UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo Fin de Máster del
Máster Universitario en Ingeniería y Ciencia de Datos

# Diseño y comparación de modelos para la predicción del flujo de tráfico

Elena Sobrini García

Dirigido por: María der Rocío Muñoz mansilla

Agustín Carlos Caminero Herráez

Curso: 2021-2022: 1ª Convocatoria

# Resumen

La organización del tráfico es un problema que tienen que afrontar día a día la mayoría de las grandes ciudades del mundo. Una de las herramientas que ayuda a mantener una vías más libres y con circulación más fluida es conocer con la mayor precisión posible el estado del tráfico en las próximas horas.

En este proyecto se utilizan los datos públicos de tráfico y meteorología disponibles en el Portal de datos abiertos del Ayuntamiento de Madrid para diseñar, evaluar y comparar modelos predictivos del flujo de tráfico en la misma ciudad. Los modelos comparados varían en cuanto a las variables utilizadas, las transformaciones aplicadas y el modelo de aprendizaje automático usado. Además, algunos de los modelos son de predicción a corto plazo, mientras que otros, son de predicción a largo plazo.

Podemos agrupar los modelos utilizados en los siguientes tipos: modelos básicos de referencia, modelos de regresión lineal, modelos de potenciación del gradiente y un modelo de redes neuronales en grafos.

Los modelos de regresión lineal y de potenciación del gradiente los vamos a considerar modelos a largo plazo, mientras que el modelo de redes neuronales en grafos, a corto plazo. Estos dos grupos los comparamos con sus correspondientes modelos de referencia y veremos en qué casos una estructura más compleja obtiene mejores resultados que las sencillas de referencia.

Finalmente, se extraen conclusiones y se proponen futuras vías de trabajo en el área.

# Abstract

The organization of the traffic is a problem that most of the big cities in the world have to face day after day. One of the tools that helps by keeping a smoother flow of traffic is to know as accurately as possible the traffic conditions for the next few hours.

In this project, the public data about the traffic and meteorology of the city of Madrid, published by the local government, are used to design, evaluate and compare predictive models of the traffic flow in the city. The models compared vary in terms of the variables used, the transformations applied and the machine learning methods used. Moreover, some of the models are for short-term forecasting, while the others are for long-term forecasting.

We can group the models into the following sets: basic benchmark models, linear regression models, gradient boosting models and a graph neural network model.

The linear regression and the gradient boosting models are the long-term forecasting methods, while the graph neural network model (GNN) is a short-term forecasting method. We are going to compare these two groups with the corresponding baseline models and assess when a more complex model achieves a better result than a simple one.

Finally, conclusions are drawn and future avenues of work in the area are proposed.

# Contenido


**1 Introducción general y objetivos** **1**

1.1 Motivación y objetivos . . . . . 1

1.2 Planteamiento del problema . . . . . 1

1.3 Estructura de la memoria . . . . . 2

**2 Estado del arte** **3**

2.1 Métodos utilizados . . . . . 3

2.2 Variables a utilizar . . . . . 6

2.3 Variable objetivo . . . . . 6

**3 Análisis explorativo de los datos** **9**

3.1 Datos de tráfico . . . . . 9

3.1.1 Origen de los datos . . . . . 9

3.1.2 Magnitudes de tráfico . . . . . 11

3.1.3 Filtrado de datos . . . . . 11

3.1.4 Exploración de los datos de tráfico . . . . . 11

3.1.5 Discusión sobre la variable objetivo a utilizar . . . . . 12

3.1.6 Elección de los sensores . . . . . 13

3.2 Datos meteorológicos . . . . . 14

3.2.1 Origen de los datos . . . . . 15

3.2.2 Magnitudes meteorológicas . . . . . 15

3.2.3 Filtrado de datos . . . . . 15

3.2.4 Disponibilidad de los datos meteorológicos . . . . . 15

3.3 Combinación de los datos . . . . . 17

3.4 Dificultades respecto a la disponibilidad de datos . . . . . 18

**4 Metodología para el entrenamiento y la evaluación de modelos** **19**

4.1 Datos de entrenamiento y de test . . . . . 19

4.2 Validación cruzada . . . . . 19

4.2.1 Selección de modelos . . . . . 20

4.2.2 Evaluación de modelos . . . . . 20

**5 Teoría de los modelos utilizados 23**
5.1 Regresión lineal . . . . . 23
5.1.1 Características del modelo . . . . . 23
5.1.2 Regularización . . . . . 24
5.2 XGBoost . . . . . 24
5.2.1 Función objetivo regularizada . . . . . 25
5.2.2 Potenciación de gradiente . . . . . 25
5.2.3 *Shrinkage* y submuestreo de las variables explicativas . . . . . 27
5.2.4 Algoritmos de búsqueda de las divisiones . . . . . 27
5.2.5 Búsqueda cuando los datos son dispersos . . . . . 28
5.3 Red neuronal recurrente convolucional de difusión en grafos (GNN) . . . . . 28
5.3.1 Planteamiento del problema en un grafo . . . . . 28
5.3.2 Red neuronal recurrente con proceso de convolución de difusión . . . . . 28
5.3.2.1 Dependencia espacial . . . . . 29
5.3.2.2 Dependencia temporal . . . . . 29

**6 Aplicación de los modelos 33**
6.1 Modelos básicos de referencia . . . . . 34
6.1.1 Estimador global . . . . . 34
6.1.1.1 Regresor de media . . . . . 34
6.1.1.2 Regresor de mediana . . . . . 34
6.1.2 Regresor dependiente de las otras variables . . . . . 35
6.1.2.1 Estimador dependiente de la hora . . . . . 35
6.1.2.2 Estimador dependiente de la hora y del tipo de día . . . . . 36
6.1.3 Modelos basados en repetición . . . . . 36
6.1.3.1 Repetición completa . . . . . 37
6.1.3.2 Repetición del último valor . . . . . 37
6.1.4 Modelo de variación . . . . . 37
6.2 Regresión lineal . . . . . 38
6.2.1 Variables utilizadas . . . . . 38
6.2.2 Posibles transformaciones de las variables . . . . . 38
6.2.3 Selección del modelo . . . . . 39
6.2.4 Visualización de los dos modelos . . . . . 41
6.3 XGBoost . . . . . 43
6.3.1 Variables explicativas a utilizar . . . . . 43
6.3.2 Ajuste de parámetros de XGBoost . . . . . 43

6.3.3 Procedimiento de ajuste . . . . . 46
6.4 Red neuronal recurrente convolucional de difusión en grafos (GNN) . . . . . 49
6.4.1 Construcción del grafo . . . . . 49
6.4.2 Deep Graph Library (DGL) . . . . . 50
6.4.3 Observaciones sobre el modelo . . . . . 50
6.4.4 Entrenamiento final . . . . . 52
6.4.5 Interrupción del entrenamiento . . . . . 53
6.4.6 Comentarios sobre el entrenamiento . . . . . 53

**7 Resultados** **59**
7.1 Análisis cuantitativo . . . . . 59
7.1.1 Modelos a largo plazo . . . . . 59
7.1.2 Modelos a corto plazo . . . . . 61
7.1.2.1 Cuando $T = T' = 2$ . . . . . 61
7.1.2.2 Cuando $T = T' = 4$ . . . . . 62
7.1.2.3 Cuando $T = T' = 8$ . . . . . 63
7.1.2.4 Cuando $T = T' = 16$ . . . . . 63
7.1.2.5 Resumen de los mejores modelos a corto plazo . . . . . 65
7.2 Análisis cualitativo . . . . . 65

**8 Conclusiones y trabajos futuros** **71**
8.1 Conclusiones . . . . . 71
8.1.1 Predicción a largo plazo . . . . . 71
8.1.2 Predicción a corto plazo . . . . . 71
8.2 Trabajos futuros . . . . . 72
8.2.1 Variables explicativas . . . . . 72
8.2.2 Red neuronal de grafos con difusión convolucional . . . . . 72
8.2.3 Tratamiento de los valores nulos . . . . . 72

**A Transformación de las variables temporales** **79**
A.1 Mantener la variable es su forma original . . . . . 79
A.2 Transformación *one-hot* . . . . . 79
A.3 Transformada de Fourier . . . . . 80
A.4 Splines periódicos . . . . . 81

**B Transformación de las variables meteorológicas** **83**
B.1 Precipitación . . . . . 83
B.1.1 Mantener las variables en su forma original . . . . . 83
B.1.2 Transformación *one-hot* . . . . . 84

B.1.3 Transformación ordinal . . . . . 84
B.1.4 Transformación de potencia: Yeo-Johnson . . . . . 84
B.1.5 Transformaciones de cuantiles . . . . . 85
B.2 Viento . . . . . 86

# Índice de figuras


3.1 Distribución de todos los sensores de tráfico en la ciudad de Madrid . . . . . 10
3.2 Análisis de las magnitudes de tráfico a lo largo de los años 2019, 2020 y 2021 en el punto de medida 1001. . . . . 12
3.3 Número de observaciones para el total de los puntos de medida de cada magnitud de tráfico a lo largo de los años 2019, 2020 y 2021. . . . . 13
3.4 Los 37 sensores seleccionados para evaluar los modelos. . . . . 14
3.5 Número de observaciones para el total de los estaciones de medida de cada magnitud meteorológica a lo largo de los años 2019, 2020 y 2021. . . . . 16

4.1 Validación cruzada en series temporales . . . . . 20

5.1 Esquema del modelo GNN con convolución de difusión . . . . . 31

6.1 Estimador de media . . . . . 35
6.2 Estimador de mediana . . . . . 35
6.3 Estimador de media por hora del día . . . . . 36
6.4 Estimador de mediana por hora del día . . . . . 36
6.5 Estimador de media por hora y tipo de día . . . . . 36
6.6 Estimador de mediana por hora y tipo de día . . . . . 37
6.7 Repetición completa . . . . . 37
6.8 Repetición de la observación realizada en $t$ durante $T$ veces . . . . . 37
6.9 Modelo de variación . . . . . 38
6.10 Ejemplo de regresión en cinco sensores y un período de 24 horas. . . . . 42
6.11 Coeficientes de la regresión lineal con la primera configuración . . . . . 44
6.12 Coeficientes de la regresión lineal con la segunda configuración . . . . . 45
6.13 Distribución de los parámetros finales para XGBoost . . . . . 47
6.14 Importancia de las variables explicativas en un modelo de XGBoost . . . . . 48
6.15 La red de puntos de medida en forma de grafo, utilizando tres valores de $\kappa$ diferentes 50
6.16 Evolución del MAE en el entrenamiento de la configuración 1 de la red neuronal recurrente convolucional de difusión en grafos (GNN) . . . . . 54

6.17 Evolución del MAE en el entrenamiento de la configuración 2 de la red neuronal recurrente convolucional de difusión en grafos (GNN) . . . . . 55
6.18 Evolución del MAE en el entrenamiento de la configuración 3 de la red neuronal recurrente convolucional de difusión en grafos (GNN) . . . . . 56
6.19 Evolución del MAE en el entrenamiento de la configuración 4 de la red neuronal recurrente convolucional de difusión en grafos (GNN) . . . . . 57
6.20 Evolución del MAE en el entrenamiento de la configuración 5 de la red neuronal recurrente convolucional de difusión en grafos (GNN) . . . . . 58

7.1 Mejores modelos a corto plazo para MAE . . . . . 65
7.2 Mejores modelos a corto plazo para MSE . . . . . 66
7.3 Los dos modelos de regresión lineal y XGBoost en un ejemplo . . . . . 66
7.4 Los dos modelos de regresión lineal y XGBoost en un segundo ejemplo . . . . . 66
7.5 Las cinco configuraciones de GNN en ejemplos en un mismo sensor cuando $T = T' = 2$ . . . . . 67
7.6 Las cinco configuraciones de GNN en ejemplos en un mismo sensor cuando $T = T' = 4$ . . . . . 68
7.7 Las cinco configuraciones de GNN en ejemplos en un mismo sensor cuando $T = T' = 8$ . . . . . 68
7.8 Las cinco configuraciones de GNN en ejemplos en un mismo sensor cuando $T = T' = 16$ . . . . . 69

A.1 Forma original de la variable hora durante dos días. . . . . 80
A.2 Transformación one-hot sobre las horas de un día. . . . . 80
A.3 Cuatro términos de la transformada de Fourier sobre las horas de dos días. . . . 81
A.4 Cuatro splines codificando la variable hora durante dos días. . . . . 81

B.1 Forma original de la variable precipitación durante dos días. . . . . 84
B.2 Transformación *one-hot* de la variable precipitación durante dos días. . . . . 84
B.3 Transformación ordinal de la variable precipitación durante dos días. . . . . 85
B.4 Transformación de potencia de la variable precipitación durante dos días. . . . . 85
B.5 Transformación cuantil con distribución uniforme de la variable precipitación durante dos días. . . . . 86
B.6 Transformación cuantil con distribución normal de la variable precipitación durante dos días. . . . . 86

# Índice de tablas


6.1 Las dos configuraciones de regresión lineal utilizadas . . . . . 41
6.2 Las 5 configuraciones diferentes de la red neuronal de grafos (GNN) . . . . . 52

7.1 MAE y MSE de cada modelo a largo plazo. . . . . 60
7.2 Mejores modelos a largo plazo según MAE y MSE. . . . . 60
7.3 MAE y MSE de cada modelo a corto plazo con $T = T' = 2$. . . . . 61
7.4 Mejores modelos a corto plazo, para $T = T' = 2$ según MAE y MSE. . . . . 62
7.5 MAE y MSE de cada modelo a corto plazo con $T = T' = 4$. . . . . 62
7.6 Mejores modelos a corto plazo, para $T = T' = 4$ según MAE y MSE. . . . . 63
7.7 MAE y MSE de cada modelo a corto plazo con $T = T' = 8$. . . . . 63
7.8 Mejores modelos a corto plazo, para $T = T' = 8$ según MAE y MSE. . . . . 64
7.9 MAE y MSE de cada modelo a corto plazo con $T = T' = 16$. . . . . 64
7.10 Mejores modelos a corto plazo, para $T = T' = 16$ según MAE y MSE. . . . . 64

# Capítulo 1

# Introducción general y objetivos

## 1.1 Motivación y objetivos

La congestión de tráfico tiene consecuencias en la productividad, costes de energía, contaminación (y, por consiguiente, salud) y seguridad, entre otros. La predicción y consiguiente planificación de tráfico son herramientas que pueden marcar una diferencia para combatir este problema. A grandes rasgos, el objetivo de esta área de investigación consiste en estimar, de la manera más precisa posible, una magnitud sobre el tráfico en una región e intervalo en el futuro concretos.

El código con el que se han realizado los experimentos para este trabajo se encuentra en el repositorio Sobrini García (2022).

## 1.2 Planteamiento del problema

El problema que nos planteamos consiste en predecir los niveles de tráfico en un conjunto de sensores $\mathcal{V}$, con $|\mathcal{V}| = N$, en la ciudad de Madrid, en los $T$ períodos de 15 minutos posteriores a un instante $t$. Si $P$ es el número de variables (diferentes al tráfico) en cada punto (por ejemplo, variables que indiquen si es un día laborable, que contengan información meteorológica, que indiquen la hora del día, etc), entonces definimos $X \in \mathbb{R}^{N \times P}$ como la matriz de variables explicativas. Por otra parte, definimos $Y \in \mathbb{R}^N$como el vector con los valores de tráfico. Sean $X(t)$ e $Y(t)$ los valores observados en el momento $t$.
Como veremos en 6, los modelos que utilizamos se pueden clasificar en dos grupos, dependiendo de cuál sea el planteamiento exacto del problema.

1. **Modelos no dependientes de los instantes anteriores** o **modelos de predicción de tráfico a largo plazo**: En el primer grupo de modelos, el objetivo consiste en aprender una función $h(\cdot)$:

$$[X^{(t+1)}, \ldots, X^{(t+T)}] \xrightarrow{h(\cdot)} [\hat{Y}^{(t+1)}, \ldots, \hat{Y}^{(t+T)}]$$

donde $\hat{Y}^{(t+1)}, \ldots, \hat{Y}^{(t+T)}$ estén lo más cerca posible de $Y^{(t+1)}, \ldots, Y^{(t+T)}$. Es decir, el modelo aprende a predecir el tráfico en un momento concreto, dadas las otras variables en ese instante. En la literatura, estos modelos se conocen como **modelos de predicción de tráfico a largo plazo**, o ***Long-term traffic forecasting***, ya que tratan de encontrar patrones utilizando variables explicativas, como, por ejemplo, las variables temporales o las características del punto en que se quiere modelar el tráfico.

