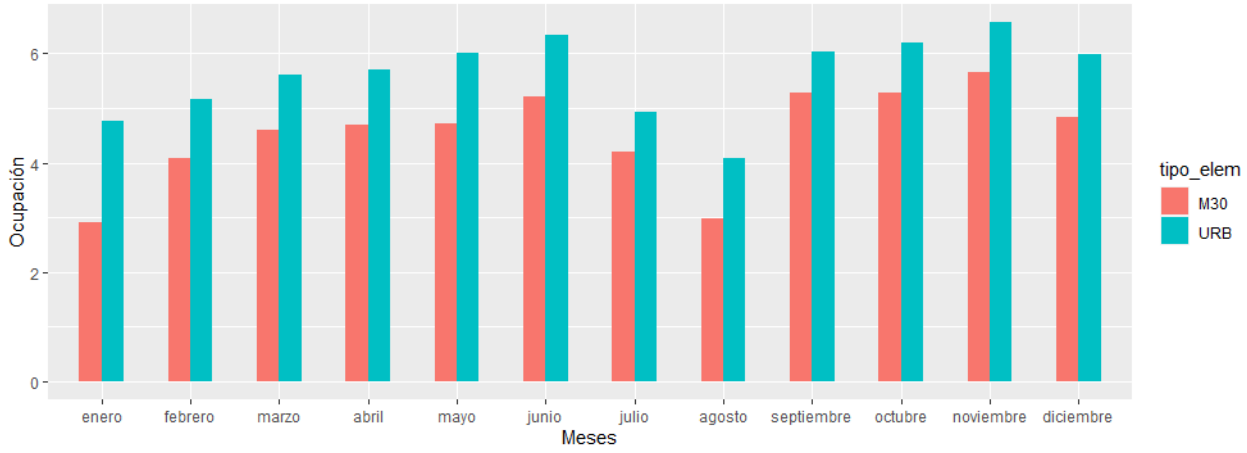


Figure 17: Ocupación media por tramo en el año 2021

presentan un menor porcentaje medio de ocupación por tramo frente a los meses de primavera, que presentan uno mayor. También se puede apreciar que el porcentaje medio de ocupación es siempre mayor en los tramos urbanos, debido quizás a las limitaciones de velocidad y a la afluencia de peatones.

En la Figura 17 se puede observar un descenso general del porcentaje medio de ocupación por tramo respecto a la Figura 16, descendiendo una media del 17% en tramos interurbanos, y del 18% en tramos urbanos. Que en la nueva normalidad no se hayan recuperado los niveles previos a la pandemia, quizás pueda ser motivado por la implantación del teletrabajo, además de las posibles obras de larga duración o el número de días lluviosos del mes.

Para hacer un estudio más detallado, en la Figura 18 y 19 se representa la ocupación media a lo largo de las horas del día, de lunes a viernes en el mes de octubre de ambos años. En este caso nos fijamos en las horas puntas, de *06:00 a 10:00* correspondería con el inicio de la jornada laboral, de *13:00 a 15:00* correspondería con el periodo de

Figure 18: Ocupación media por horas en el mes de octubre en tramos interurbanos (Años 2019 y 2021)

Figure 19: Ocupación media por horas en el mes de octubre en tramos urbanos (Años 2019 y 2021)

la comida, y de *17:00 a 20:00* correspondería con el fin de la jornada laboral.

Si nos centramos en el año 2021, en ambos gráficos se puede observar una caída del porcentaje de ocupación con respecto a 2019, tanto en la primera franja de *06:00 a 10:00* como en la última de *17:00 a 20:00*, por lo que reforzamos la posibilidad de que esa caída se deba a que hay menos desplazamientos a la oficina y más teletrabajo, en cuanto a la franja de *13:00 a 15:00*, la ocupación media reduce su diferencia con las otras dos franjas en comparación con 2019, por lo que se podría interpretar que el porcentaje de gente que va a trabajar y luego se desplaza para comer es mayor.

También se puede apreciar en los tramos interurbanos (Figura 18) del año 2021, la franja de *17:00 a 20:00* presenta menor ocupación media que la franja de *13:00 a 15:00*, esto podría deberse a que la gente que se desplaza a la hora de comer por tramos interurbanos, es decir, fuera del núcleo de la ciudad, no regrese después a la oficina y

haga teletrabajo por las tardes.

0.5 Integración de R con Python

Para integrar Python en R, a la hora de realizar el mapa de calor, nos documentamos de las posibles opciones, finalmente nos decidimos por la librería *reticulate* ya que ofrece muchas formas de combinar ambos lenguajes, permitiéndonos aprovechar lo mejor de ambos según las circunstancias.

En nuestro caso nos decantamos por desarrollar todo el código Python en un script **.py** y hacer una llamada desde el script **.R** a ese script **.py** para ejecutarlo. Puntualmente realizamos llamadas al script **.R** desde el script **.py** para extraer información de tablas, evitando así tener que guardarlas en un **.csv**.

Para el desarrollo del mapa utilizamos las siguientes librerías:

- *shapefile*: Para la lectura y tratamiento de datos de archivos shapefile.
- *pandas*: Lectura y tratamiento de archivos (en nuestro caso **.csv**).
- *matplotlib*: Para la generación de gráficos a partir de datos contenidos en listas o arrays.
- *numpy*: Para realizar cálculos avanzados.
- *os*: Acceso a funcionalidades dependientes del Sistema Operativo. Información sobre el entorno del mismo y manipulación de la estructura de directorios.
- *imageio*: Para leer y escribir una amplia gama de datos de imágenes, incluidas imágenes animadas, datos volumétricos y formatos científicos.

La generación de nuestro mapa consta de tres partes:

1. Preparación de los datos.
    - Utilizamos un fichero *.shp* que contiene la información de los distritos, tanto sus nombres como sus coordenadas, y un *dataframe* que contiene el valor medio de carga por mes para cada distrito.
    - Modificamos el fichero *.shp* (distritos) de forma que su columna *COD_DIS* (identificador único del distrito) fuera de tipo *int* para poder ordenarla por este factor.
    - El *dataframe* (*carga_x_distrito*) lo obtuvimos con una llamada al script *.R* en el cual habíamos procesado y estructurado los datos para crear este *dataframe*, que recogía para cada mes, la *carga* media por distrito.