2. **Modelos dependientes de los instantes anteriores** o **modelos de predicción de tráfico a corto plazo**: En el segundo grupo de modelos, el objetivo consiste en aprender una función $h(\cdot)$:

$$[X^{(t-T'+1)}, \ldots, X^{(t)}, Y^{(t-T'+1)}, \ldots, Y^{(t)}] \xrightarrow{h(\cdot)} [\hat{Y}^{(t+1)}, \ldots, \hat{Y}^{(t+T)}] \tag{1.1}$$

donde $\hat{Y}^{(t+1)}, \ldots, \hat{Y}^{(t+T)}$ estén lo más cerca posible de $Y^{(t+1)}, \ldots, Y^{(t+T)}$. Es decir, el modelo aprende a predecir el tráfico en un momento concreto, dadas todas las otras variables y el tráfico en los $T'$ instantes anteriores. En la literatura, estos modelos se conocen como **predicción de tráfico a corto plazo**, ya que tienen como objetivo predecir el tráfico en los instante futuros, dadas observaciones recientes.

Como podemos deducir de esta explicación, para utilizar un modelo de los dependientes de los instantes anteriores, necesitamos conocer el tráfico en los $T'$ períodos exactamente previos al momento de la predicción. En cambio, los modelos no dependientes de los instantes anteriores necesitan conocer las otras variables en el momento exacto de la predicción.

## 1.3 Estructura de la memoria

La memoria de esta proyecto se estructura en los siguientes capítulos:

1. Introducción general y objetivos
2. Estado del arte
3. Análisis explorativo de los datos
4. Metodología para el entrenamiento y evaluación de modelos
5. Teoría de los modelos utilizados
6. Aplicación de los modelos
7. Comparación de resultados
8. Conclusiones y trabajos futuros

# Capítulo 2

# Estado del arte

El área de investigación de la predicción de tráfico es principalmente activa desde finales de los años 70, y su interés ha sido creciente a lo largo de las últimas décadas, debido a el exponencial aumento de capacidad computacional y desarrollo de métodos predictivos, así como por la creciente necesidad y el aumento de la población en las grandes ciudades.

Con la misma clasificación que se ha formulado en 1.2, la mayoría de los modelos predictivos de tráfico se pueden encuadrar en uno de los siguientes grupos, dependiendo del problema que se quiere resolver:

- **Predicción a corto plazo**: Son aquellos modelos que toman una cantidad observaciones recientes (con nuestra notación, $T'$ observaciones recientes), y predicen un valor para los próximos instantes (con nuestra notación, para los próximos $T$ instantes).
- **Predicción a largo plazo**: Son aquellos modelos que deducen patrones del comportamiento del tráfico a partir de otras variables, y permiten predecir el tráfico en un momento determinado del futuro, basado en las mismas variables.

Sin embargo, como se menciona en el aparatado *When is a forecast considered long-term?* en Manibardo et al. (2020), la definición de estos dos grupos se formula a menudo en términos de distancia temporal al objetivo de predicción. Por ejemplo, algunos autores consideran que una predicción con menos de una hora de antelación es a corto plazo, mientras que una predicción con más de una hora de antelación es largo plazo.

## 2.1 Métodos utilizados

En van Lint and van Hinsbergen (2012) se hace la siguiente clasificación de modelos predictivos de tráfico, en cuanto a los métodos usados:

- **Métodos naive**: Son métodos que hacen conjeturas simples sobre los datos y basan sus predicciones en ellas. Por ejemplo, tomando siempre como predicción la media histórica.

En este trabajo vamos a ver este tipo de modelos como modelos básicos de referencia, descritos en 6.1.

- **Modelos paramétricos**: Son métodos que basan su conocimiento en principios teóricos, y cuyos parámetros se ajustan mediante la evaluación en los datos. Un ejemplo es estos modelos son los basado en la simulación. En este trabajo no se consideran este tipo de modelos.

- **Modelos no paramétricos**: Estos modelos basan su aprendizaje en los datos disponibles. El nombre no implica que no utilicen parámetros, solo que su número y valor no están prefijados. La estructura y ajuste de estos modelos depende de los datos. En este trabajo se encuadran en este grupo los siguientes modelos: regresión lineal, XGBoost y red neuronal de grafos.

La primera etapa de investigación en este área se basaba mayormente en métodos paramétricos que utilizan métodos clásicos de la estadística para predecir el tráfico en un punto concreto, pero ha ido evolucionando gradualmente hacia métodos no paramétricos.

Dentro de los modelos no parámetricos, hasta hace pocos años, se consideraba que había dos *escuelas de pensamiento* en cuanto a la modelización de tráfico: aquella que basaba sus modelos en teorías estadísticas y aquella que basaba sus modelos en aprendizaje automático, aún sin una teoría estadística detrás. Entre estas dos corrientes de investigación había una mala o inexistente comunicación (Karlaftis and Vlahogianni (2011)).

En los primeros, de los más populares fueron los modelos autoregresivos integrados de media móvil (ARIMA, Hyndman and Athanasopoulos (2021a)). Por ejemplo, Lee and Fambro (1999) aplica modelos autoregresivos por subconjuntos y Van Der Voort et al. (1996) introduce un modelo llamado *KARIMA*, que encadena un clasificador basado en mapas autoorganizados de características (Kohonen (1982)) con un modelo ARIMA (que depende del resultado del clasificador). También se han evaluado modelos autoregresivos estacionales en Williams and Hoel (2003).

Debido a la complejidad de las dependencias del flujo de tráfico, la investigación se ha ido desligando de modelos puramente estadísticos, para concentrarse en modelos de aprendizaje automático.

En Dong et al. (2018) se utiliza el modelo *XGBoost* para realizar predicciones a corto plazo sobre el número de coches cada dos minutos. En Lu et al. (2020) se utilizan procesos de Markov, aprendizaje conjunto o *ensemble* y métodos basados en regresión y autoregresión para predecir el flujo de tráfico a corto plazo. En Zhang and Haghani (2015) se utiliza un modelo de *gradient boosting* para capturar la dinámica del tráfico y realizar predicciones multipaso.

Multitud de problemas en muchas áreas de investigación se han intentado resolver en los últimos años con redes neuronales y, concretamente, aprendizaje profundo. Esto es debido a su

flexibilidad y capacidad de aprendizaje (teóricamente pueden aproximar cualquier función no lineal), así como a que actualmente sí se tiene disponibilidad a la cantidad necesaria de datos para su entrenamiento, algo que antes raramente era posible. Además el aprendizaje profundo, al contrario que los métodos tradicionales de aprendizaje automático, puede teóricamente aprender la jerarquía de las variables de entrada, sin necesidad de realizar transformaciones previas.

Una explicación muy resumida de la teoría detrás de estos métodos es la siguiente: las redes neuronales son estructuras de varias capas ordenadas, en cada una de las cuales hay un número de neuronas. Una neurona toma un vector de entrada, lo combina con un vector de parámetros locales (los pesos) y con información local y da como resultado un escalar, el cual o bien se transfiere a la siguiente capa, o bien es el resultado de la red neuronal.

En Manibardo et al. (2020) se hace una revisión crítica de los métodos de esta clase aplicados en la predicción de tráfico a corto plazo y remarca que el aprendizaje profundo se utiliza en este área principalmente desde la segunda década de este siglo. En este artículo se realiza además un estudio empírico, comparando modelos básicos de referencia, modelos de regresión, modelos de aprendizaje conjunto y, finalmente, modelos de aprendizaje profundo. Los autores argumentan que los resultados obtenidos por los modelos de aprendizaje profundo no son tan buenos en comparación con los los de los otros tipos de modelos como para justificar la complejidad y necesidad de capacidad de computación que tienen. Sin embargo, este artículo utiliza modelos que no están diseñados para aprender de las dependencias espaciales en una región.

En Liu et al. (2019) se utilizan redes convolucionales para aprender estructuras espacio-temporales y predecir el flujo de tráfico de autobuses. En Sun et al. (2020) se propone una red profunda llamada *TFFNET*, que utiliza capas convolucionales para aprender dependencias espaciotemporales en la ciudad de Wuhan y predecir el tráfico.

Con más impacto mediático, en el artículo de prensa Fisher (2020) se cuenta como Google colabora con la empresa DeepMind para mejorar las estimaciones de estimación de tiempo de llegada, basándose en redes reuronales de grafos (Lange and Perez (2020)). En Tampubolon and Hsiung (2018) se utiliza una red neuronal completamente conectada, entrenada con técnicas de normalización de batch (Ioffe and Szegedy (2015)) y *dropout* (Hanson (1990)).

En cuanto las dependencias temporales, en Osipov et al. (2020) se propone una red neuronal recurrente (RNN) con estructura espiral y que, por lo tanto, puede aprender de manera continua. En Yang et al. (2019) se utiliza también una red neuronal recurrente, en concreto LSTM, con secuencias largas. En estos casos, la red tiene dificultades para aprender. En el mencionado artículo se utiliza un mecanismo de atención, que da más importancia a las variables explicativas que más influencia tienen para solucionar este problema.

Los enfoques mencionado hasta ahora tienen como objetivo predecir el tráfico a corto plazo, aunque, en ocasiones, los modelos tengan un componente de aprendizaje a largo plazo.

Existen otros modelos cuyo objetivo es la predicción a largo plazo. Es decir, extraer patrones de comportamiento del tráfico, dependientes de otras variables, que nos permitan estimar el

tráfico en un instante del futuro, independientemente de la proximidad temporal. Los autores de la publicación Belhadi et al. (2020) mencionan en ella misma que, según su entendimiento, son los primeros en predecir flujos del tráfico a largo plazo. En ella, utilizan información contextual, como los eventos del día o la información meteorológica, para entrenar varios modelos basado en redes profundas. En Wang et al. (2021) se menciona el problema que tienen en muchas ocasiones los modelos iterativos con muchos pasos, y es que el error se acumula y amplifica. Por ello, utilizan un modelo *encoder-decorder* con LSTM, llamado *seq2seq* (Sutskever et al. (2014)), y que normalmente se utiliza para la traducción automática y otras tareas de procesamiento del lenguaje natural. En Karim et al. (2019) se utiliza una red neuronal para aprender a predecir el flujo de tráfico a partir de simplemente el mes, el día de la semana, la hora del día y si es festivo. Aunque nosotros no usamos redes neuronales para el problema de la predicción a largo plazo (sino que usamos regresión lineal y *XGBoost*), el planteamiento de esta publicación en cuanto a variables de entrada y de salida es muy parecido al que nosotros seguimos en los dos modelos mencionados.

Otro modelo que también se autodenomina de largo plazo, pero que tiene un planteamiento menos parecido al nuestro, ya que usa los datos del día anterior es Zang et al. (2018). En esta publicación también se menciona un problema recurrente en este tipo de modelos, y es que es frecuente la falta de datos en determinados instantes. El enfoque que siguen es reconstruir los datos que faltan mediante métodos estadísticos y entrenar el modelo con ellos.

Un enfoque más sofisticado a este problema lo presenta Dong et al. (2022), que encadena la integración de Laplace, para completar los datos que faltan, con una red profunda.

## 2.2 Variables a utilizar

Respecto a las variables a utilizar para realizar las predicciones, se han hecho intentos de incluir otros tipos de información diferente a la temporal y al tráfico para realizar las predicciones. Por ejemplo: datos sobre accidentes o información meteorológica. En Li and Chen (2012) se realizan experimentos para evaluar la importancia de variables como la velocidad, la detección de vehículos pesados, la precipitación y los datos históricos para predecir el tiempo de viaje en una vía. El artículo concluye que los datos de precipitación pueden mejorar las predicciones a 5 minutos, haciendo el modelo más robusto. en este trabajo utilizaremos este hecho para probar las variables meteorológicas en algunos de los modelos, como se explica en 3.2.

## 2.3 Variable objetivo

A la hora de modelar o predecir el tráfico en un punto o en un conjunto de puntos, es importante elegir bien la variable objetivo. Por ejemplo, si queremos predecir la cantidad de coches que van

a pasar por un punto o su velocidad media.

En Dougherty and Cobbett (1997) ya se señala que seleccionar la velocidad media como variable objetivo puede ser problemático, ya que coches aislados viajando a baja velocidad distorsionan los datos y dificultan el aprendizaje, especialmente en situaciones de poco tráfico. Sin embargo, en Samoili and Dumont (2012) se observa que la velocidad o el tiempo de viaje son variables más favorables a la hora de proveer información a los conductores. En Wang et al. (2017) se extraen patrones de comportamiento de tráfico, usando una nueva variable llamada *Índice de congestión*, que relaciona la cantidad de vehículos con su velocidad media.

# Capítulo 3

# Análisis explorativo de los datos

En el trabajo utilizamos dos tipos de datos públicos: los de tráfico y los meteorológicos.

También se han utilizado datos de los días festivos y vacaciones escolates en los últimos tres años, así como de las fechas del estado de alarma, pero su cantidad es despreciable, en comparación con los otros dos tipos de datos mencionados.

## 3.1 Datos de tráfico

### 3.1.1 Origen de los datos

El Ayuntamiento de Madrid dispone actualmente de 4609 detectores de vehículos, que forman parte de diversos sistemas de control de tráfico en la ciudad. Los datos recogidos por los detectores se registran en la base de datos SICTRAM (Sistema Integral de Control de Tráfico de Madrid) y se integran sobre períodos de 15 minutos. A su vez, estos datos se publican con periodicidad mensual en formato CSV en el Portal de datos abiertos del Ayuntamiento de Madrid.

En ocasiones, los sensores pueden tener errores, lo que impide la publicación de datos en determinados instantes. En otras ocasiones, los sensores funcionan, pero los parámetros de calidad de la información recogida no son óptimos; en este caso, esta se registra y publica, pero indicando la posibilidad de presencia de error.

La distribución de los 4609 sensores en la ciudad se puede ver en la figura 3.1. La visualización se ha realizado usando Folium python visualization (2020) y GeoPandas (Jordahl et al. (2020)).

Además, es importante comentar que estos sensores detectan los vehículos en una sola dirección. De hecho, en muchos casos, hay dos detectores a la misma altura de la calle, pero que se encargan de direcciones contrarias.

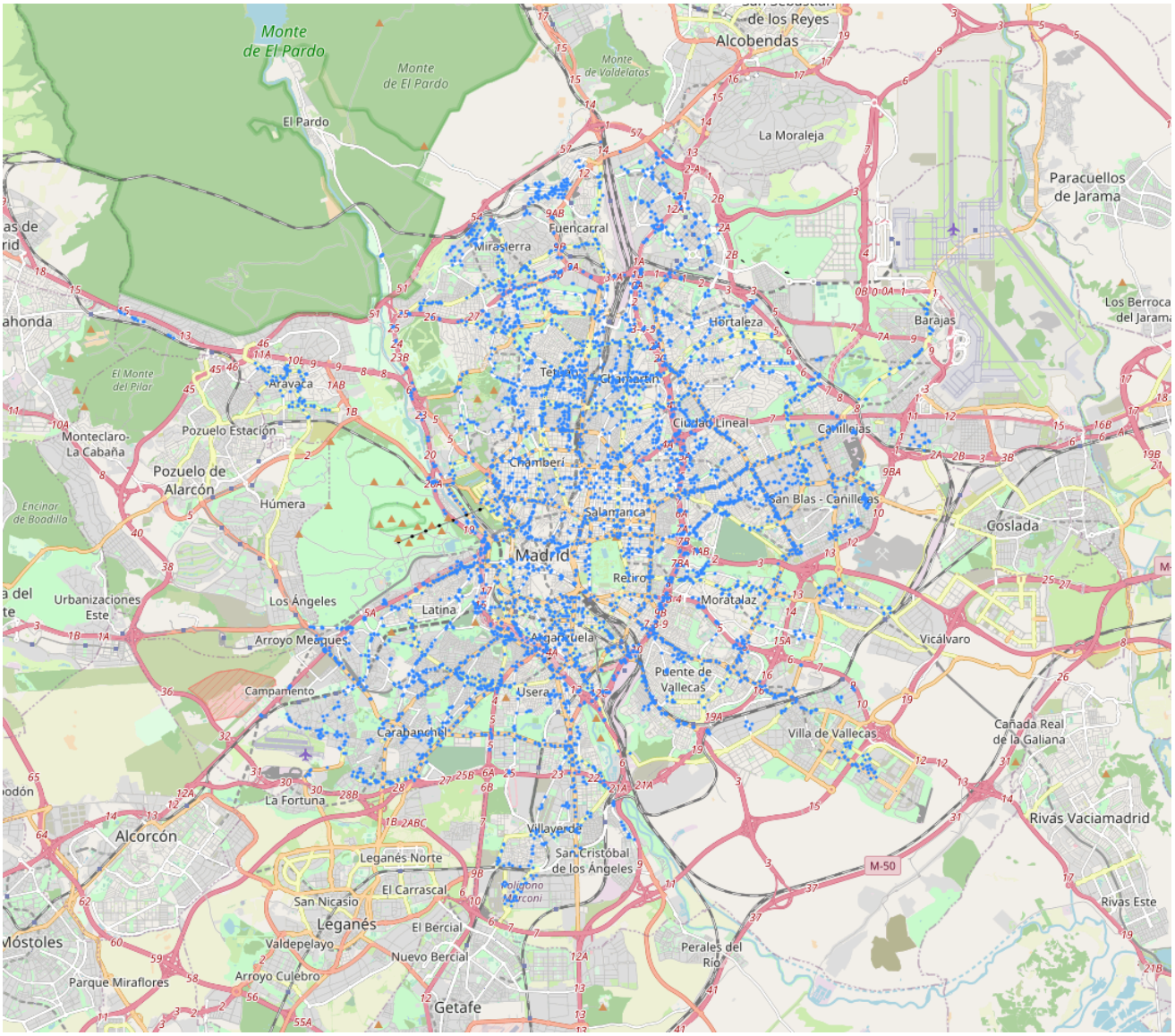


Figure 3.1: Distribución de todos los sensores de tráfico en la ciudad de Madrid

### 3.1.2 Magnitudes de tráfico

Las magnitudes publicadas en dicha fuente son las siguientes:

1. **Intensidad**: Vehículos por hora en el punto de medida en el periodo de 15 minutos.
2. **Ocupación**: Tiempo de ocupación del punto de medida en el periodo de 15 minutos, en porcentaje.
3. **Velocidad media**: Velocidad media de los vehículos en el periodo de 15 minutos, en Km./h.