- Para preparar la paleta de colores que usamos en el mapa de calor, guardamos en un *array* todos los máximos y mínimos de la *carga* media de cada mes (para posteriormente hacer una leyenda con valores absolutos para todos los meses), y en otro *array* guardamos los códigos de color de una paleta, y finalmente ejecutamos la función de la librería *pandas qcut()*, de la que obtenemos un *array* **'bins'** con los máximos y mínimos divididos en seis rangos indicando tan solo el máximo y mínimo de cada rango.

2. Creación del mapa.

   Para generar nuestro mapa, ejecutamos tres funciones por mes.

   - La primera, una función en la que asignamos un código de color en función del valor de *carga* del distrito, haciendo uso del *array* **'bins'**, y guardamos esta información en el *array* **tonalidad**.
   - La segunda, una función que, en primer lugar, utiliza las coordenadas del archivo *.shp* para generar la silueta de los distritos, y en segundo lugar, recorre un *array* con el *id* de cada distrito con el que se consulta, en el fichero *.shp* que coordenadas tiene, y en el *array* **tonalidad** el color que le corresponde, y con esto rellena la información de ese distrito haciendo uso de la función *fill()* de la librería *matplotlib*. Una vez finalizado este bucle guardamos el mapa en local.
   - La tercera, la función *show()* de la librería *matplotlib*, que muestra el mapa por pantalla.

3. Creación del GIF.

   - Para generar el GIF, utilizamos la librería *Imageio*. Recorremos el directorio con los archivos *.png* creados para cada mapa de cada mes y guardamos en un *array* el nombre de la imagen leído con la función *imread()*. Por último guardamos en local el GIF con la función *mimwrite()*.

Además, el modelo de regresión logística multi-clase también lo hemos desarrollado empleando el lenguaje Python, a través del entorno PyCharm para trabajar de una forma más cómoda y potente. En este caso, no conectamos Pyhton a R a través de librerías sino que desde Phython realizamos una lectura de los archivos .csv que habíamos guardado tras el proceso de limpieza. Los motivos de que nos han llevado a emplear Phyton es que ya teníamos cierta familiaridad empleando este tipo de algoritmos y además nos parecía interesante trabajar y familiarizarnos con los dos lenguajes más potentes para DataScience, para, de cara al futuro, tener más perspectiva y experiencia con ambos lenguajes.

0.6 Modelado

El objetivo de este modelado es poder identificar que factores son los más importantes a la hora de predecir la carga que tendrá una carretera en Madrid en cierto momento.

Para el entrenamiento de nuestros modelos hemos reducido el volumen de datos con el que contábamos, ya que a pesar de haber agrupado los registros por horas, y tras todo el proceso de limpieza, el volumen de datos era poco práctico para la realización de los modelos, ya que colapsaban por problemas de falta de memoria y de velocidad de procesamiento. Para disminuir el volumen de datos, hemos aplicado una función pseudo-aleatoria (antes de agrupar los datos en una sola estructura), de forma que nos hemos quedado con el mismo porcentaje de datos de cada uno de los meses para de esta forma mantener el balance de los datos. Tras este proceso, finalmente contamos con un dataset final *datos_modelado_completo.csv* con 1,032,260 de filas y 11 columnas (de las cuales 10 serán empleadas como predictores, descartando el id ya que no aporta información relevante). En la figura 20 se puede ver una representación del esquema de este dataset en R.

	id	hora	intensidad	vmed	carga	ocupacion	tipo_elem	mes	dia_semana	anio	fase
1	3544	00	284	71	0	0	M30	enero	martes	2019	prepandemia
2	3560	00	407	67	8	1	M30	enero	martes	2019	prepandemia
3	3585	00	26	0	7	0	URB	enero	martes	2019	prepandemia
4	3704	00	539	48	10	3	M30	enero	martes	2019	prepandemia
5	3722	00	74	0	7	3	URB	enero	martes	2019	prepandemia
6	3929	00	9	0	5	0	URB	enero	martes	2019	prepandemia
7	3949	00	373	0	8	2	URB	enero	martes	2019	prepandemia
8	4042	00	105	0	5	0	URB	enero	martes	2019	prepandemia
9	4047	00	44	0	6	0	URB	enero	martes	2019	prepandemia
10	4111	00	51	0	6	1	URB	enero	martes	2019	prepandemia
11	4295	00	78	0	11	2	URB	enero	martes	2019	prepandemia
12	4415	00	133	0	4	0	URB	enero	martes	2019	prepandemia
13	4776	00	133	0	11	1	URB	enero	martes	2019	prepandemia
14	4818	00	42	0	3	0	URB	enero	martes	2019	prepandemia
15	4845	00	25	0	5	3	URB	enero	martes	2019	prepandemia
16	4867	00	17	0	7	5	URB	enero	martes	2019	prepandemia
17	4873	00	74	0	2	0	URB	enero	martes	2019	prepandemia
18	5106	00	45	0	4	0	URB	enero	martes	2019	prepandemia
19	5196	00	195	0	8	1	URB	enero	martes	2019	prepandemia
20	5505	00	97	0	4	1	URB	enero	martes	2019	prepandemia
21	5664	00	45	0	1	0	URB	enero	martes	2019	prepandemia
22	5852	00	289	0	40	3	URB	enero	martes	2019	prepandemia
23	6281	00	38	0	13	8	URB	enero	martes	2019	prepandemia
24	6326	00	16	0	2	0	URB	enero	martes	2019	prepandemia
25	6341	00	119	0	6	0	URB	enero	martes	2019	prepandemia
26	6754	00	272	59	8	1	M30	enero	martes	2019	prepandemia
27	6817	00	88	42	8	1	M30	enero	martes	2019	prepandemia
28	6964	00	29	0	4	1	URB	enero	martes	2019	prepandemia
29	7088	00	62	0	4	0	URB	enero	martes	2019	prepandemia
30	7126	00	144	61	15	0	M30	enero	martes	2019	prepandemia
31	9981	00	129	0	6	1	URB	enero	martes	2019	prepandemia
32	10068	00	56	0	4	0	URB	enero	martes	2019	prepandemia
33	10084	00	21	0	3	0	URB	enero	martes	2019	prepandemia
34	10091	00	67	0	2	0	URB	enero	martes	2019	prepandemia
35	10114	00	24	0	6	0	URB	enero	martes	2019	prepandemia
36	10203	00	569	68	16	1	M30	enero	martes	2019	prepandemia
37	10337	00	8	0	0	0	URB	enero	martes	2019	prepandemia
38	3445	01	31	0	3	5	URB	enero	martes	2019	prepandemia