### 3.1.3 Filtrado de datos

En la selección de datos para este trabajo, se han seguido los siguientes criterios:

1. Se han descargado todos los datos correspondientes a 2019, 2020 y 2021 del apartado *Tráfico. Histórico de datos del tráfico desde 2013* del Portal de datos abiertos del Ayuntamiento de Madrid [1].
2. Se han descartado todos registros en los que el Ayuntamiento de Madrid, como se explica en 3.1.1, ha indicado que existe la posibilidad de error.
3. Además, se detectó que, incluso tras eliminar los registros con posibilidad de error, hay ocasiones en los que los valores de tráfico se mantienen constantes durante un período largo del tiempo. En este trabajo se ha considerado que, si el valor de un parámetro de tráfico en un sensor se mantiene constante durante más de cuatro períodos (es decir, una hora, ya que cada período es de 15 minutos), entonces el valor es erróneo y, por lo tanto, se descarta.
4. Para poder aplicar algunos de los modelos que utilizaremos en este trabajo, vamos a necesitar también continuidad en la disponibilidad de los datos. Es decir, vamos a necesitar que las observaciones estén dentro de secuencias de al menos $T' + T$ observaciones seguidas (donde, como se explica en 1.2, $T'$ es el número de observaciones previas a $t$ que necesitamos para la predicción, y $T$ es el número de instantes posteriores a $t$ en los que realizamos la predicción), cada 15 minutos. Cuanto más alto sea $T' + T$, mayor será el número de observaciones que no podremos usar para entrenar el modelo.

### 3.1.4 Exploración de los datos de tráfico

La disponibilidad de los datos y, tras realizar el filtrado indicado en 3.1.3 es variable a lo largo de los tres años. En la figura 3.2, se muestra un resumen de las magnitudes descritas en 3.1.2 en

[1] https://datos.madrid.es/portal/site/egob

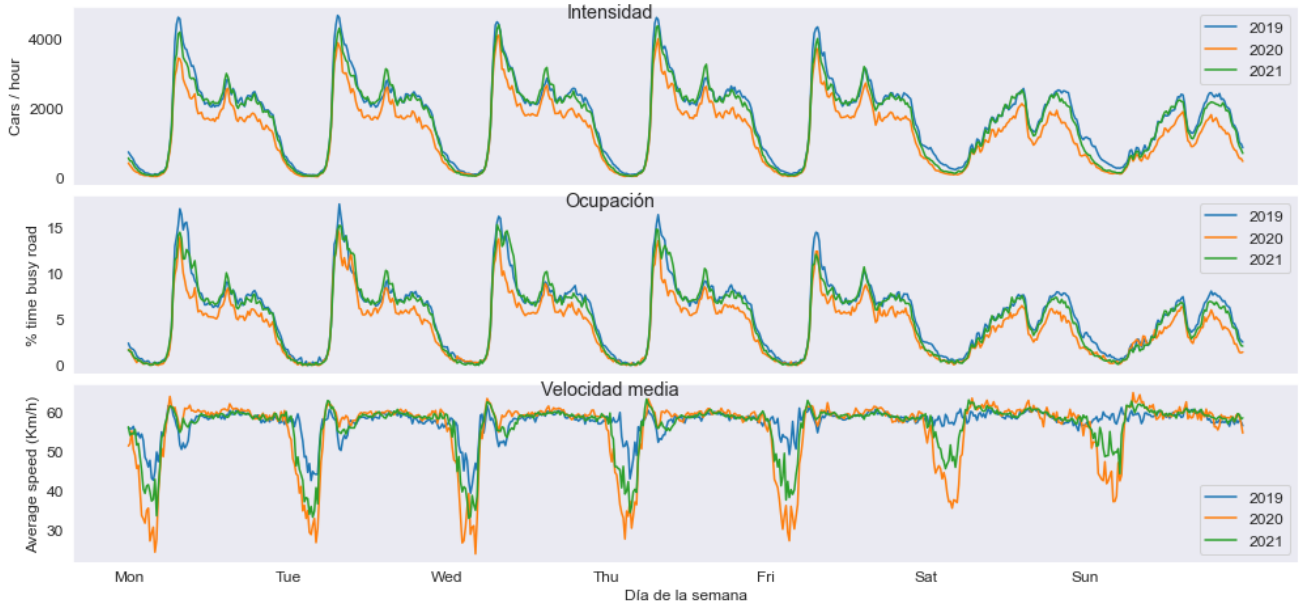


Figure 3.2: Análisis de las magnitudes de tráfico a lo largo de los años 2019, 2020 y 2021 en el punto de medida 1001.
Se indica la media de la magnitud correspondiente por día de la semana y hora (en períodos de 15 minutos), para cada año.

un sensor concreto de la ciudad (con identificador 1001), en el barrio de La Latina. Para cada magnitud, se indica el valor medio por día de la semana y hora (en intervalos de 15 minutos), en cada año.

Si hacemos un recuento de las observaciones disponibles a lo largo del tiempo, obtenemos la figura 3.3. De ella podemos extraer las siguientes conclusiones:

1. La disponibilidad de datos es relativamente estable a lo largo del tiempo, exceptuando los meses a mediados del año 2020, donde se observa un descenso.

2. Hay aproximadamente diez veces más datos sobre intensidad (coches por hora) que de velocidad media. El número de datos sobre ocupación (porcentaje del tiempo en que una vía está ocupada) es algo menor que el de intensidad, pero también mucho más alto que el de velocidad media.

### 3.1.5 Discusión sobre la variable objetivo a utilizar

La elección de una variable objetivo entre las explicadas en 3.1.2 es el primer problema al que nos enfrentamos. En 2.3 se han mencionado algunas ventajas y desventajas que se han discutido en la literatura respecto a las diferentes variables objetivo.


Figure 3.3: Número de observaciones para el total de los puntos de medida de cada magnitud de tráfico a lo largo de los años 2019, 2020 y 2021.

Debido a la baja disponibilidad, descartamos primero la velocidad media como variable objetivo. Tenemos más datos sobre intensidad, pero la ocupación presenta una gran ventaja, y es que es un porcentaje. El hecho de que tenga el límite en 100 va a facilitar mucho el entrenamiento de los modelos, así como su evaluación y comparación. Por esta razón, elegimos la **ocupación** como variable objetivo.

### 3.1.6 Elección de los sensores

Es crítico elegir un subconjunto de sensores con gran disponibilidad de datos de nuestra variable objetivo y meteorológicas. Además, para el último modelo, basado en grafos, es importante que los diferentes sensores estén próximos entre ellos. Para elegir los más adecuados, he seguido los siguientes pasos:

1. He buscado los sensores que tienen más de 80000 observaciones entre 2019, 2020 y 2021. Esto equivale a algo más de dos años de observaciones sin interrupciones, ya que

$$\frac{80000}{4 \text{ observaciones por hora} \times 24 \text{ horas al día} \times 365 \text{días al año}} \approx 2.3$$

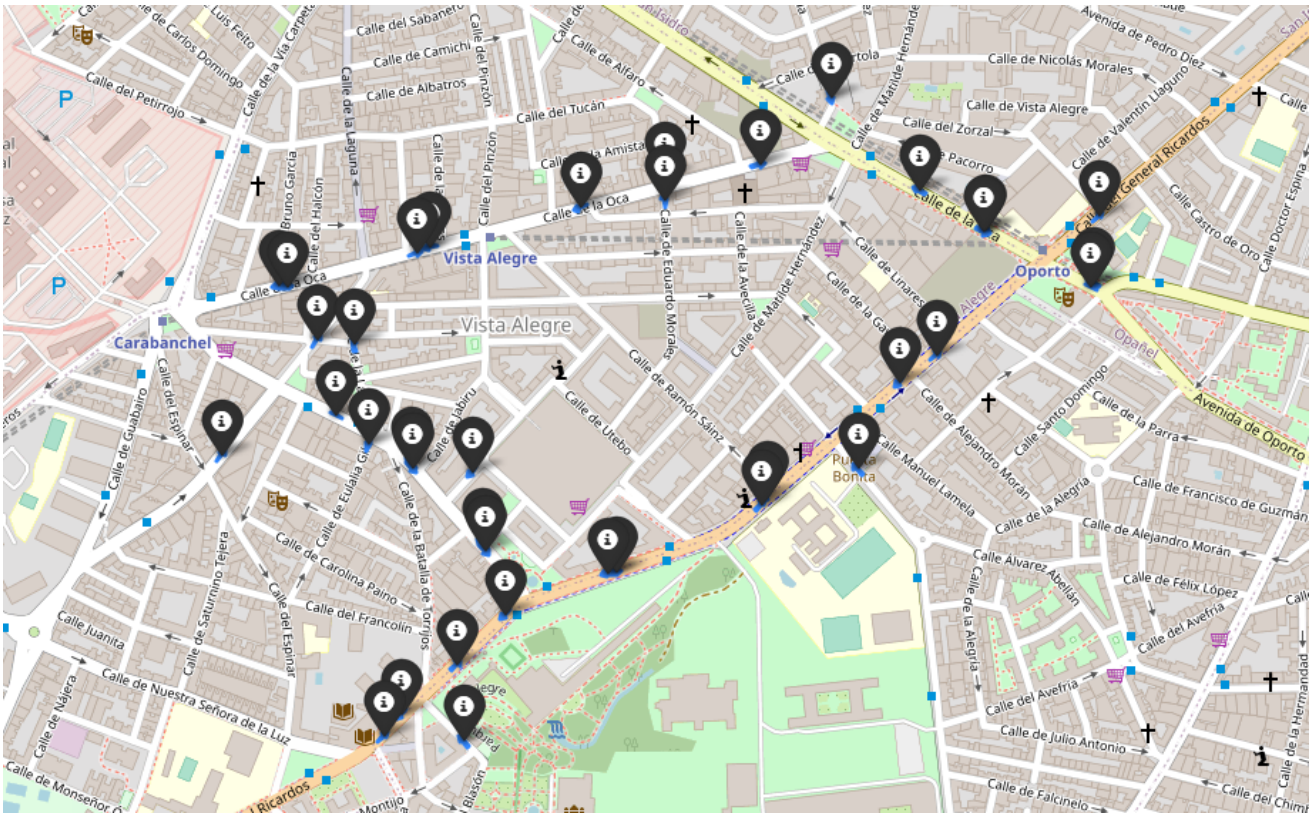


Figure 3.4: Los 37 sensores seleccionados para evaluar los modelos.
Todos ellos tienen más de 80000 observaciones entre 2019, 2020 y 2021, y se sitúan en el barrio de Carabanchel.

2. Visualizo en el mapa los 791 sensores que cumplen este requisito (del total de 4609), y observo que muchos de ellos se encuentran en el barrio de Carabanchel. Ya que es una zona urbana y con tráfico frecuente, decido elegir esta zona para evaluar los modelos.

3. En concreto, de entre los 791 sensores, selecciono aquellos que tienen coordenadas en los siguientes rangos, usando el sistema geodésico de coordenadas WGS84:

   - Longitud entre $-3.744$ y $-3.7308$.
   - Latitud entre 40.3830 y 40.3904.

4. Al final, resultan 37 sensores seleccionados, que se pueden visualizar en la figura 3.4.

## 3.2 Datos meteorológicos

Aprovechamos la disponibilidad de datos meteorológicos en el Portal de datos abiertos del Ayuntamiento de Madrid ([2]) para integrarlos en algunos de nuestros modelos y analizar su

[2]https://datos.madrid.es/portal/site/egob

influencia en la predicción.

### 3.2.1 Origen de los datos

La red meteorológica municipal de Madrid tiene 26 estaciones de control repartidas por la geografía de la ciudad. Estas estaciones recogen las magnitudes mencionadas en 3.2.2 y los registros, con frecuencia horaria, se publican mensualmente en el portal en formato CSV.

Cada registro lleva un código de validación, que indica la presencia o no de error en el valor registrado.

### 3.2.2 Magnitudes meteorológicas

Las magnitudes meteorológicas disponibles son:

1. Velocidad del viento
2. Dirección del viento
3. Temperatura
4. Humedad relativa
5. Presión barométrica
6. Radiación solar
7. Precipitación

   Cada estación de control recoge datos sobre un subconjunto de estas magnitudes

### 3.2.3 Filtrado de datos

El filtrado de datos es similar al explicado para los datos de tráfico en 3.1.3, con una diferencia:

Algunas de las variables meteorológicas (precipitación, radiación solar y presión barométrica) son muy constantes. Por ejemplo, el valor de la precipitación en Madrid es en la gran mayoría de los casos 0 y la radiación solar es durante toda la noche 0. Por ello, en estas variables no vamos a efxigir la condición que el valor cambie al menos cada cuatro períodos, como sí hacíamos en el caso del tráfico.

### 3.2.4 Disponibilidad de los datos meteorológicos

El número de observaciones por magnitud meteorológica y semana se puede ver en la figura 3.5. Podemos observar que la disponibilidad es estable a lo largo del tiempo.

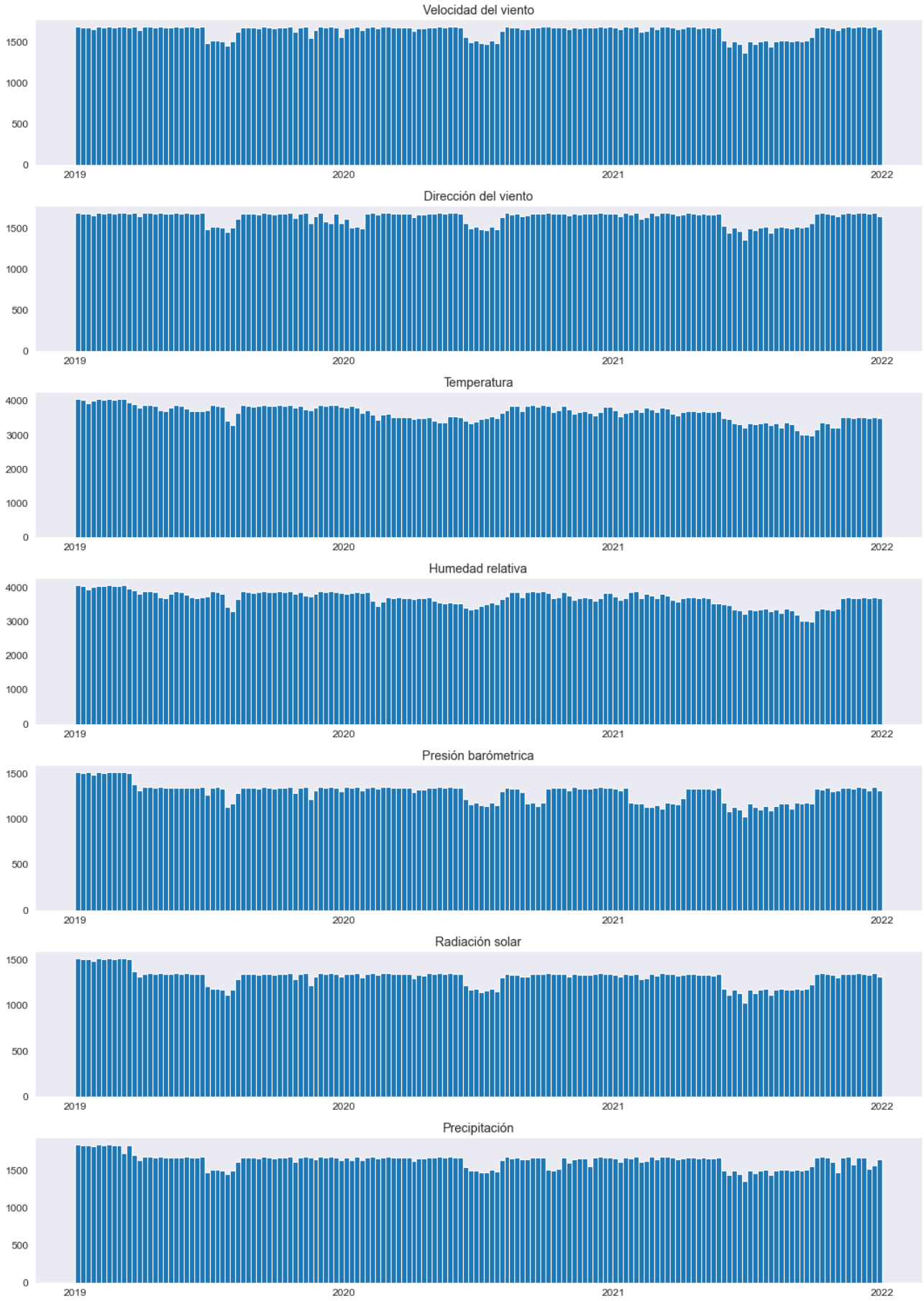


Figure 3.5: Número de observaciones para el total de los estaciones de medida de cada magnitud meteorológica a lo largo de los años 2019, 2020 y 2021.