Showing 1 to 39 of 1,031,160 entries, 11 total columns

Figure 20: Estructura del dataset para nuestros modelos

Dado que los modelos hay que entrenarlos con unos datos distintos de los que luego usamos para validar (porque sino hay sesgo), el primer paso que hemos llevado a cabo es aplicar la función sample() con la finalidad de crear dos subconjuntos distintos. La hemos aplicado de la siguiente forma: **sample (2, nrow (dataset), replace = TRUE, prob = c (0.7,0.3))**. Esta función nos permite asociar un identificador (en

nuestro caso 1 o 2) a cada una de las filas del dataset de acuerdo a la probabilidad que hemos escogido en cada caso, de esta forma, cada fila tiene un 0.7 de ser asociada con un 1 y un 0.3 de asociarse al valor 2. Posteriormente, basándonos en esos identificadores, dividimos en dos subconjuntos de acuerdo al identificador. Finalmente, nuestro dataset de entrenamiento *datos_modelado_training.csv* cuenta con un 70% de los datos y el dataset de validación el 30% restante. El motivo de elegir estas proporciones es que es más apropiado contar con un conjunto de entrenamiento más grande que el de validación, puesto que el conjunto de entrenamiento es el que el algoritmo emplea para aprender a estimar la variable dependiente.

0.6.1 1ª Fase del modelado - Regresión lineal múltiple en R

Como primera fase del modelado hemos decidido emplear un algoritmo de regresión lineal para ver como se comportaba con nuestros datos y que información nos podía aportar. Para ello, empleamos la función *lm()* de la librería *stats*, la cual crea un modelo basándose en regresión lineal simple mediante la siguiente formula **y = x1 + x2 ... + xn**. Donde *y* representa la variable dependiente y *x1 ... xn* representan los predictores que queremos emplear para generar el modelo.

En esta iteración hemos empleado el subconjunto de training *datos_modelado_training.csv* para entrenar el modelo mediante la siguiente formula: **iteracion1 = lm(carga = ocupacion + intensidad + hora + vmed + tipo_elem + mes + dia_semana + anio + fase, data = training)**, de forma que hemos empleado todos los predictores disponibles para ver como se comportaba cada uno de ellos.

Tras generar el modelo, empleamos la función *summary()*, la cual nos aporta diversa información que nos va a permitir estudiar el modelo generado. En la figura 21 podemos observar la información obtenida.

```
> summary(iteracion1)

Call:
lm(formula = carga ~ ocupacion + intensidad + hora + vmed + tipo_elem +
    mes + dia_semana + anio + fase, data = training)

Residuals:
     Min       1Q   Median       3Q      Max
-132.372   -4.250   -1.186    3.332   90.056

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)             -2.991e+02  6.315e+01  -4.737 2.17e-06 ***
ocupacion                8.326e-01  1.334e-03 624.262  < 2e-16 ***
intensidad               1.280e-02  2.467e-05 518.607  < 2e-16 ***
hora01                  -1.211e+00  7.928e-02 -15.271  < 2e-16 ***
hora02                  -1.961e+00  7.940e-02 -24.695  < 2e-16 ***
hora03                  -2.284e+00  7.974e-02 -28.641  < 2e-16 ***
hora04                  -2.409e+00  7.971e-02 -30.218  < 2e-16 ***
hora05                  -2.288e+00  7.967e-02 -28.713  < 2e-16 ***
hora06                  -7.808e-01  7.926e-02  -9.851  < 2e-16 ***
hora07                   2.491e+00  7.947e-02  31.342  < 2e-16 ***
hora08                   5.642e+00  7.977e-02  70.726  < 2e-16 ***
hora09                   6.524e+00  7.956e-02  81.995  < 2e-16 ***
hora10                   6.088e+00  7.945e-02  76.637  < 2e-16 ***
hora11                   6.722e+00  7.957e-02  84.483  < 2e-16 ***
hora12                   7.542e+00  7.965e-02  94.683  < 2e-16 ***
hora13                   7.950e+00  7.976e-02  99.686  < 2e-16 ***
hora14                   8.654e+00  8.004e-02 108.121  < 2e-16 ***
hora15                   7.311e+00  7.972e-02  91.704  < 2e-16 ***
hora16                   6.377e+00  7.954e-02  80.171  < 2e-16 ***
hora17                   7.327e+00  7.975e-02  91.871  < 2e-16 ***
hora18                   7.727e+00  7.988e-02  96.742  < 2e-16 ***
hora19                   7.813e+00  7.986e-02  97.838  < 2e-16 ***
hora20                   7.401e+00  7.983e-02  92.711  < 2e-16 ***
hora21                   5.795e+00  7.933e-02  73.048  < 2e-16 ***
hora22                   3.338e+00  7.914e-02  42.184  < 2e-16 ***
hora23                   1.330e+00  7.911e-02  16.808  < 2e-16 ***
vmed                     1.359e-01  1.615e-03  84.137  < 2e-16 ***
tipo_elemother           1.012e+01  2.235e+00   4.530 5.90e-06 ***
tipo_elemURB             1.080e+01  1.055e-01 102.400  < 2e-16 ***
mesagosto               -1.651e+00  5.983e-02 -27.595  < 2e-16 ***
mesdiciembre             1.745e-01  5.938e-02   2.939 0.003296 **
mesenero                -3.428e-01  5.991e-02  -5.722 1.05e-08 ***
mesfebrero               3.686e-01  6.074e-02   6.069 1.29e-09 ***
mesjulio                 1.468e-01  5.977e-02   2.456 0.014059 *
mesjunio                 7.202e-01  6.234e-02  11.553  < 2e-16 ***
mesmarzo                 2.207e-01  5.686e-02   3.882 0.000104 ***
mesmayo                  5.913e-01  5.687e-02  10.397  < 2e-16 ***
mesnoviembre             5.738e-01  5.977e-02   9.601  < 2e-16 ***
mesoctubre               4.627e-01  5.939e-02   7.790 6.69e-15 ***
messeptiembre            5.421e-01  5.991e-02   9.048  < 2e-16 ***
dia_semanajueves         2.830e+00  4.310e-02  65.667  < 2e-16 ***
dia_semanalunes          2.368e+00  4.311e-02  54.933  < 2e-16 ***
dia_semanamartes         2.619e+00  4.315e-02  60.686  < 2e-16 ***
dia_semanami<e9>rcoles   2.715e+00  4.309e-02  63.010  < 2e-16 ***
dia_semanas<e1>bado      8.449e-01  4.294e-02  19.675  < 2e-16 ***
dia_semanaviernes        2.782e+00  4.311e-02  64.535  < 2e-16 ***
anio                     1.411e-01  3.126e-02   4.513 6.40e-06 ***
fase1                    3.160e+00  1.172e-01  26.950  < 2e-16 ***
fase2                    4.210e+00  1.278e-01  32.940  < 2e-16 ***
fasepostpandemia         4.948e+00  6.117e-02  80.877  < 2e-16 ***
faseprepandemia          6.190e+00  6.038e-02 102.509  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.73 on 722601 degrees of freedom
Multiple R-squared:  0.6848,    Adjusted R-squared:  0.6848
F-statistic: 3.14e+04 on 50 and 722601 DF,  p-value: < 2.2e-16
```