## 3.3 Combinación de los datos

Como veremos más adelante, algunos modelos usarán como variables explicativas el tráfico en instantes anteriores, otros usarán algunas variables meteorológicas, y otros usarán solo variables temporales para predecir el tráfico.

A la hora de consultar los datos para un modelo concreto, el actual código realiza los siguientes pasos:

1. Extrae la columna correspondiente a nuestra variable objetivo de tráfico, en el rango de fechas elegido.

2. Extrae las columnas de datos meteorológicos, si alguna, que necesitemos para el modelo, en el mismo rango de fechas. Dado que no todas las estaciones recogen todas las magnitudes meteorológicas, se extraen las columnas de las estaciones más cercanas para cada magnitud.

3. Filtra los datos de tráfico y meteorológicos según los pasos explicados en 3.1.3 y 3.2.3.

4. Combina todos los datos en una sola tabla donde el índice es la fecha y hora, en períodos de 15 minutos.

5. Como las variables meteorológicas tienen frecuencia horaria, se realiza una interpolación lineal para completar las filas cada 15 minutos.

6. Si, para algún instante, alguna de las columnas tiene valor nulo, entonces descarta ese instante.

7. Crea nuevas columnas que recojan información temporal: año, estación, mes, número de semana, día del mes, tipo de día (lunes-viernes, sábado o domingo), día de la semana, si es festivo, si es un día laborable, si hay vacaciones escolares, la hora (con 15 minutos de exactitud) y el minuto dentro de la hora (es decir, si es en punto, y cuarto, y media o menos cuarto).

Para seleccionar qué estación de control utilizar para una variable meteorológica concreta, originalmente realicé una correspondencia entre detectores de vehículos y estaciones meteorológicas más cercanas para cada magnitud. De esta manera, la idea es que, para predecir el tráfico en un punto, se usen unos datos meteorológicos y, para otro punto, se usen otros datos meteorológicos. Sin embargo, esto causaba los siguientes problemas:

- En algunos casos, se puede observar que estaciones relativamente cercanas recogen observaciones bastante diferentes.

- Como queremos evaluar el modelo en varios sensores en los mismos instantes, el hecho de que estos dependan de la disponibilidad de los datos en estaciones diferentes crea más problemas en la disponibilidad.

Los sensores que hemos seleccionado en 3.1.6 están todos en una zona limitada y la mayoría, de hecho, corresponde (según la correspondencia que se hizo al principio) a las misma estaciones meteorológicas. Por ello, y para facilitar el proceso, asigno las mismas estaciones a todos los sensores seleccionados.

## 3.4 Dificultades respecto a la disponibilidad de datos

Algunos modelos, como por ejemplo, el de repetición en 6.1.3.1, necesitan los datos de tráfico sin interrupciones en los $T'$ instantes anteriores a $t$ para poder realizar la predicción. Si queremos realizar predicciones para varios sensores en un instante $t$, entonces necesitamos que todos los sensores tengan datos disponibles en los $T'$ instantes anteriores a $t$. Y, si queremos comparar las predicciones con los valores verdaderos, entonces necesitamos tener para todos los sensores, estos valores en los $T$ instantes siguientes.

Además, en el caso de modelos que usan variables meteorológicas, también necesitaremos esta información para todos los sensores.

Cuanto mayor es el número de sensores en los que queremos evaluar el modelo y, cuanto mayores son $T$ y $T'$, más dificultades tendremos para encontrar un rango de tiempo en el que poder entrenar nuestro modelo.

# Capítulo 4

# Metodología para el entrenamiento y la evaluación de modelos

## 4.1 Datos de entrenamiento y de test

Utilizaremos los datos de tráfico y meteorológicos de 2019, 2020 y 2021, tras el filtrado explicado en 3.1.3 y 3.2.3.

Dado que estamos tratando con datos a lo largo del tiempo, es importante que los datos de entrenamiento sean previos a los datos de test. Aproximadamente el primer 80% de datos los usaremos para entrenamiento (desde el 1 de enero de 2019 hasta el 31 de mayo de 2021) y, el 20% restante (desde el 1 de junio de 2021 hasta el 31 de diciembre del mismo año), para test.

En concreto, el primer 80% de los datos los usaremos para seleccionar modelos y ajustar parámetros. Para dar una puntuación final a los modelos usaremos la estrategia de validación cruzada explicada en 4.2.2

## 4.2 Validación cruzada

La selección de los modelos y su entrenamiento los realizaremos con validación cruzada adaptada a series temporales, basada en Hyndman and Athanasopoulos (2021c). Los conjuntos de test se componen de las observaciones durante una mes, y su correspondiente conjunto de entrenamiento se compone solo de observaciones que ocurrieron anteriormente a ese mes, como se ilustra en la figura 4.1. De esta manera, nos aseguramos de que las observaciones futuras no van a influir en la predicción. La puntuación de un modelo se calculará como la media de la puntuación en todas las observaciones de test.

Vamos a utilizar dos variantes de este método:

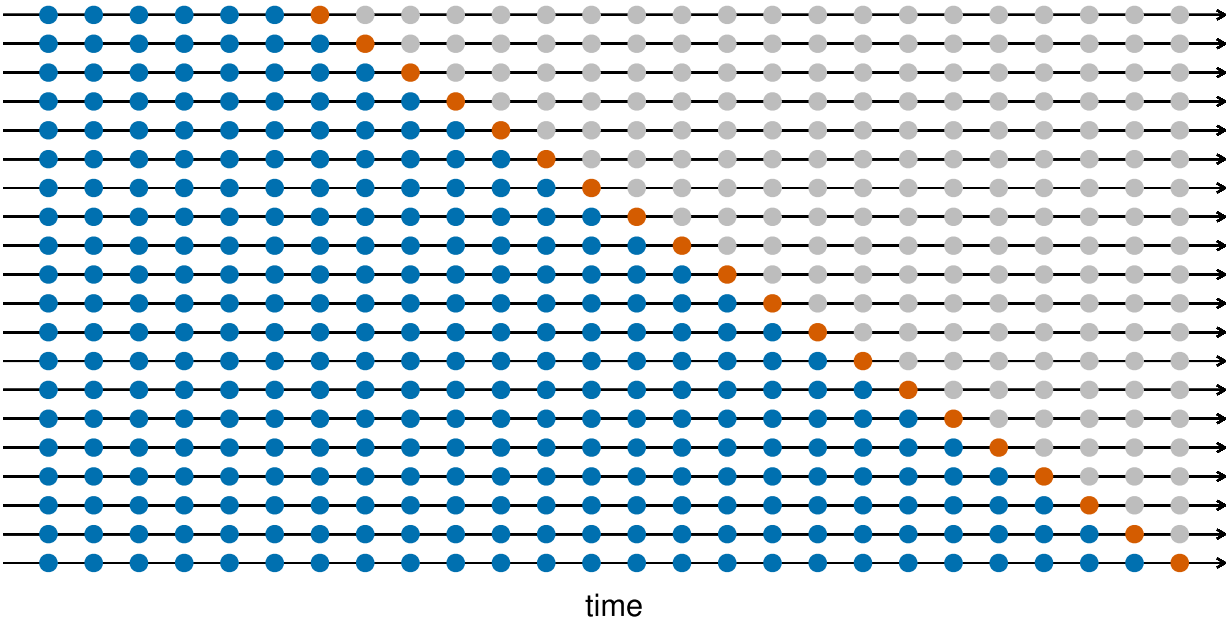


Figure 4.1: Validación cruzada en series temporales
El diagrama ilustra las series de conjuntos de entrenamiento y de test. Los puntos azules son observaciones de entrenamiento y, los naranjas, observaciones de test. En nuestro método, cada punto se compone de todas las observaciones de un mes.
Fuente: Hyndman and Athanasopoulos (2021c)

### 4.2.1 Selección de modelos

Cuando seleccionamos modelos y ajustamos parámetros, dado que se prueban muchas configuraciones diferentes, es conveniente reducir el tiempo de computación. Por ello, en este caso dividimos el conjunto de entrenamiento en solo 4 grupos. Realizaremos entrenamiento y validación tres veces:

1. La primera, el conjunto de entrenamiento será el grupo 1 y, el de test, el 2.
2. La segunda, los grupos 1 y 2 compondrán el conjunto de entrenamiento y, el 3, el de test.
3. En la tercera y última vez, entrenaremos el modelo con los datos de los grupos 1, 2 y 3, y el 4 será el conjunto de test.

Entrenando el modelo solo tres veces para una configuración, conseguimos finalizar más rápido y probar más parámetros en menos tiempo.

### 4.2.2 Evaluación de modelos

En la evaluación final de los diferentes modelos en 7 utilizaremos el método descrito en 4.2 sin modificaciones. Para que sirva como ejemplo: empezamos entrenando con todas las observaciones hasta mayo de 2021 y evaluamos en junio de 2021. A continuación, entrenamos con todas

las observaciones hasta junio de 2021 y evaluamos en julio de 2021. Así, continuamos hasta diciembre del mismo año.

Vamos a utilizar el error absoluto medio y el error cuadrático medio. Debido al procedimiento indicado, vamos a obtener una puntuación para cada mes desde junio hasta diciembre de 2021. Es decir, 7 puntuaciones. Lo que al final evaluaremos será la media de estas puntuaciones en todos los sensores para cada modelo.

# Capítulo 5

# Teoría de los modelos utilizados

En esta sección hacemos un repaso de la teoría de lo modelos que utilizamos. Algunos de los modelos (los de *baseline* o referencia) están basados en propiedades muy básicas y, por lo tanto, no necesitan un contexto teórico. Veremos la teoría de la regresión lineal, *XGBoost* y la red neuronal recurrente convolucional de difusión, que es una red neuronal de grafos (GNN).

## 5.1 Regresión lineal

### 5.1.1 Características del modelo

Reutilizando la notación de 1.2 y basándonos en Freedman (2009), podemos formular la regresión múltiple, en un punto de medida $v_i \in \mathcal{V}$ para un instante $t$, de la siguiente manera:

$$Y_{v_i}^{(t)} = X_{v_i}^{(t)} \beta_{v_i} + \epsilon_{v_i}^{(t)} \tag{5.1}$$

donde $Y_{v_i}^{(t)} \in \mathbb{R}$ es la observación de tráfico en el instante $t$ en el sensor $v_i$. Es decir, $Y_{v_i}^{(t)}$ es la variable dependiente o respuesta. $X_{v_i}^{(t)} \in \mathbb{R}^{1\times P}$ es la matriz de diseño, donde $P$ es el número de variables explicativas. Queremos estimar el vector $\beta_{v_i} \in \mathbb{R}^{P\times 1}$, que no es dependiente de $t$. La variable $\epsilon_{v_i}^{(t)} \in \mathbb{R}$ recoge el error causado por factores no observables.

Para un sensor, podemos combinar las ecuaciones como la de 5.1 para todos los instantes $t$ en una sola: Sea $H_{v_i}$ el número total de instantes en los que tenemos observaciones de entrenamiento para el sensor $v_i$. Entonces, la fórmula sería:

$$Y_{v_i} = X_{v_i} \beta_{v_i} + \epsilon_{v_i} \tag{5.2}$$

donde $Y_{v_i} \in \mathbb{R}^{H_{v_i}\times 1}$ y $X_{v_i} \in \mathbb{R}^{H_{v_i}\times P}$. Podemos suponer que $H_{v_i}$ es lo suficientemente alta como para que $H_{v_i} > P$ y que sus columnas son linealmente independientes. $\beta \in \mathbb{R}^{P\times 1}$ es el mismo vector de parámetros que en 5.1 y $\epsilon_{v_i} \in \mathbb{R}^{H_{v_i}\times 1}$ el vector de error. Podemos asumir que:

1. Los datos en $Y_{v_i}$ son observaciones de $X_{v_i}\beta_{v_i} + \epsilon_{v_i}$.

2. Los componentes de $\epsilon_{v_i}$ son independientes e igualmente distribuidos, con media 0 y varianza $\sigma^2_{v_i}$.

3. Si $X_{v_i}$ es aleatoria, $\epsilon_{v_i}$ es independiente de $X_{v_i}$.

Entonces estimamos $\beta_{v_i}$ usando los mínimos cuadrados ordinarios:

$$\hat{\beta_{v_i}} = \left(X^T_{v_i} X_{v_i}\right)^{-1} X^T_{v_i} Y_{v_i}$$

En este trabajo, usamos un enfoque empírico con validación cruzada para seleccionar las variables explicativas a usar, como se explica en 6.2.3.

### 5.1.2 Regularización

Además, utilizaremos la regresión de Ridge con validación cruzada (Rifkin and Lippert (2007)). Este método impone una penalización en el tamaño de los coeficientes de la regresión. Los coeficientes de Ridge minimizan la suma de cuadrados penalizada:

$$\min_w \left\| Xw - \hat{Y} \right\|_2^2 + \alpha \left\| w \right\|_2^2$$

El parámetro $\alpha$ controla la regularización de la regresión: cuanto mayor sea $\alpha$ más robusto será el modelo a la colinealidad.

También utilizamos validación cruzada para escoger el valor de $\alpha$ con el que se obtienen los mejores resultados, entre 0.1, 1 y 10.

## 5.2 XGBoost

El contenido de este apartado se ha extraído casi en su totalidad de Chen and Guestrin (2016), que es el artículo que introdujo este algoritmo. Sin embargo, vamos a continuar con la notación ya introducida en este trabajo, y no con la del artículo del algoritmo. Es importante remarcar que entrenaremos un modelo para cada uno de los 37 sensores en la red. Fijamos un sensor $v \in \mathcal{V}$, y suponemos que queremos entrenar un modelo XGBoost para dicho sensor.

XGBoost es un algoritmo de aprendizaje conjunto o *ensemble learning*, basado en los árboles de regresión y la potenciación de gradiente. La idea del aprendizaje conjunto es entrenar una cantidad de modelos o aprendices débiles que, combinados, forman un clasificador más fuerte.

Basándose en las ideas de Friedman (2001), Chen and Guestrin (2016) introdujeron modificaciones en la función objetivo y en el tratamiento de valores nulos, que resultaron en la versión mejorada del algoritmo: XGBoost.

### 5.2.1 Función objetivo regularizada

Sea $H_v$ el número de instantes en los que tenemos los datos de las variables explicativas y respuesta: $\mathcal{D} = \{(X_v^{(t)}, Y_v^{(t)})\}(|\mathcal{D}| = H_v, X_v^{(t)} \in \mathbb{R}^{1\times P}, Y_v^{(t)} \in \mathbb{R})$, un modelo de aprendizaje conjunto de árboles, o *tree ensemble method*, utiliza $K$ funciones aditivas para predecir la variable respuesta:

$$\hat{Y_v}^{(t)} = \phi(X_v^{(t)}) = \sum_{k=1}^{K} f_k\left(X_v^{(t)}\right), \quad f_k \in \mathcal{F} \tag{5.3}$$

donde $\mathcal{F} = \{f(X) = w_{q(x)}\} \quad (q : \mathbb{R}^{1\times P} \to B, w \in \mathbb{R}^B)$ es el espacio de árboles de regresión. En esta expresión, $B$ es el número de hojas en el árbol y, por tanto, $q$ crea la correspondencia entre un elemento de los datos y la hoja. Cada $f_k$ se corresponde con una estructura de árbol independiente $q$ y los pesos $w$. La diferencia entre estos árboles de regresión y los árboles de decisión, es que los primeros tienen una puntuación continua en cada hoja $i$: $w_i$. Dado un ejemplo de nuestros datos, utilizamos las reglas de decisión en los árboles (dadas por $q$) para colocarlo en una hoja y calcular la predicción final, sumando la puntuación de las hojas correspondientes (dadas por $w$). Para aprender las funciones usadas en el modelo, minimizamos el objetivo *regularizado*:

$$\mathcal{L}(\phi) = \sum_t l(\hat{Y_v}^{(t)}, Y_v^{(t)}) + \sum_k \Omega(f_k) \tag{5.4}$$

donde $\Omega(f) = \gamma B + \frac{1}{2}\lambda \|w\|^2$.

En la ecuación 5.4, $l$ es una función de pérdida diferenciable y convexa, que cuantifica la diferencia entre la predicción $\hat{Y_v}^{(t)}$ y la variable objetivo $Y_v^{(t)}$. El segundo término, $\Omega$, penaliza la complejidad del modelo. De esta manera, obtendremos unos pesos más regulares, lo que ayudará a reducir el sobreajuste u *overfitting*.

Con este planteamiento, podemos esperar que el método escoja funciones simples. Cuando los parámetros de regularización ($\gamma$ y $\lambda$) se anulan, entonces el método es igual que la potenciación de gradiente tradicional.

### 5.2.2 Potenciación de gradiente

Dado que el modelo en la ecuación 5.4 incluye funciones como parámetros, no se puede optimizar con los métodos tradicionales en el espacio euclídeo. En su lugar, el modelo se entrena de manera aditiva. Sea $\hat{Y_v}^{(t)}(i)$ la predicción en el instante $t$ en la iteración $i$. Entonces, vamos a sumar $f_i$ para minimizar el siguiente objetivo:

$$\mathcal{L}^{(i)} = \sum_{t=1}^{H_v} l\left(Y_v^{(t)}, \hat{Y_v}^{(t-1)}(i) + f_i(X_v^{(t)})\right) + \Omega(f_i)$$

Es decir, de manera *greedy*[1] sumamos la función $f_i$ que más mejora el modelo, de acuerdo

[1]Un algoritmo *greedy* es aquel que elige la opción óptima en cada paso local, con el objetivo de llegar a una

con la ecuación 5.4. También se puede utilizar la aproximación de segundo orden (Friedman et al. (2000)) para optimizar más rápido la función objetivo:

$$\mathcal{L}^{(i)} \simeq \sum_{i=1}^{n} \left[ l(Y_v^{(t)}, \hat{Y}_v^{(t-1)}(i)) + g^{(t)} f_i(X_v^{(t)}) + \frac{1}{2} h^{(t)} f_i^2(X_v^{(t)}) \right] + \Omega(f_i)$$

donde $g^{(t)} = \delta_{\hat{Y}_v(i-1)} l\left(Y_v^{(t)}, \hat{Y}_v{}^t(i-1)\right)$ y $h^{(t)} = \delta^2_{\hat{Y}_v(i-1)} l\left(Y_v^{(t)}, \hat{Y}_v{}^t(i-1)\right)$ son los estadísticos de primer y segundo orden de la función pérdida. Si obviamos los términos constantes, obtenemos la siguiente simplificación de la función objetivo en el paso $t$:

$$\tilde{\mathcal{L}}^{(i)} = \sum_{i=1}^{n} \left[ g^{(t)} f_i(X_v^{(t)}) + \frac{1}{2} h^{(t)} f_i^2(X_v^{(t)}) \right] + \Omega(f_i) \tag{5.5}$$

Sea $I_j = \{t \| q(X_v^{(t)}) = j\}$ el conjunto de instantes cuyo valor de la variable respuesta están en la hoja $j$. Podemos reescribir la ecuación 5.5, desarrollando $\Omega$, de la siguiente manera:

$$\begin{aligned} \tilde{\mathcal{L}}^{(i)} &= \sum_{i=1}^{n} \left[ g^{(t)} f_i(X_v^{(t)}) + \frac{1}{2} h^{(t)} f_i^2(X_v^{(t)}) \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{B} w_j^2 \\ &= \sum_{j=1}^{T} \left[ \left( \sum_{t \in I_j} g^{(t)} \right) w_j + \frac{1}{2} \left( \sum_{t \in I_j} h^{(t)} + \lambda \right) w_j^2 \right] + \gamma B \end{aligned} \tag{5.6}$$

Sea $q(x)$ una de las estructuras de árbol, fija. entonces, podemos calcular el peso óptimo $w_j^\star$ para la hoja $j$ como

$$w_j^\star = -\frac{\sum_{t \in I_j} g^{(t)}}{\sum_{t \in I_j} h^{(t)} + \lambda} \tag{5.7}$$

y calcular el valor óptimo correspondiente como

$$\tilde{\mathcal{L}}^{(i)}(q) = -\frac{1}{2} \sum_{j=1}^{B} \frac{\left( \sum_{t \in I_j} g^{(t)} \right)^2}{\sum_{t \in I_j} h^{(t)} + \lambda} + \gamma B. \tag{5.8}$$

La ecuación 5.8 se puede usar como puntuación para estimar la calidad de la estructura de un árbol $q$, donde un valor más bajo se corresponde con mayor calidad de la estructura de árbol.

Como normalmente es imposible enumerar todas las estructuras de árbol posibles, se sigue un algoritmo *greedy*, de manera que se empieza con una sola hoja y, de manera iterativa, se van añadiendo ramas al árbol. Sean $I_L$ e $I_R$ os conjuntos de ejemplos a izquierda y derecha, respectivamente, después de una división de una hoja. Si $I = I_L \cup I_R$, entonces la reducción de

solución general óptima.

la pérdida después de la división esta dada por

$$\mathcal{L}_{\text{división}} = \frac{1}{2}\left[\frac{\left(\sum_{t\in I_L} g^{(t)}\right)^2}{\sum_{t\in I_L} h^{(t)}+\lambda} + \frac{\left(\sum_{t\in I_R} g^{(t)}\right)^2}{\sum_{t\in I_R} h^{(t)}+\lambda} - \frac{\left(\sum_{t\in I} g^{(t)}\right)^2}{\sum_{t\in I} h^{(t)}+\lambda}\right] - \gamma \tag{5.9}$$

Esta fórmula se utiliza en la práctica para evaluar los candidatos a división.

### 5.2.3 *Shrinkage* y submuestreo de las variables explicativas

Además de la función objetivo regularizada, explicada en 5.2.1, Chen and Guestrin (2016) introdujeron dos técnicas adicionales para prevenir el sobreajuste:

1. El *shrinkage* o encogimiento escala los nuevos pesos con un factor $\eta$, después de cada paso de la potenciación del gradiente. De esta manera, se reduce la influencia de cada árbol individual y deja espacio para que futuros árboles mejoren el modelo

2. El submuestreo o *subsampling* de variables explicativas se utiliza en los algoritmos de bosques aleatorios (Breiman (2001)), y reduce el sobreajuste, así como la carga computacional.

### 5.2.4 Algoritmos de búsqueda de las divisiones

El artículo de Chen and Guestrin (2016) menciona dos algoritmos: el exacto *greedy* y el de aproximación. En este trabajo, se utiliza el primero, por lo que no explicamos el segundo:

**Algoritmo 1** Algoritmo *greedy* para la búsqueda de divisiones Chen and Guestrin (2016)

**Input**: $I$ ▷ Conjunto de ejemplos en el nodo actual
**Input**: $d$ ▷ Dimensión de las variables explicativas
puntuación $\leftarrow 0$
$G \leftarrow \sum_{t\in I} g^{(t)}$, $H \leftarrow \sum_{t\in I} h^{(t)}$
**for** $k = 1$ to P **do** ▷ Recorrer las variables explicativas
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  **for** $t$ in sorted($I$, by $x^{(t)}$) **do** ▷ Recorrer los ejemplos (instantes) en $I$
    $G_L \leftarrow G_L + g^{(t)}$, $H_L \leftarrow H_L + h^{(t)}$
    $G_R \leftarrow -G_L$, $H_R \leftarrow H - H_L$
    puntuación $\leftarrow \max\left(\text{puntuación}, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$
  **end for**
**end for**
**Output**: División con la máxima puntuación

### 5.2.5 Búsqueda cuando los datos son dispersos

En el artículo también se introduce un método para aprender de datos con valores nulos, pero no hemos aprovechado esta característica en el trabajo, por lo que la dejamos sin explicar.

## 5.3 Red neuronal recurrente convolucional de difusión en grafos (GNN)

### 5.3.1 Planteamiento del problema en un grafo

Las redes neuronales de grafos o *Graph Neural Networks (GNN)* se propusieron en 2009 en Scarselli et al. (2009), con el objetivo de extender las redes neuronales clásicas, de manera que utilicen la información estructural de los grafos (nodos y aristas).

Las redes neuronales de grafos pueden resolver dos tipos de tareas:

1. Tareas centradas en el grafo: Este tipo de modelos son clasificadores o regresores que se aplican sobre un grafo entero.

2. Tareas centradas en los nodos: Estos modelos son clasificadores o regresores de nodos y, por lo tanto, dependen de las propiedades y aristas de cada nodo en particular.

Nuestro problema es una tarea centrada en nodos, ya que queremos predecir el tráfico para cada nodo en la red por separado.

Utilizando la notación de Li et al. (2017), representamos la red de sensores mediante un grafo dirigido $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, donde $\mathcal{V}$ es el conjunto de nodos, $|\mathcal{V}| = N$, $\mathcal{E}$ es el conjunto de aristas y $\mathbf{W} \in \mathbb{R}^{N \times N}$ es la matriz de adyacencia con pesos, que representa la proximidad entre los nodos.

Continuando con la notación de 1.2, sea $Y^{(t)} \in \mathbb{R}^N$ el tráfico observado en el instante $t$ en la red de sensores $\mathcal{G}$, y sea $X^{(t)} \in \mathbb{R}^{N \times P}$ la matriz de variables diferentes al tráfico en el instante $t$. Entonces el objetivo consiste en aprender una función $h(\cdot)$ como en la ecuación 1.1. Es decir, buscamos un modelo encuadrado dentro de los dependientes de los instantes anteriores, explicados en 1.2.

### 5.3.2 Red neuronal recurrente con proceso de convolución de difusión

El método que utilizaremos estará basado en el descrito en Li et al. (2017). La implementación que utilizamos es una modificación de Eric (2021), que utiliza la librería Deep Graph Library (DGL, Wang et al. (2019)) y Pytorch (Paszke et al. (2019)). El artículo también tiene una implementación original, pero no es la que estamos utilizando en este trabajo.

Este método modela la dependencia espacial y la dependencia temporal de dos maneras diferentes:

#### 5.3.2.1 Dependencia espacial

El método modela la dependencia espacial relacionando el flujo del tráfico con un proceso de difusión (Ibe (2013)), un tipo de proceso de Markov. Este proceso de difusión se caracteriza por un camino aleatorio en la red $\mathcal{G}$ con matriz de transición $D_O^{-1}\mathbf{W}$, donde $D_O = \text{diag}(\mathbf{W1})$ es la matriz de grado *hacia fuera* o de *outdegree* y $\mathbf{1} \in \mathbb{R}^N$ es el vector unidad. Después de muchos pasos, un proceso de Markov de tales características converge a una distribución estacionaria $\mathcal{P} \in \mathbb{R}^{N\times N}$, donde la fila $i$, $\mathcal{P}_i \in \mathbb{R}^N$ representa la probabilidad de difusión desde el nodo $v_i \in \mathcal{V}$ o, dicho de otra manera, la proximidad desde el nodo $v_i$ hacia todos los demás nodos.

En lugar de calcular la solución estacionaria, el autor propone una operación de difusión de convolución sobre las características extraídas del grafo, $X \in \mathbb{R}^{N\times P}$, y un filtro $f_\theta$:

$$X_{:,p} \star_{\mathcal{G}} f_\theta = \sum_{k=0}^{K-1} \left( \theta_{k,1} \left( D_O^{-1}\mathbf{W} \right)^k + \theta_{k,2} \left( D_I^{-1}\mathbf{W}^T \right)^k \right) X_{:,p} \tag{5.10}$$

donde $\theta \in \mathbb{R}^{K\times 2}$ son los parámetros para el filtro y $D_O^{-1}\mathbf{W}$ y $D_I^{-1}\mathbf{W}^T$ son las matrices de transixión del proceso de difusión y el opuesto.

Una vez definida la operación de convolución en la ecuación 5.10, poodemos construir una capa convolucional que transforme $P$ características en vectores de dimensión $Q$. Sea $\Theta \in \mathbb{R}^{Q\times P\times K\times 2} = [\theta]_{q,p}$, donde $\Theta_{q,p,:,:} \in \mathbb{R}^{K\times 2}$ parametriza el filtro convolucional para la característica de entrada $p$ y de salida $q$. Entonces, la capa convolucional se pueda escribir como:

$$\mathcal{H}_{:,q} = a \left( \sum_{p=1}^{P} X_{:,p} \star_{\mathcal{G}} f_{\theta_{q,p,:,:}} \right) \tag{5.11}$$

donde $X \in \mathbb{R}^{N\times P}$ son las variables de entrada, $\mathcal{H} \in \mathbb{R}^{N\times Q}$, las de salida, $\{\theta_{q,p,:,:}\}$ son los filtros y $a$ es la función de activación

#### 5.3.2.2 Dependencia temporal

El método modela la dependencia temporal con redes neuronales recurrentes (RNN); en concreto, usa unidades recurrentes cerradas (GRU) (Cho et al. (2014)). Estas unidades GRU utilizan normalmente multiplicación matricial, la cual sustituiremos por la convolución de difusión de la ecuación 5.10. El autor del método llama por consiguiente a este tipo de unidades

*unidades recurrentes cerradas con convolución de difusión* (DCGRU).

$$
\begin{aligned}
r^{(t)} &= \sigma\left(\Theta_r \star_{\mathcal{G}} \left[X^{(t)}, H^{(t-1)}\right] + b_r\right) \\
C^{(t)} &= \tanh\left(\Theta_C \star_{\mathcal{G}} \left[X^{(t)}, \left(r^{(t)} \odot H^{(t-1)}\right)\right] + b_c\right) \\
u^{(t)} &= \sigma\left(\Theta_u \star_{\mathcal{G}} \left[X^{(t)}, H^{(t-1)}\right] + b_u\right) \\
H^{(t)} &= u^{(t)} \odot H^{(t-1)} + \left(1 - u^{(t)}\right) \odot C^{(t)}
\end{aligned}
$$

donde $X^{(t)}$ y $H^{(t)}$ denotan la entrada y salida en el instante $t$. Las puerta de *reset* y de *update* son $r^{(t)}$ y $u^{(t)}$ respectivamente. $\star_{\mathcal{G}}$ denota la difusión de convolución definida en la ecuación 5.10 y $\Theta_r, \Theta_u, \Theta_C$ son los parámetros para los filtros correspondientes. Igual que una unidad GRU, una unidad DCGRU se puede utilizar para construir capas de redes neuronales y entrenarse con *backpropagation* a través del tiempo. Para la predicción en múltiples pasos, se utiliza la arquitectura *seq2seq* (Sutskever et al. (2014)). Tanto el *encoder* como el *decoder* son redes neuronales recurrentes con DCGRU. El e*encoder* se entrena con series temporales históricas y utiliza su estado final para inicializar el *decoder*. el *decoder* se entrena utilizando observaciones reales. Sin embargo, a la hora de evaluar, las observaciones reales se sustiyen por las predicciones hechas por el propio modelo. Dado que esta discrepancia puede degradar los restultados del modelo, se ha integrado *scheduled sampling* o muestreo programdo (Bengio et al. (2015)), de manera que el modelo se entrena con una observación verdadera con probabilidad $\epsilon_i$, o con una predicción del modelo, con probabilidad $1 - \epsilon_i$, en la iteración $i$. A lo largo del proceso de entrenamiento, $\epsilon_i$ disminuye gradualmente hasta 0, para permitir que el modelo aprenda la distribución de test.