Figure 21: Reporte de la 1ª iteración mediante regresión lineal simple

El valor *Adjusted R-squared* = 0.6848 señala que los predictores que hemos empleado en este modelo consiguen explicar un 68.4% del comportamiento de la *carga*, por lo que para un primer modelo, y empleando regresión lineal simple es un valor bastante significativo.

Además si observamos el *p-value*, podemos ver que la mayoría de nuestros predictores cuentan con valores muy bajos, indicando que tienen baja probabilidad de hacer que nuestra hipótesis falle. Es curioso observar como los registros de los meses de Julio y Diciembre tienen un poco más de tendencia al fallo, probablemente debido a que coincide con los meses de comienzo de periodo vacacional, donde es más difícil predecir los movimientos ya que interviene más el factor humano.

Otra de las métricas que observamos, es el *RSE (residual standard error)* con un valor de 9.73, lo cual implica que nuestra predicción de la carga se estima que tiene ese fallo medio en cada predicción. Dado que nuestra variable carga toma valores en el intervalo [0-100], un fallo de 9.73 no es significativo ya que tampoco nos interesa predecir el valor exacto de la carga, sino una estimación razonable.

El problema de las métricas anteriores es que dependen de los grados de libertad del modelo, por lo que solo nos sirven para comparar modelos en los que empleemos el mismo número de predictores y por ello tengamos los mismos grados de libertad. Dado que para encontrar la mejor combinación de predictores hemos ido creando diferentes iteraciones, y en cada una hemos trabajado con diferente número de predictores, hemos recurrido a métricas absolutas en lugar de a las previamente mencionadas de carácter relativo. De esta forma, podamos comparar todos nuestros modelos entre sí. Por lo que para comparar nuestras diversas iteraciones hemos empleado las siguientes métricas: *ECM* $\rightarrow$ *error cuadrático medio y el EMA* $\rightarrow$ *error medio absoluto.* Las métricas previamente nombradas están definidas en el punto 0.3.2 del presente trabajo.

Este primer análisis en R nos permitió identificar la calidad de cada uno de los predictores, así como ver la calidad de nuestros primeros modelos generados. Tras este paso inicial cambiamos a Python simplemente con la finalidad de experimentar con los dos lenguajes y no restringirnos solo a R.

Para el desarrollo de la regresión lineal en Phyton, nos servimos de la función *LinearRegression* de la librería *sklearn* además de la librería *numpy* para emplear funciones vectorizadas y minimizar los tiempos de ejecución. En la figura 22 se muestra el código desarrollado.

```
variable_validacion = Xvalidation_lineal
mval_lineal = np.shape(variable_validacion)[0]

#regresion
regressor = lm.LinearRegression()
regressor.fit(Xtrain_lineal, Ytrain_lineal)
prediccion_lineal = regressor.predict(variable_validacion)

en_rangos = prediccion_lineal
en_rangos2 = Yvalidation_lineal

#agrupar en rangos
for fila in range (len(prediccion_lineal)):

    if  ((en_rangos[fila] >= 0)  & (en_rangos[fila] < 25)):
        en_rangos[fila] = 0
    elif ((en_rangos[fila] >= 25)  & (en_rangos[fila] < 49)):
        en_rangos[fila] = 1
    elif ((en_rangos[fila] >= 50)  & (en_rangos[fila] < 74)):
        en_rangos[fila] = 2
    elif ((en_rangos[fila] >= 75)  | (en_rangos[fila] == 100)):
        en_rangos[fila] = 3

for fila2 in range (len(Yvalidation_lineal)):

    if ((en_rangos2[fila2] >= 0)  & (en_rangos2[fila2] < 25)):
        en_rangos2[fila2] = 0
    elif ((en_rangos2[fila2] >= 25)  & (en_rangos2[fila2] < 49)):
        en_rangos2[fila2] = 1
    elif ((en_rangos2[fila2] >= 50)  & (en_rangos2[fila2] < 74)):
        en_rangos2[fila2] = 2
    elif ((en_rangos2[fila2] >= 75)  | (en_rangos2[fila2] == 100)):
        en_rangos2[fila2] = 3

#calcular % acierto
aciertos2 = np.sum(en_rangos == en_rangos2)
porcentaje2 = (aciertos2 / mval_lineal) * 100

print ("Porcentaje de acierto regresión lineal: ", porcentaje2, " Número de aciertos: ", aciertos2, " Casos Totales: ", mval_lineal)
```

```
Porcentaje de acierto regresión lineal:  77.27660581377006  Número de aciertos:  215543  Casos Totales:  278924
```

Figure 22: Código de la regresión lineal en Phyton

Puesto que posteriormente hemos empleado regresión logística multi-clase y las métricas de error que empleamos no son válidas para comparar ambos modelos, hemos recurrido al % de acierto como referencia de la calidad de nuestros modelos. Es por ello que la agrupación por rangos mostrada en la figura 22 es necesaria para poder comparar este modelo con el de regresión logística, ya que la regresión logística indica la pertenencia o no a una clase (en nuestro caso, rangos de la variable carga) y en la regresión lineal predecimos el valor de la carga. Para ello hemos llevado a cabo el siguiente proceso:

1. Entrenar el modelo con el conjunto de entrenamiento.

2. Emplear el modelo para predecir la carga. Para ello hemos empleado el conjunto de validación.

3. Agrupar la carga en los mismos rangos que los empleados en la regresión logística.

4. Comparar la predicción con la carga real (ambas ya discretizadas).

5. Calcular el % de acierto del modelo empleando los datos de validación.

Finalmente tras todo el análisis en R y su posterior implantación en Phyton concluimos con una iteración que cuenta con un **porcentaje de acierto del 77.2%**, la cual emplea todos los predictores del dataset *regresión_logistica_training* menos el predictor que hace referencia al id.

0.6.2 2ª Fase del modelado - Regresión logística multi-clase

Este algoritmo ha sido desarrollado por nosotros empleando las técnicas vistas en asignaturas como Aprendizaje Automático, a diferencia de la regresión lineal anteriormente vista, en la cual empleábamos una librería para su cálculo, hemos decidido crear nosotros y realizar todo el proceso con el desarrollo de nuestras fórmulas . En las siguientes figuras 30,28,27,26 y 23 se muestra todo el código desarrollado (omitiendo carga de datos, librerías ...) para realizar este modelo.

A continuación explicamos por orden de ejecución, el funcionamiento de nuestro modelo desarrollado:

1. **Llamada a la función:**

   Indicamos como argumentos los conjuntos de entrenamiento y de validación, donde las variables X contienen todas las columnas con los predictores que deseamos emplear, y las variables Y contienen las 4 columnas que representan las clases de nuestra carga.

   La variable *num_labels* indica el número de clases con el que estamos trabajando, en nuestro caso 4.

   La variable *regularización* representa el termino de regularización que queremos emplear a nuestro modelo, en caso de no querer aplicar regularización se le indica un valor 0. La técnica de regularización permite disminuir el peso de los predictores en caso de estar sufriendo sobre-ajuste en nuestro modelo. El sobre-ajuste supone un problema porque lo que implica es que nuestro modelo está generalizando mal, es decir se ha producido sesgo. Una forma de identificarlo es probar a validar el modelo con los datos de entrenamiento y también de validación, en caso de estar sufriendo sobre-ajuste, el modelo tendrá mucho porcentaje de acierto con los datos de entrenamiento (es normal, ya que se ha entrenado con ellos), pero, a la hora de trabajar con datos nuevos, como es el conjunto de validación, el porcentaje de acierto se verá drásticamente disminuido.

Por último *mval* es número de datos con el que estamos validando, esto nos sirve para posteriormente comparar el número de aciertos con el número de casos y poder extraer el porcentaje de acierto de nuestro modelo.

```
print(oneVsAll(Xtrain_unos, Ytrain.T, num_labels, regularizacion, mval, Xvalidation_unos, Yvalidation))
```

Figure 23: Llamada a nuestro modelo con los parámetros

2. **Función One vs All**

Emplea el método ya explicado previamente en el apartado0.3.2. Además nos servimos de la función *opt.fmin_tnc* de la librería *optimize* la cual mediante la llamada que se muestra en la figura 26 nos permite minimizar la función de coste, es decir el error total que comete nuestro modelo al predecir las cargas. Definimos nuestra función de coste de la siguiente forma (figura: 24):

$$-\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Figure 24: Función de coste

Donde nuestra hipótesis es:

$$h_\theta(x) = g(\theta^T x)$$
$$g(z) = \frac{1}{1+e^{-z}}$$

Figure 25: Hipótesis de la regresión logística

La fórmula mostrada en la figura 25 indica que nuestra hipótesis es igual a la *función sigmoide* de un vector Theta (que contiene todos los pesos que el modelo ha calculado para nuestros predictores) multiplicado por la matriz X (que contiene todos los datos de entrada que le estamos suministrando al modelo.

El funcionamiento general de esta función One vs All es el siguiente:

1) Calcula el vector Theta con todos los pesos de los predictores que minimizan la función de coste.

2) Además, devuelve el coste asociado a nuestro modelo empleando esos pesos que permiten tener un coste mínimo.

3) Después, con esos pesos que hemos calculado y nuestro conjunto de validación, realizamos la predicción.

4) Nos quedamos con el valor más alto de los 4 asociados a la probabilidad de pertenecer a cada clase.

5) Por último, comparamos la predicción con los valores reales y devolvemos el porcentaje de acierto del modelo.

```
def oneVsAll(X, y_onehot, n_labels, reg, mval, Xvalidation, Yvalidation):
    theta_opt_por_clase = np.zeros((n_labels, np.shape(X)[1]))

    for etiqueta in range(n_labels):
        Thetas = np.zeros(np.shape(X)[1])
        result = opt.fmin_tnc(func=coste_regularizado, x0=Thetas,
                              fprime=gradiente_regularizado,
                              args=(X, y_onehot[etiqueta][:], reg))

        theta_opt_por_clase[etiqueta] = result[0]

    ThetaT = np.transpose(theta_opt_por_clase)
    producto = np.matmul(Xvalidation, ThetaT)
    prediccion = funcion_sigmoide(producto)

    columna_predecida = np.argmax(prediccion, axis= 1)

    aciertos = np.sum(columna_predecida == np.argmax(Yvalidation, axis= 1))
    porcentaje = (aciertos / mval) * 100
    print(aciertos)

    return porcentaje
```

Figure 26: Función One vs All

3. **Función de coste**

Implementación en código de la función de coste descrita en la figura 24. Recibe un vector *Theta* con los pesos asociados a los predictores, la matriz X con los datos de entrada de los predictores, la matriz Y con los datos sobre la carga y sus clases y el parámetro *landa* que representa el termino de regularización.

Esta función se encarga de calcular y devolver el coste de nuestro modelo al emplear unos pesos concretos para nuestro predictores (parámetro Theta).

```
def coste_regularizado(Thetas, X, Y, landa):

    m = np.shape(X)[0]
    matriz_traspuesta = np.matmul(np.transpose(np.log(funcion_sigmoide
                                          (np.matmul(X, Thetas)))), Y)

    matriz_traspuesta2 = np.matmul(np.transpose(np.log
                        (1 - funcion_sigmoide(np.matmul(X, Thetas)))), 1 - Y)

    primera_parte = -(matriz_traspuesta + matriz_traspuesta2) / m
    segunda_parte = (landa * (np.sum(Thetas ** 2))) / (2 * m)

    return primera_parte + segunda_parte
```

Figure 27: Función de coste de nuestro modelo

4. **Función de gradiente**

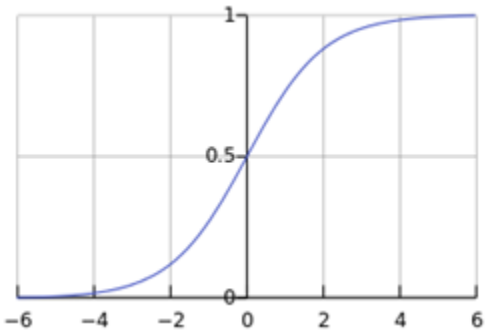
   La función mostrada en la figura 28 va recalculando el vector *Thetas* en cada iteración, de forma que empleando el método de descenso de gradiente, nuestro modelo finalmente converge en los *Thetas* óptimos que minimizan la función de coste.

```
def gradiente_regularizado(Thetas, X, Y, landa):

    Xtraspuesta = np.transpose(X)
    m = np.shape(X)[0]
    mult = np.dot(X, Thetas)
    H = funcion_sigmoide(mult)
    dif = H - Y
    primera_parte = np.matmul(Xtraspuesta, dif) / m
    regularizacion = (landa * Thetas[1:]) / m
    regularizacion = np.insert(regularizacion, 0, 0.0)

    return primera_parte + regularizacion
```