El modelo se esquematiza en la figura 5.1

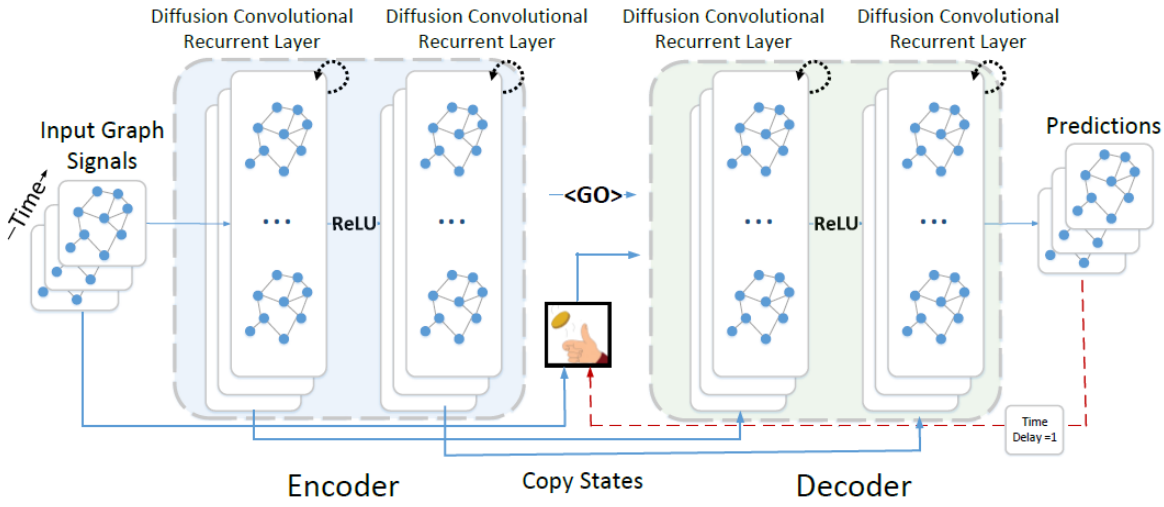


Figure 5.1: Esquema del modelo GNN con convolución de difusión
El *encoder* se entrena con la series temporales. El estado final del *encoder* se utiliza para inicializar el *decoder*. El *decoder* hace predicciones a partir de, bien las observaciones reales, o bien la salida del modelo. Fuente: Li et al. (2017).

# Capítulo 6

# Aplicación de los modelos

Como ya se ha mencionado en 1.2, vamos a ver dos tipos de modelos: aquellos dependientes de los instantes anteriores o de predicción a corto plazo, y aquellos que no dependen de los instantes anteriores, o de predicción a largo plazo. Por poner un ejemplo, un modelo dependiente de los instantes anteriores sería un modelo que dé como predicción siempre el último valor observado. En cambio, un modelo no dependiente de los instantes anteriores sería uno que dé siempre como predicción el valor medio del tráfico en ese punto. A continuación enumeramos los modelos que vamos a ver de cada tipo. El término junto a la referencia indica el nombre que tiene el modelo en las tablasy gráficos de resultados. Por ejemplo, en las tablas denominamos la primera regresión lineal como `lin-reg1`.

- **Modelos dependientes de los instantes anteriores o de predicción a corto plazo**:
  - Modelos de repetición de los últimos $T' = T$ valores (6.1.3.1, `repeat`). Lo evaluaremos para $T \in \{2, 4, 8, 16\}$.
  - Modelos de repetición del valor en $t$ (el último valor antes de la predicción), $T$ veces (6.1.3.2, , `repeat_last`). Lo evaluaremos para $T \in \{2, 4, 8, 16\}$.
  - Modelo de variación (6.1.4, , `drift`). Lo evaluaremos para $T \in \{2, 4, 8, 16\}$.
  - Red neuronal recurrente convolucional de difusión en grafos (GNN, 6.4, , `gnn_config1`, `gnn_config2`, `gnn_config3`, `gnn_config4`, `gnn_config5`). También lo evaluaremos para $T \in \{2, 4, 8, 16\}$.
- **Modelos no dependientes de los instantes anteriores o de predicción a largo plazo**:
  - Regresor de media (6.1.1.1, `mean`).
  - Regresor de mediana (6.1.1.2, `median`
  - Regresor de media por hora del día (6.1.2.1, `daytime_mean`), punto 1.

- Regresor de mediana por hora del día (6.1.2.1, `daytime_median`), punto 2.
- Regresor de media por hora del día y tipo de día (6.1.2.2, `daytime_mean_ww`, punto 1).
- Regresor de mediana por hora del día y tipo de día (6.1.2.2, `daytime_median_ww`, punto 2).
- Regresión lineal (6.2, `lin-reg1`, `lin-reg2`): Utilizaremos dos tipos de regresión lineal, cuyas variables explicativas son información del propio instante sobre el que se quiere realizar la predicción.
- Regresor XGBoost (6.3, `xgboost`). Utilizaremos las mismas variables que en la regresión lineal.

Con esto, queremos comparar los resultados de modelos más sencillos con los de los más complejos.

## 6.1 Modelos básicos de referencia

Empezaremos proponiendo algunos modelos sencillos, basados en propiedades básicas de los datos. El objetivo de esta evaluación es adquirir una idea de los resultados que se pueden obtener sin demasiadas complicaciones.

Para cada modelo, veremos un ejemplo de su aplicación en uno de los sensores durante ocho períodos. Para ello, visualizamos los valores reales durante los ocho períodos (2 horas, ya que cada período son 15 minutos) anteriores a la predicción, los valores reales en el momento de la predicción y las predicciones.

### 6.1.1 Estimador global

Los primeros modelos que probaremos se basan en un estimador. Estos modelos siempre dan la misma predicción, sea cual sea el instante $t$.

#### 6.1.1.1 Regresor de media

Para cada punto de medida, da siempre como valor de la predicción la media de los datos de entrenamiento en ese punto de medida. Véase el ejemplo en la figura 6.1.

#### 6.1.1.2 Regresor de mediana

Igual que en 6.1.1.1, pero tomando la mediana de los datos de entrenamiento. Véase el ejemplo en la figura 6.2.

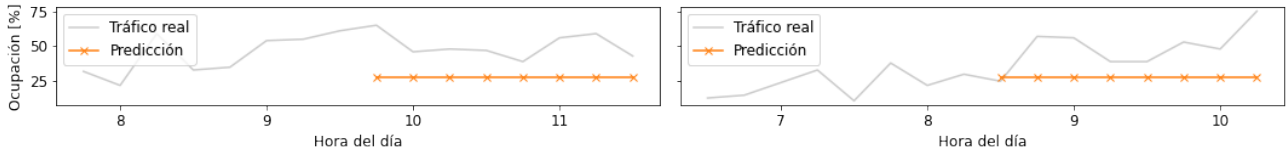


Figure 6.1: Estimador de media

El valor de la predicción siempre es el mismo: la media de la intensidad en los datos de entrenamiento.

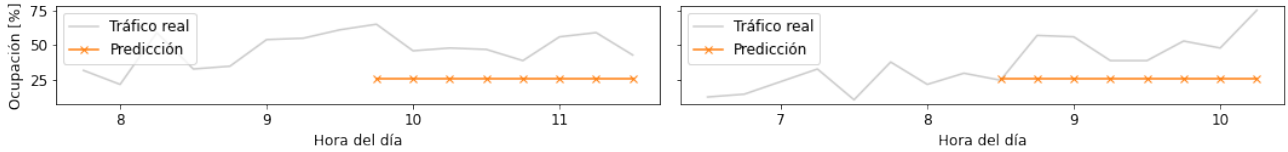


Figure 6.2: Estimador de mediana

El valor de la predicción siempre es el mismo: la mediana de la intensidad en los datos de entrenamiento.

### 6.1.2 Regresor dependiente de las otras variables

La predicción de estos modelos no es siempre la misma en cualquier instante $t$ para un punto de medida, sino que depende de la hora (en períodos de 15 minutos) y, opcionalmente, del hecho de ser un día laborable o no. Dado que la predicción depende de las otras variables en el momento indicado, estos modelos se incluyen dentro de los modelos no dependientes de los instantes anteriores, o a largo plazo.

#### 6.1.2.1 Estimador dependiente de la hora

1. **Media por hora del día**: El modelo se entrena calculando la media de la variable objetivo en un punto de medida a una hora. Véase el ejemplo en la figura 6.3.

2. **Mediana por hora del día**: Igual que en el punto anterior, pero usando la mediana. Véase el ejemplo en la figura 6.4.

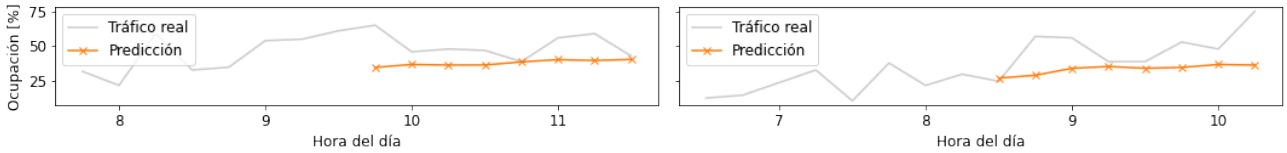


Figure 6.3: Estimador de media por hora del día
El valor de la predicción depende de la hora del día y es la media de la intensidad en los datos de entrenamiento a esa hora.

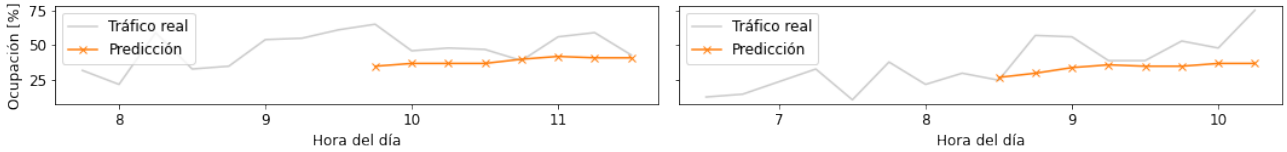


Figure 6.4: Estimador de mediana por hora del día
El valor de la predicción depende de la hora del día y es la mediana de la intensidad en los datos de entrenamiento a esa hora.

#### 6.1.2.2 Estimador dependiente de la hora y del tipo de día

Igual que en 6.1.2.1, pero el estimador no solo es dependiente de la hora, sino también del hecho de ser un día laborable o no serlo. Igual que en el punto anterior, también se usan media y mediana. Véanse los ejemplos en las figuras 6.5 y 6.6.

### 6.1.3 Modelos basados en repetición

Estos modelos basan su estimación en las observaciones de la variable objetivo en los $T'$ períodos inmediatamente anteriores a la predicción. Por ello, estos modelos se encuadran dentro de los modelos dependientes de los instantes anteriores, o de predicción a corto plazo.

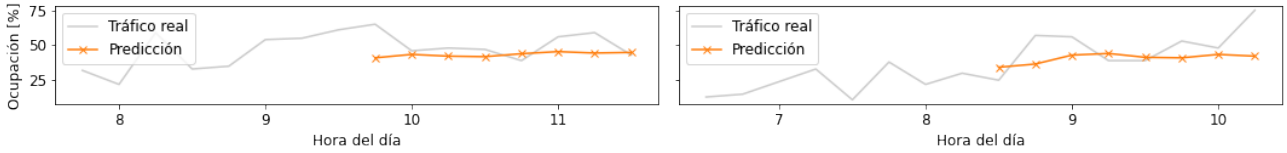


Figure 6.5: Estimador de media por hora y tipo de día
El valor de la predicción depende de la hora y del tipo de día y es la media de la intensidad en los datos de entrenamiento a esa hora y en ese tipo de día.

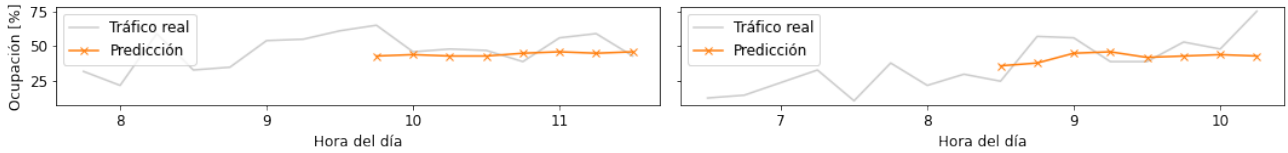


Figure 6.6: Estimador de mediana por hora y tipo de día

El valor de la predicción depende de la hora y del tipo de día y es la mediana de la intensidad en los datos de entrenamiento a esa hora y en ese tipo de día.

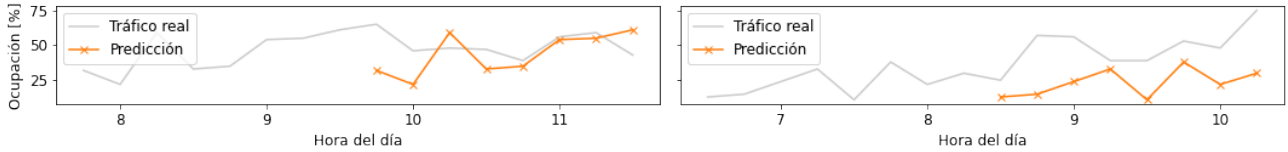


Figure 6.7: Repetición completa

Repetición completa de las $T$ observaciones anteriores

#### 6.1.3.1 Repetición completa

Este modelo solo se puede aplicar en el caso de que $T = T'$. La predicción en los instantes $t+1, \ldots, t+T$ será $Y^{(t-T'+1)}, \ldots, Y^{(t)}$. Este modelo tiene sentido aplicarlo si se observa algún tipo de estacionalidad en la variable objetivo con periodicidad $T$. Véase un ejemplo en la figura 6.7.

#### 6.1.3.2 Repetición del último valor

En este caso, para todos los instantes $t+1, \ldots, t+T$, la predicción es $Y^{(t)}$. Véase un ejemplo en la figura 6.8.

### 6.1.4 Modelo de variación

Este modelo, también dentro de los modelos dependientes de los instantes anteriores, toma las variación media de $Y$ en un período de tiempo y lo extiende a las predicciones. Es decir, dados $Y^{(t-T'+1)}, \ldots, Y^{(t)}$, calcula la variación media: $v = \dfrac{Y^{(t)} - Y^{(t-T'+1)}}{T'}$ y la predicciones serán:

$$Y^{(t+i)} = Y^{(t)} + i \times v$$

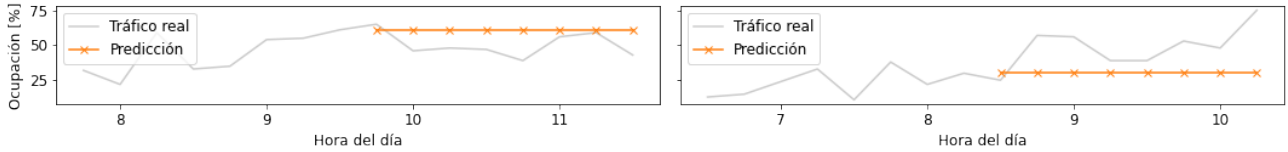


Figure 6.8: Repetición de la observación realizada en $t$ durante $T$ veces

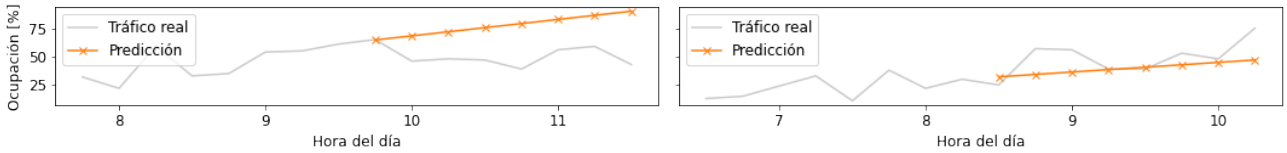


Figure 6.9: Modelo de variación
Se supone que la pendiente se mantiene constante y se realizan las predicciones en base a esta propiedad

## 6.2 Regresión lineal

Aquí damos un paso importante en complejidad y utilizaremos diversas variables, con diferentes tipos de transformaciones, para pronosticar la magnitud de tráfico. Dado que la predicción no depende de los valores inmediatamente previos de la variable objetivo, los modelos de regresión que utilizaremos se encuadran dentro de los modelos no dependientes de los instantes anteriores, o de predicción a largo plazo.

### 6.2.1 Variables utilizadas

Al contrario que en los modelos básicos presentados en 6.1, aquí tenemos la oportunidad de utilizar muchas otras variables, temporales y meteorológicas, que nos permitan pronosticar los niveles de tráfico. Sin embargo, una selección de estas variables es necesaria para evitar el sobreajuste u *overfitting*.

Aquí utilizaremos las transformaciones explicadas en el anexo A y compararemos su rendimiento en las predicciones finales.

Además, antes de aplicar la regresión, normalizamos las variables (restamos media y dividimos entre la desviación estándar).