Figure 28: Función que va recalculando el vector *Thetas*

5. **Función sigmoide**

   La función logística o también llamada función sigmoide, puede tomar cualquier número de valor real y asginarle un valor en el intervalo [0,1]. Si se obtiene como salida un valor mayor 0.5 podemos clasificar el resultado como 1 (pertenece a la clase), si el valor es menor de 0.5 por el contrario será clasificado como 0 (no pertenece a la clase que estamos estudiando).



Fuente: Rafael Del Lama (2016), CC BY-SA 4.0 via Wikimedia Commons

Figure 29: Gráfica de la función sigmoide

```
def funcion_sigmoide(z):
    val = np.power(np.e, z)
    val = 1 / val

    return 1 / (1 + val)
```

Figure 30: Código de la función sigmoide

En las siguiente figura 31 se muestra el resultado de nuestra mejor iteración empleando regresión logística multi-clase, en la cual ya hemos seleccionado las métricas que mejoran el funcionamiento del algoritmo, como por ejemplo la elección de las variables que empleamos como predictores o el rango de valor adecuado para el término de regularización.

TIPO	Predictores	REGULARIZACIÓN	CASOS VALIDACION	ACIERTOS	% ACIERTOS
Logisitica	all	0	278,924	228652	81,97645
Logisitica	all	1	278,924	228753	82,01266
Logisitica	all	5	278,924	226230	81,10812
Logisitica	all	3	278,924	226232	81,10883
Logisitica	all	1,5	278,924	228631	81,96892
Logisitica	all - id	1	278,924	226264	81,12031

Figure 31: Ejemplo de iteraciones de la regresión logística multi-clase

0.7 Evaluación de los resultados

Puesto que la evaluación ha ido de la mano del modelado, aquí realizamos una breve comparativa de los resultados obtenidos en nuestros modelos finales, junto con un análisis de los mismos.

Por un lado resulta interesante ver como la variable *id* la cual hace referencia a la estación de medida donde se tomó el registro, no aporta información al modelo, nuestros modelos se mantenían igual o empeoraban al hacer uso de este predictor, lo que indica que solo aporta ruido y que por tanto la carga de las carreteras depende más del resto de factores que de la localización en sí.

Respecto a la regresión lineal simple hemos obtenido un primer modelo en R que nos indicaba que el modelo a través de los predictores suministrados, era capaz de explicar un 68.4% del comportamiento de la carga. Posteriormente, tras todo el trabajo realizado en las diferentes iteraciones y el paso al lenguaje de Phyton y el uso de sus librerías, obtuvimos unos resultados en los cuales nuestro modelo era capaz de acertar en un 77% de los casos el rango en el cual se encontraba la carga, lo cual para hacernos una aproximación sobre el estado de la carretera es bastante ilustrativo.

Finalmente, empleando la segunda técnica de modelado expuesta en este trabajo, regresión logística multi-clase, obtuvimos un modelo el cual cuenta con un 82% de acierto al realizar las predicciones sobre el rango de carga. Como podemos observar en los resultados el modelo de regresión logística multi-clase realiza una mejor estimación que la regresión lineal múltiple, posiblemente a causa de que la naturaleza del problema que hemos estudiado es más un problema de categorización que un problema de regresión, por lo que este método está expresamente diseñado para este tipo de estudios.

0.8 Conclusiones y trabajo a futuro

El ámbito de este trabajo es enorme y puede englobar y relacionarse con multitud de temáticas, es por ello que a pesar de tener más ideas sobre hacia donde dirigir el estudio, no las hemos podido llevar a cabo porque sería un trabajo demasiado extenso. A continuación expondremos las conclusiones extraídas de la elaboración de este proyecto y las potenciales vías hacía donde se puede enfocar para escalar el proyecto.

Nuestro estudio contaba con un importante factor como es la COVID-19, tras el análisis y modelado de los datos, podemos concluir que el impacto de la pandemia ha sido bastante notable en este sector, y bastante interesante de analizar, ya que, no ha sido un impacto puntual, podríamos decir que la pandemia ha cambiado ligeramente la forma de movilidad, aunque esto debería ser corroborado con más estudios teniendo en cuenta otros medios de transporte, pero en lo que concierne a nuestros datos, hemos observado, haciendo comparativas entre la antigua y la nueva normalidad, que hay un porcentaje de caída generalizado de las variables analizadas.

Uno de los objetivos de este proyecto era conocer más de cerca las diversas herramientas y metodologías de trabajo para el Data Science. A lo largo de la elaboración de este estudio nos hemos familiarizado con diversos lenguajes, empleado el uso de nuevas librerías, tratado con nuevas metodologías exclusivas de proyectos de este ámbito y enfrentado a multitud de problemas derivados de esta clase de proyectos. Nos ha permitido aprender sobre las necesidades y requisitos de un proyecto con tanto volumen de datos, así como resaltar la importancia de las técnicas de tratamiento de datos. Además, hemos podido aprender y conocer más de cerca la evolución del tráfico de nuestra ciudad y entender por qué resulta tan difícil comprender este fenómeno a la perfección. Por otro lado hemos trabajado de forma conjunta durante un periodo de 9 meses lo que ha resaltado la importancia de un buen trabajo en equipo en el cual exista comunicación y un buen reparto y manejo de las tareas a realizar. Por último, queremos hacer referencia al hecho de que hemos podido poner en práctica muchas de las técnicas y conocimientos aprendidos durante la carrera, lo que resulta gratificante al observar las aplicaciones reales de lo aprendido.

En lo referente al trabajo futuro, tenemos multitud de vías por el cual ampliar este proyecto, como por ejemplo:

1. **Contaminación acústica:** Primero nos gustaría mencionar una vía de trabajo que descartamos al inicio del proyecto, la cual relacionaba los datos de tráfico con datos de contaminación acústica. Los archivos cuentan con unos registros de coordenadas los cuales se podrían mapear y empleando conjuntos de datos adicionales como barrios y/o distritos, crear mapas o gráficas muy interesantes para su estudio.