### 6.2.2 Posibles transformaciones de las variables

El modelo final de regresión lineal usará como variables explicativas las temporales y las meteorológicas. En A y B se explican los diferentes tipos de transformaciones que vamos a utilizar. Para cada variable, vamos a evaluar los resultados si la usamos en su forma original, si usamos cada una de las transformaciones o si no la usamos en absoluto. A continuación, una lista de todas las diferentes opciones que tenemos para cada una de las variable temporales:

1. Año: forma original, *one-hot*, no usarla.

2. Estación del año: *one-hot*, ordinal, no usarla.

3. Mes: todas las del anexo A o no usarla.
4. Día del mes: todas las del anexo A o no usarla.
5. Día de la semana: forma original, *one-hot*, no usarla.
6. Hora: todas las del anexo A o no usarla.
7. Minuto: forma original, *one-hot*, no usarla.
8. Día festivo (binaria): forma original, no usarla.
9. Vacaciones escolares (binaria): forma original, no usarla.
10. Día laborable (binaria): forma original, no usarla.
11. Estado de alarma (binaria, entre el 14 de marzo y 21 de junio de 2020): forma original, no usarla.

En total, podemos ver que tenemos hasta 162000 combinaciones diferentes ($3 \times 3 \times 5 \times 5 \times 3 \times 5 \times 3 \times 2 \times 2 \times 2 \times 2 = 162000$). En 6.2.3 veremos qué estrategia usaremos para estimar los parámetros óptimos sin tener que probar todas las variantes.

Para las variables meteorológicas, tenemos las siguientes opciones:

1. Precipitación: todas las del anexo B.1 o no usarla.
2. Viento: las del anexo B.2 o no usarla.
3. Temperatura, humedad relativa, presión barométrica y radiación solar: usarlas o no usarlas.

En este caso, tenemos $7 \times 3 \times 2 \times 2 \times 2 \times 2 = 336$ posibilidades diferentes.

### 6.2.3 Selección del modelo

El proceso para seleccionar los mejores parámetros es el siguiente:

1. Tomamos como sensor el 1001 (luego generalizaremos los parámetros para entrenar modelos para los otros sensores).
2. Utilizaremos los datos de entrenamiento descritos en 4.1 de 2019, 2020 y 2021, pero usando menos de un 10% de los datos para agilizar la selección. Esto lo hacemos usando solo una de cada 11 observaciones (es decir, cada 2 horas y tres cuartos). Este valor no es un valor redondo, para no utilizar siempre valores que sean en el mismo minuto (asi usamos observaciones a en punto, a y vuarto, a y media y a menos cuarto a partes iguales).

3. Iteramos sobre las 336 posibilidades dentro de las combinaciones de parámetros meteorológicos. Para cada una de ellas, iteramos sobre el 0.5% de las 162000 combinaciones de parámetros temporales (en total, 81 combinaciones). En cada caso, utilizando el procedimiento descrito en 4.2, almacenamos los resultados de error absoluto medio (MAE) y error cuadrático medio (MSE).

4. Entonces tenemos, para cada una de las 336 posibilidades de uso de las variables meteorológicas, 81 valores de MAE y MSE. Hacemos la media de MAE y la media de MSE para cada una de ellas.

5. Creamos un ranking de los dos tipos de error (MAE y MSE). Es decir, ordenamos las 336 posibilidades en función del MAE y del MSE, en orden ascendente, y le asignamos una posición según este orden. Combinamos los dos rankings seleccionando la posición máxima para cada uno de los 336 elementos.

6. Una vez tenemos esta nueva clasificación, tomamos las 20 combinaciones de parámetros con los mejores resultados. Entre ellas, las configuraciones de las variables meteorológicas son las siguientes:

   - Precipitación: no usarla.
   - Viento: Todas las del anexo B.2.
   - Temperatura: En su forma original o no usarla.
   - Humedad relativa: En su forma original.
   - Presión barométrica: No usarla.
   - Radiación solar: Usarla en su forma original o no usarla.

   En contra de lo que podíamos esperar, nuestra estrategia dice que los datos de humedad relativa dan más información sobre el flujo del tráfico que la precipitación.

7. Ahora vamos a probar las 162000 combinaciones de transformaciones temporales, junto con las combinaciones de transformaciones de las variables meteorológicas del paso 6. Igual que en el paso 3, utilizamos el procedimiento de validación cruzada descrito en 4.2, con menos de un 10% de los datos de entrenamiento y almacenamos los MAE y MSE.

8. Con los nuevos MAE y MSE, creamos un ranking siguiendo el mismo procedimiento que en el paso 5. Debido a que las mejores combinaciones, según este ranking, son muy parecidas, vamos a optar por elegir como segunda configuración una con peores resultados, pero con más diferencias respecto a la primera. En concreto, la número 320 en el ranking. Las dos configuraciones se pueden ver en la tabla 6.1.

   Aun así, vemos que las dos configuraciones solo se diferencian en la manera de usar la precipitación, el día del mes, el día de la semana, la hora y el minuto.

| Variable | Primera configuración | Segunda configuración |
|---|---|---|
| **Precipitación** | Transformación de potencia Yeo-Johnson | Ordinal |
| **Viento** | No usarla | |
| **Temperatura** | Forma original | |
| **Humedad relativa** | Forma original | |
| **Presión barométrica** | No usarla | |
| **Radiación solar** | Forma original | |
| **Año** | Forma original | |
| **Estación del año** | Ordinal (verano-primavera-otoño-invierno) | |
| **Mes** | Forma original | |
| **Día del mes** | Forma original | Transformada de Fourier (dos términos) |
| **Día de la semana** | No usarla | Forma original |
| **Hora** | One-hot | Spline |
| **Minuto** | No usarla | Forma original |
| **Día festivo** | Forma original | |
| **Vacaciones escolares** | No usarla | |
| **Día laborable** | Forma original | |
| **Estado de alarma** | Forma original | |

Table 6.1: Las dos configuraciones de regresión lineal utilizadas

### 6.2.4 Visualización de los dos modelos

Una vez que nuestro algoritmo ha estimado las mejores variables a utilizar y con qué transformaciones, creamos una *pipeline* con los siguientes pasos:

(a) Las transformaciones correspondientes.

(b) Estandarización de las variables.

(c) Regresión de Ridge con validación cruzada, como se explica en 5.1.2.

Para visualizar las dos configuraciones, escogemos cinco sensores de los seleccionados en 3.1.6 y entrenamos, para cada uno de ellos, la *pipeline* correspondiente con cada una de las dos configuraciones. A continuación tomamos una muestra aleatoria del conjunto de test: en concreto, las observaciones ocurridas entre las 20:15 del 13 y 14 de junio de 2021, y aplicaremos los modelos en este período. Recordamos que no es casualidad que esta muestra esté en el tramo final de nuestros datos, ya que los datos de test se encuentran siempre al final.

Los valores reales de ocupación y los estimados por sus respectivos modelos se pueden observar en las figuras 6.10, donde cada columna es uno de los dos períodos de tiempo y, cada fila, cada uno de los puntos de medida.

En la figura 6.10, la diferencia más clara es que las predicciones en la primera configuración están escalonadas por horas, mientras que, en la segunda, tienen un recorrido mucho más suave. Esto es porque, en la primera, los modelos están utilizando las horas con transformación *one-hot*, mientras que en la segunda, están utilizando las horas con transformación de splines.

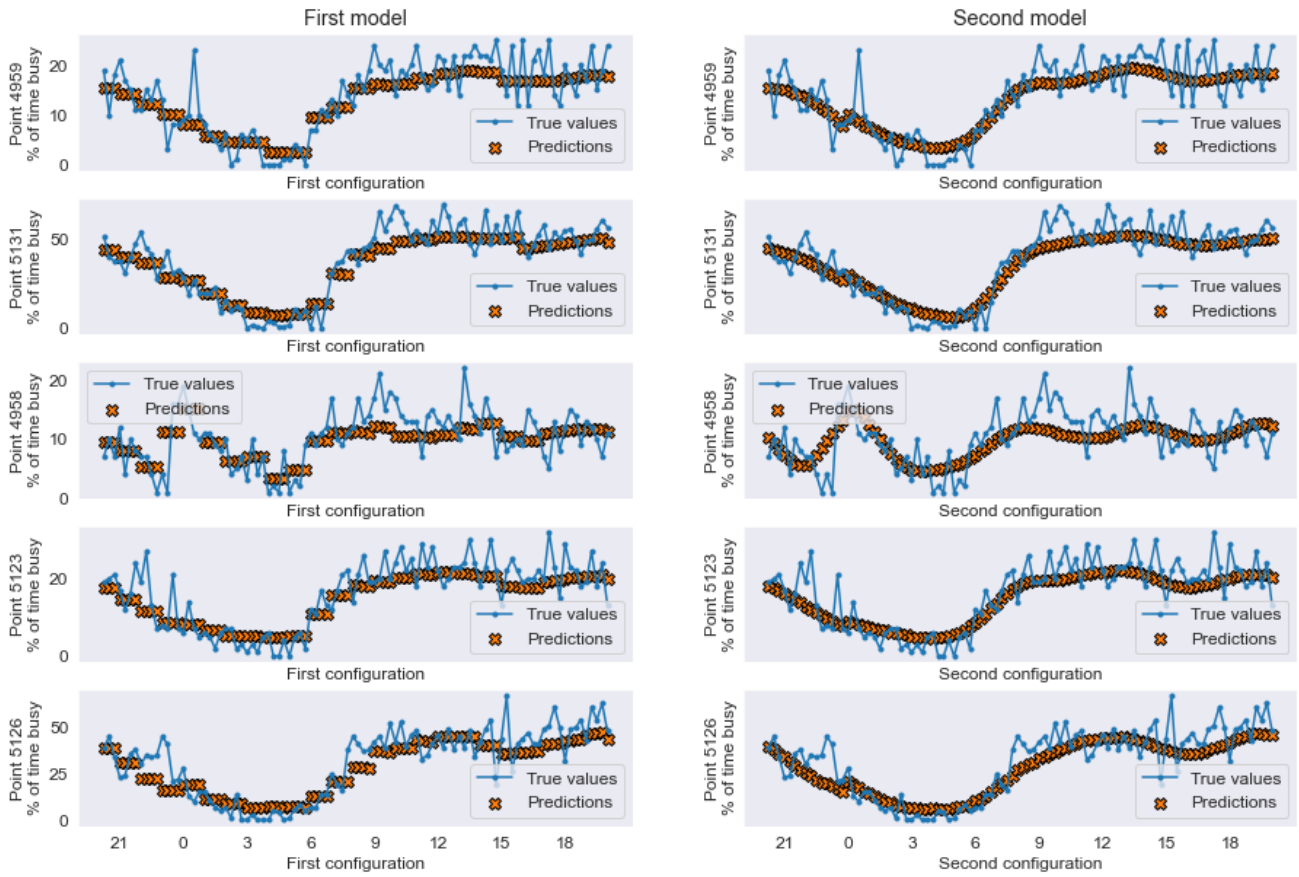


Figure 6.10: Ejemplo de regresión en cinco sensores y un período de 24 horas. Cada fila muestra los valores reales y las predicciones del modelo en el punto de medida. Cada columna se corresponden con una de las dos configuraciones.

Los coeficientes de regresión que los cinco modelos de cada configuración han aprendido son los de las figuras 6.11 y 6.12. Las variables a la izquierda son diferentes en cada imagen, ya que las dos configuraciones utilizan diferentes variables o con diferentes transformaciones.

En la primera, en 6.11, como está utilizando *one-hot*, cada hora tiene una fila. Las horas correspondientes a la madrugada tienen un peso muy negativo, como podíamos esperar. La variable que indica que es un día laborable tiene un peso muy positivo, y la variable que indica los días de estado de alarma de 2020, muy negativo, lo cual también podíamos esperar. El resto de variables, incluyendo todas las meteorológicas, tienen pesos muy reducidos. Sería necesario un estudio detallado del efecto de estas variables para decidir si tiene sentido mantenerlas en el modelo.

En la segunda, como estamos usando 12 splines, que no se corresponden directamente con cada hora, la interpretación no es tan directa. Pero podemos intuir que los splines del 1 al 5 se corresponden con las horas de la madrugada, ya que tienen pesos muy negativos.

Igual que en el caso anterior, las otras variables que tienen un peso importante son las del día laboral y estado de alarma, mientras que todas las demás tienen un efecto muy pequeño.

Respecto a los valores de $\alpha$ obtenidos en la regresión de Ridge (exlicado en 5.1.2), de entre los que se dan como opción, (0,1, 1 y 10), siempre ha salido elegido 10. Esto quiere decir que los modelos que han obtenido los mejores resultados son aquellos que se han regularizado más. Sería conveniente analizar esta situación, que podría significar que se están usando demasiadas variables que aportan poca información y aumentan el sobreajuste u *overfitting*.

## 6.3 XGBoost

### 6.3.1 Variables explicativas a utilizar

Las variables explicativas con las que vamos a entrenar los modelos de XGBoost, así como sus transformaciones, son exactamente las mismas que hemos utilizado en la primera configuración de la regresión lineal. Dado que, en ese caso, nuestro método ha estimado que esa es la mejor manera de usar las variables, asumimos que es también un buen punto de partida para XGBoost.

### 6.3.2 Ajuste de parámetros de XGBoost

Este paso lo realizaremos únicamente usando los datos de entrenamiento, siguiendo el procedimiento descrito en 6.2.3. Vamos a ajustar los hiperparámetros de *booster*, que regulan el *booster* en cada paso:

1. `n_estimators`: El número de árboles.

2. `learning_rate`: Es la tasa de aprendizaje. Valores cada vez más pequeños hacen que el modelo sea más robusto.

3. `min_child_weight`: Es el umbral mínimo que debe tener la suma de pesos de los elementos en un *hijo* para que se continúe la división.

4. `max_depth`: Es la profundidad máxima de un árbol. Valores altos pueden causar el sobreajuste.

5. `gamma`: Un nodo se divide solo si el resultado de la división produce un resultado igual o mayor que `gamma`, que es un valor positivo o 0.

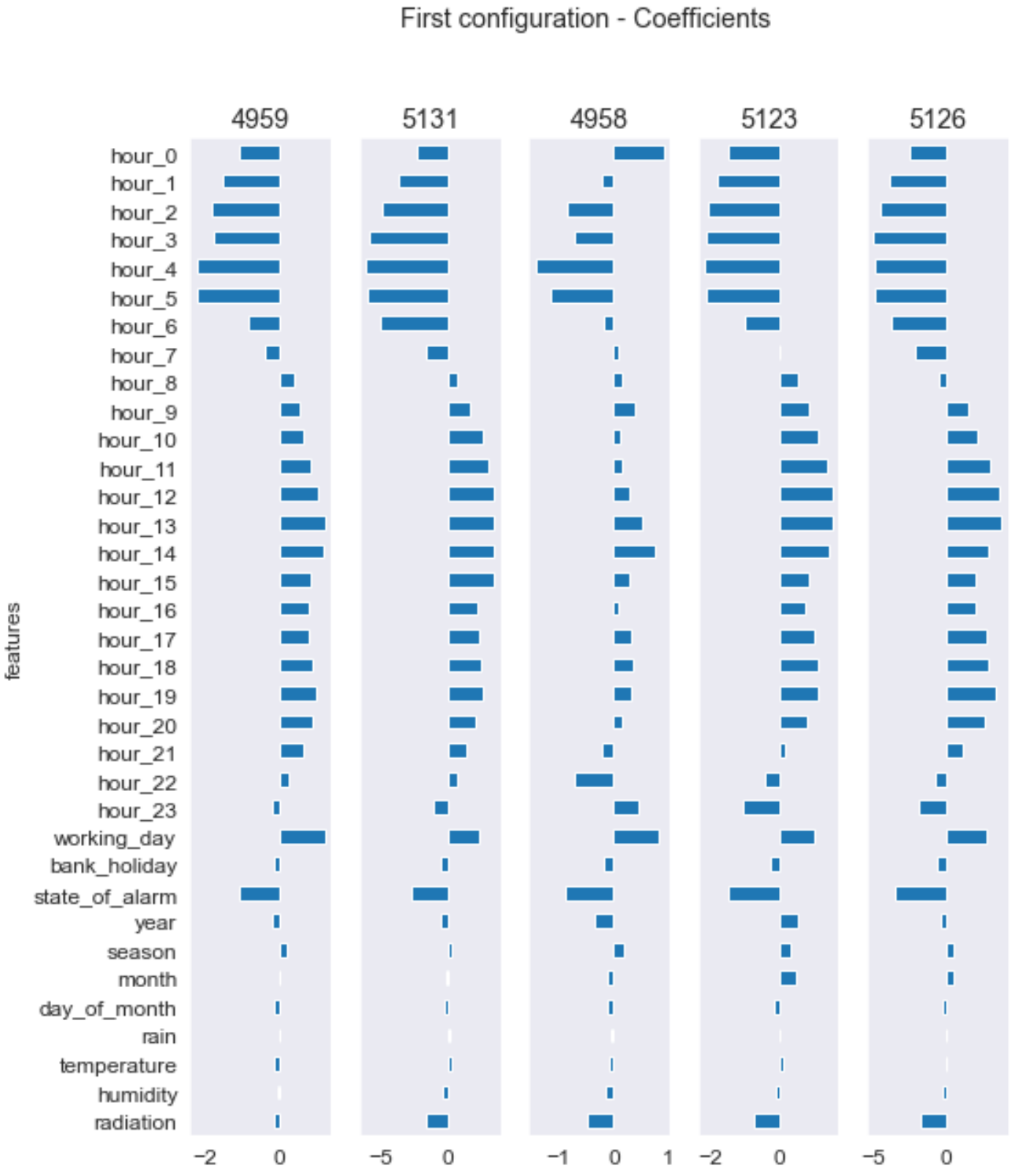


Figure 6.11: Coeficientes de la regresión lineal con la primera configuración
Cada fila muestra una variable y, cada barra, su aportación positiva o negativa en el modelo.