2. **Creación de una red neuronal**: Nos hubiese gustado emplear más tipos de modelos como por ejemplo las redes neuronales, creemos que sería muy interesante ver como se puede adaptar este modelo a nuestros datos y ver que información nos arroja.

3. **Relacionarlo con la crisis en el sector del transporte público**: Una idea interesante sería relacionarlo con datos de transporte público de la ciudad de Madrid durante el mismo periodo, para ver que impacto ha tenido la pandemia, y si ha decantado a las personas hacia algún tipo de transporte en concreto.

4. **Relacionarlo con accidentes de tráfico:** Otra vía potencial es obtener datos sobre los siniestros ocurridos en el mismo periodo de tiempo en la ciudad de Madrid, estos datos, al igual que los referentes al uso del transporte público, se pueden encontrar en el Portal de Datos Abierto del Ayuntamiento de Madrid[1]. La idea sería llegar a encontrar ciertos patrones en las métricas de tráfico los cuales estén relacionados en un aumento de los accidentes de tráfico.

5. **Simulador de tráfico**: Por último, el proyecto más ambicioso, sería la creación de un simulador de tráfico el cual cuente con una interfaz gráfica y emplee todos las registros con los que hemos estado trabajando, de forma que no solo prediga la

carga sino todo tipo de variables. De esta forma se ofrecería una visión dinámica del flujo de tráfico y se podrían evitar situaciones no deseadas como colapsos o accidentes.

0.8 Conclusions and future work

The scope of this work is enormous and can encompass and relate to a multitude of topics. This is why, despite having more ideas about where to direct the study, we have not been able to carry them out because it would be too extensive a project. We will now present the conclusions drawn from the development of this project and the potential ways in which it can be focused to scale up the project.

Our study included an important factor such as COVID-19, after the analysis and modeling of the data, we can conclude that the impact of the pandemic has been quite notable in this sector, and quite interesting to analyze, since it has not been a one-off impact, we could say that the pandemic has slightly changed the form of mobility, although this should be corroborated with more studies taking into account other means of transport, but as far as our data is concerned, making comparisons between the old normalilty and the new one, that there is a generalized percentage drop in all the variables.

One of the goals of this project was to learn more about the various tools and methodologies for Data Science. Throughout the elaboration of this of this study we have familiarized ourselves with different languages, employed the use of new libraries, dealt with new methodologies exclusive to projects in this field and faced a multitude of problems derived from this kind of projects. It has allowed us to learn about the needs and requirements of a project with such a large volume of data, as well as to highlight the importance of data processing techniques. In addition, we were able to learn more about the evolution of traffic in our city and to understand why it is so difficult to understand this phenomenon perfectly. On the other hand, we have worked together for a period of 9 months, which has highlighted the importance of good teamwork in which there is communication and good Finally, we would like to refer to the fact that we have been able to put into practice many of the techniques and knowledge learned during the course, which is gratifying to see the real applications of what we have learned.

As far as future work is concerned, we have multitude ways through which to expand this project, such as:

1. **Noise pollution**: First, we would like to mention one way of work that we discarded at the beginning of the project, which related traffic with noise pollution data. The files have some coordinate records which could be mapped and using additional datasets such as neighborhoods and/or districts, to create very interesting maps or graphs for study.

2. **Creation of a neural network**: We would have liked to use more types of models such as neural networks, we think it would be very interesting to see how this model can be adapted to our data and see what information it gives us.

3. **Relate it to the crisis in the public transport sector**: An interesting idea would be to relate it to public transport data in the city of Madrid during the same period, to see what impact the pandemic has had, and if it has shifted people towards any particular type of transport.

4. **Relacionarlo con accidentes de tráfico**: Another potential way is to obtain data on accidents that occurred in the same period of time in the city of Madrid. these data, as well as those related to the use of public transport, can be found in the Open Data Portal of the Madrid City Council[1]. The idea would be to find certain patterns in traffic metrics which are related to an increase in traffic accidents.

5. **Traffic simulator**: Finally, the most ambitious project would be the creation of a traffic simulator with a graphical interface and uses all the registers we have been working with, so that it not only predicts the load but all kinds of variables. This would provide a dynamic view of the traffic flow and undesired situations such as collapses or accidents could be avoided.

0.9 Aportaciones individuales

0.9.1 Ana Álava Papí

En primer lugar, realicé tutoriales del lenguaje *R* en *RStudio* para la familiarización con el lenguaje y el entorno de desarrollo, para ello desarrollé proyectos individuales ayudándome de tutoriales y de la documentación oficial de *R*.

En cuanto a la búsqueda de datos para nuestro TFG, nuestra tutora Sonia Estévez nos recomendó el *Portal de datos abiertos del Ayuntamiento de Madrid*[1], por lo que revisamos los datos de la página buscando una temática que pudiera ofrecernos un estudio amplio y diversas formas de enfoque.

Inicialmente nos decantamos por hacer un estudio de la relación del tráfico con la contaminación acústica de la ciudad de Madrid acompañados de las ubicaciones de sus respectivos puntos de medida, pero al comenzar el estudio nos dimos cuenta del gran volumen de datos, información y posibilidades que ofrecían los datos de tráfico, por lo que finalmente, nos decidimos por hacer un estudio únicamente con los datos de tráfico y las ubicaciones de sus puntos de medida.

Con los datos ya seleccionados nos dispusimos a realizar un análisis preliminar para la comprensión de las variables y la localización de posibles anomalías. A partir de aquí comenzamos a realizar limpiezas sencillas, es decir, nos centramos en eliminar valores

NAs que corrompían la información, así como patrones anómalos. Durante el proceso de limpieza de los datos, me centré en la optimización de funciones, ya que, al tener un gran volumen de datos, ralentizaba mucho su procesamiento y en ocasiones el ordenador no era capaz de procesarlos.

Puntualmente, nos surgieron dudas con respecto al comportamiento de ciertos datos, por lo que, a través de nuestra tutora, nos pusimos en contacto con los responsables de los datos del Ayuntamiento de Madrid enviando un vídeo y exponiendo nuestras dudas, las cuales nos resolvieron y pudimos continuar y readaptar nuestra limpieza.

En este punto decidimos comenzar a redactar la memoria, para ir documentando los pasos de nuestra limpieza de forma más detallada, para esto, realizamos una reunión con nuestra tutora en la que nos inició en Latex proporcionándonos una plantilla inicial y unas reglas básicas.

A continuación, nos repartimos las tareas.

Mi tarea consistió en realizar un análisis descriptivo de las variables, para ello, me centré en las variables más significativas, y traté de reflejar el impacto de la pandemia en los resultados:

- En primer lugar, se hace una representación anual de la variable *intensidad*, de manera que se pueda ver el impacto de la pandemia en el número de vehículos en las vías, de esta forma presentamos la tendencia que van a seguir nuestros datos a lo largo del estudio.

  Para hacer un mayor hincapié, se representa el mes de abril para los tres años, perteneciendo abril de 2020 íntegramente al periodo de confinamiento.

  Ambas representaciones en un gráfico de barras.

- Para la representación de la variable *ocupacion*, se analiza mensualmente, en un gráfico de barras, pero únicamente para los años 2019 y 2021, de manera que se pueda hacer una comparativa entre la antigua y la nueva normalidad, centrándonos posteriormente en un análisis por horas en el que se pretende relacionar la evolución de la *ocupacion* con la aparición del teletrabajo.

- Para la representación de la variable *carga*, se analizan, en un gráfico de barras, los tres años en el periodo correspondiente a las tres fases de desescalada.

  Adicionalmente, representé la variable *carga* en un mapa de calor por distritos, integrando Python y R, para ello me informé de las distintas librerías, tanto en R como en Python para la generación de mapas, finalmente decidí realizar la preparación de las tablas de datos en R y el tratamiento de los datos y la generación del mapa en Python. Para ello utilicé los datos proporcionados de la ubicación de los puntos de medida a modo de conexión entre los datos de los distritos y los datos de tráfico como se puede ver en la tabla 10, de esta forma representé los valores medios de carga en cada distrito con distintas tonalidades, siendo la más oscura la de mayor carga y la más clara la de menor.

Finalmente participé en el desarrollo de la memoria.

0.9.2 Martín Rovira Barrionuevo

Al comienzo del curso académico empecé con la búsqueda de información relativa a las herramientas que íbamos a emplear, como por ejemplo el lenguaje R. Para ello, nuestra tutora, Sonia Estévez, nos facilitó alguna documentación la cual proponía mini trabajos que a pesar de ser de temáticas distintas vino muy bien para ir cogiendo soltura con el lenguaje. Además de esta documentación, investigué por otras fuentes como son la biblioteca de la facultad (online), youtube e internet en general.

Durante las primeras semanas estuvimos buscando conjuntos de datos que nos pareciesen interesantes y contasen con datos de calidad para realizar nuestro estudio. Tras evaluar diversas opciones y centrarnos en los datos de tráfico de la ciudad de Madrid, empezamos el trabajo con los datos de forma conjunta a través de reuniones en la plataforma Discord, donde mi compañera y yo empezamos a realizar las primeras fases de limpieza y comprensión de los datos de manera conjunta. Posteriormente, contactábamos con nuestra tutora para comentar los avances y las dudas que iban surgiendo. Inicialmente estas reuniones eran de carácter semanal, lo cual era propiciado por la gran cantidad de dudas originadas de empezar el trabajo y el uso de un lenguajes prácticamente nuevos como eran R y Latex.

Después de este comienzo, pasamos a un análisis más profundo de los datos y esto implicó nuevas dudas y nuestra tutora Sonia nos facilitó el contacto de los responsables del Portal de Datos del Ayuntamiento de Madrid, a los cuales les mandamos un vídeo con una presentación sobre las dudas que no eramos capaces de resolver (dicho vídeo y la presentación asociada está disponible en la carpeta de anexos del trabajo). Tras unos cuantos correos con el Ayuntamiento de Madrid, resolvimos todas nuestras dudas y procedimos a corregir y aumentar la limpieza de acuerdo a la información obtenida.

Una vez terminadas las fases de limpieza y los meses iniciales, cambiamos el plan de trabajo de forma que nos repartíamos tareas para poder avanzar más rápidamente y las reuniones empezaron a ser cada 2 semanas.

Por mi lado me centré en el formateo y una segunda limpieza de los datos para prepararlos para entrenar a los modelos que íbamos a emplear posteriormente. Me centré en estudiar las variables, ver como podía adaptarlas para que se comportasen de forma adecuada e investigué las posibles predictores a crear para aportar un extra de información. Esta fase previamente mencionada requirió mucho tiempo pues tuve que volver constantemente a realizar reajuste de acuerdo a las necesidades que iban surgiendo, por lo que no se trató de una sola iteración, sino multitud de pequeñas iteraciones desarrolladas a lo largo de los meses. Posteriormente pasé a la creación de los modelos mediante Python. Una vez acabados los modelos, empecé a trabajar en el modelado, intentando buscar la combinación óptima de las métricas para maximizar el rendimiento de nuestros modelos.

Durante todo el proceso realizado fui documentando junto a mi compañera todos los pasos realizados, de forma que el presente documento ha sido creado poco a poco a lo largo de los meses. Finalmente y de forma conjunta, mi compañera y yo revisamos toda la documentación y completamos de acuerdo a las especificaciones.

Bibliography

[1] Honorio Enrique Crespo Díaz Alejo. Portal de datos abiertos del ayuntamiento de madrid. *Consultor de los ayuntamientos y de los juzgados: Revista técnica especializada en administración local y justicia municipal*, (3):128–144, 2020.

[2] Estudio de la intensidad media diaria de vehículos (imd), 2020. `https://www.comunidad.madrid/servicios/transporte/estudio-intensidad-media-diaria-vehiculos-imd` (última visita, 16-05-2022).

[3] Estado de la movilidad de la ciudad de madrid, 2019. `https://transparencia.madrid.es/FWProjects/transparencia/Movilidad/Trafico/InformesMovilidad/InformeMovilidad2019.pdf` (última visita, 16-05-2022).

[4] Daniele Grasso y Borja Andrino. ¿cómo ha cambiado la movilidad en madrid? un millón de viajes menos en transporte y coches como antes. 2021. `https://elpais.com/espana/madrid/2021-12-12/como-ha-cambiado-la-movilidad-en-madrid-un-millon-de-viajes-menos-al-dia-en-transporte-y-trafico-como-antes-de-la-pandemia.html` (última visita, 16-05-2022).

[5] Joaquín Amat Rodrigo. Introducción a la regresión lineal múltiple. 2016. `https://www.cienciadedatos.net/documentos/25_regresion_lineal_multiple` (última visita, 16-05-2022).

[6] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. `https://ggplot2.tidyverse.org` (última visita, 16-05-2022).