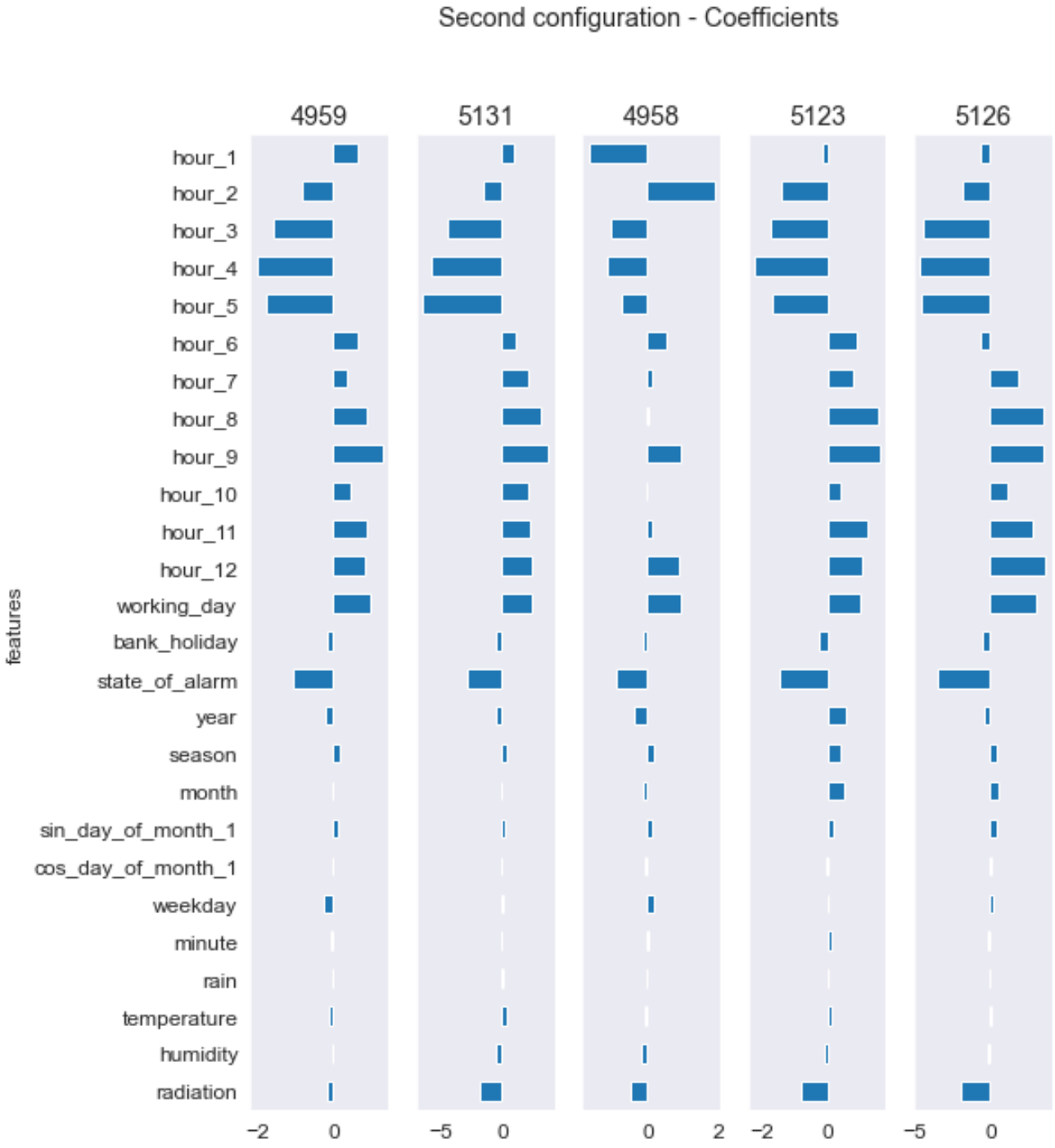


Figure 6.12: Coeficientes de la regresión lineal con la segunda configuración
Cada fila muestra una variable y, cada barra, su aportación positiva o negativa en el modelo.

6. `subsample`: Define la fracción de las observaciones que se usarán de manera aleatoria para cada árbol. Valores bajos hacen el modelo más conservativo, es decir, menos proclive al sobreajuste.

7. `colsample_bytree`: Fracción de variables explicativas que se utilizarán para cada árbol.

8. `reg_alpha`: Regularización L1 (Lasso).

9. `reg_lambda`: Regularización L2 (Ridge)

### 6.3.3 Procedimiento de ajuste

El procedimiento que seguiremos está basado en Jain (2016), y lo realizamos por separado para cada uno de los sensores, ya que finalmente entrenaremos un modelo para cada sensor. En todo caso se utiliza el error absoluto medio (MAE) para elegir los parámetros óptimos.

1. Fijamos todos los parámetros, excepto `n_estimators`, parámetro para el cual buscamos el valor óptimo entre 20 y 200.

2. Realizamos una primera ronda de optimización de los parámetros `max_depth` y `min_child_weight`.

3. Repetimos la ronda anterior, pero con valores en un rango más detallado.

4. Ajustamos `gamma`.

5. Una vez que tenemos las variables `max_depth`, `min_child_weight` y `gamma` ajustadas, volvemos a ajustar `n_estimators`.

6. Ajustamos `subsample` y `colsample_bytree`.

7. Reajustamos los mismos dos parámetros, pero en intervalos más finos.

8. Ajustamos `reg_alpha` y `reg_lambda`.

Los parámetros óptimos son diferentes para cada uno de los 37 sensores, y su distribución se puede ver en la figura 6.13.

Si tomamos uno de los sensores y visualizamos la importancia de cada una de las variables explicativas, obtenemos la figura 6.14. Es sorprendente que, aparentemente, las variables más importantes del modelo son la temperatura y la humedad relativa, mientras que habíamos esperado que las variables más importantes fuesen la hora y si es día laborable. La explicación a esto es que el modelo está aprendiendo información temporal a partir de la información meteorológica. Una vez más, aquí sería preciso hacer un análisis con menos variables y analizar los resultados.

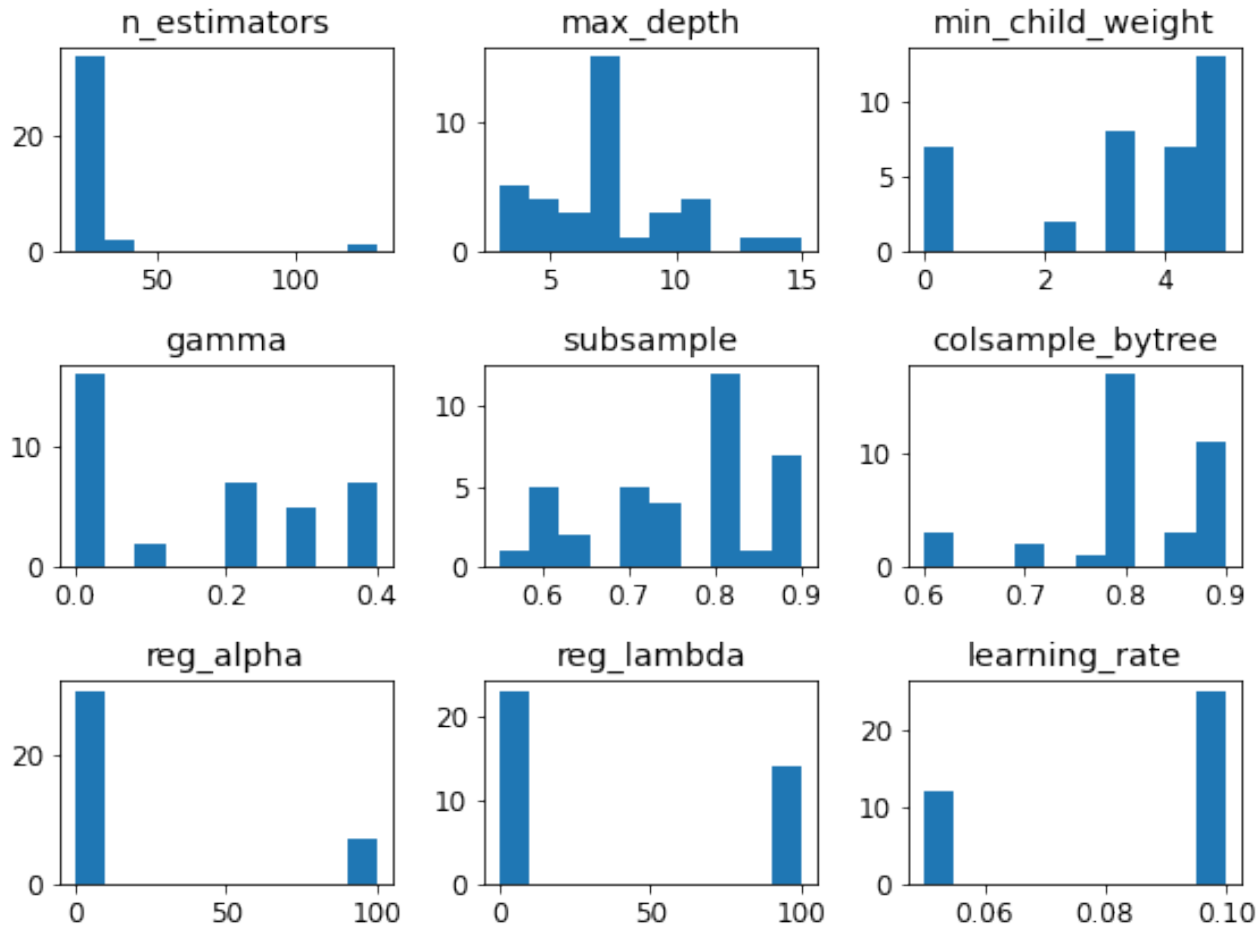


Figure 6.13: Distribución de los parámetros finales para XGBoost
Para cada sensor se entrena un modelo con el parámetro óptimo correspondiente. Estos parámetros se han estimado mediante una búsqueda con la misma estrategia, pero separada para cada sensor.

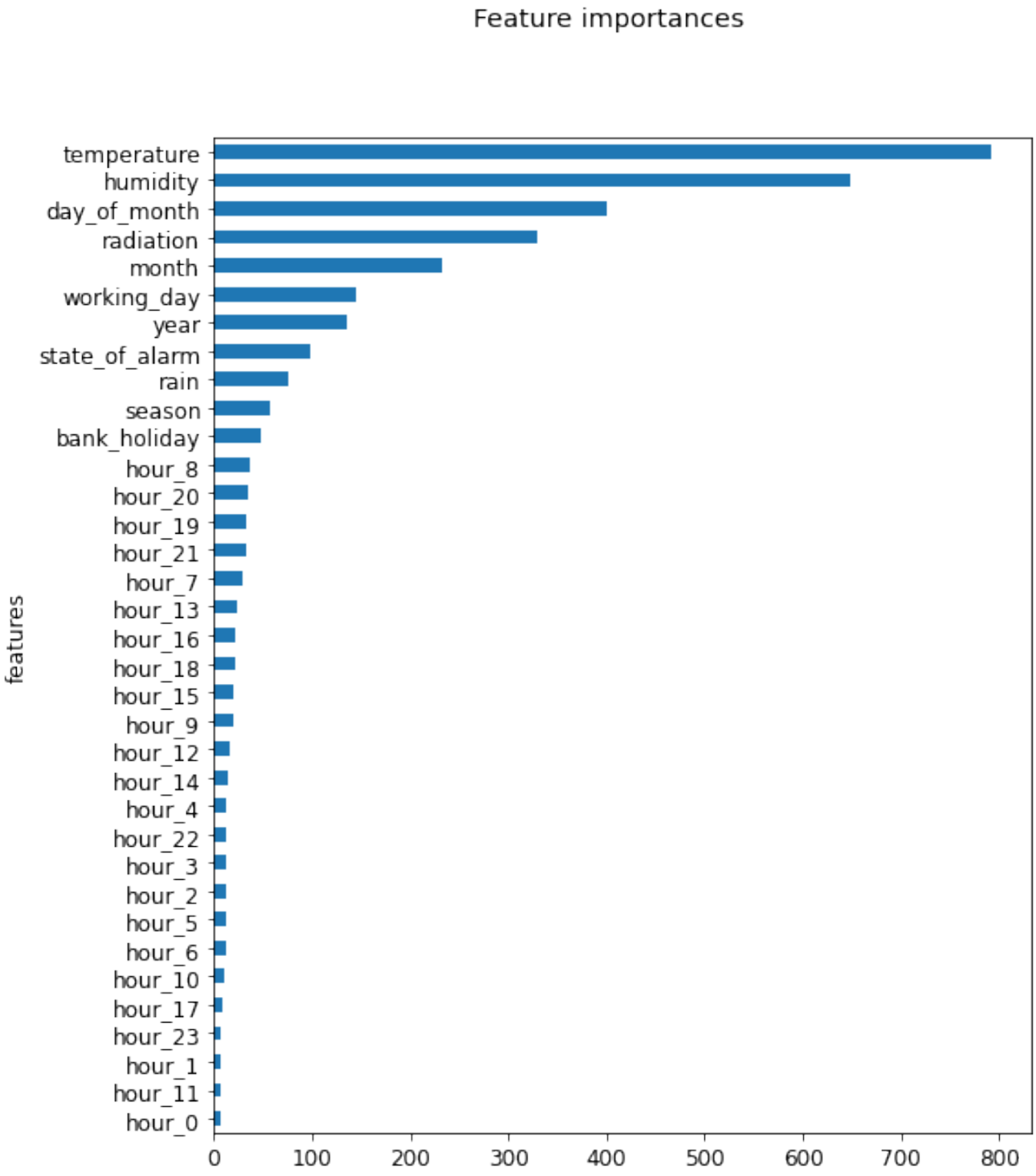


Figure 6.14: Importancia de las variables explicativas en un modelo de XGBoost

## 6.4 Red neuronal recurrente convolucional de difusión en grafos (GNN)

### 6.4.1 Construcción del grafo

Sea $\mathcal{V}$ el conjunto de nodos donde queremos construir el modelo. Entonces construimos un grafo donde cada $v_i \in \mathcal{V}$ es un nodo. Las aristas se construyen de la siguiente manera:

1. Para todo $v_i, v_j \in \mathcal{V}$ con $i \neq j$, $d_{i,j}$ es la distancia en segundos en coche desde $v_i$ a $v_j$ según OpenStreetMap contributors (2017). El valor se obtiene con la siguiente solicitud:

   `http://router.project-osrm.org/route/v1/car/<>,<>,<>,<>?overview=false`

   donde los `<>` se tienen que sustituir por la latitud y longitud de $v_i$ y de $v_j$, respectivamente. Es importante observar que, en general, $d_{i,j} \neq d_{j,i}$. Es importante observar que:

   - La ubicación del sensor tiene en cuenta la dirección. Es decir, el portal de datos publica la ubicación del sensor exactamente en el carril que le corresponde.
   - *OpenStreetMap* tiene en cuenta el carril y, por tanto, la dirección de conducción, para calcular las rutas. Dado que nuestro grafo es dirigido y sí que es importante la dirección, las distancias se calcularán correctamente.

2. Calculamos $\sigma_{\text{distancias}}$ como la desviación estándar de todas las distancias $d_{i,j}$, con $i \neq j$.

3. Los pesos los calculamos con un kernel Gaussiano truncado, descrito en Shuman et al. (2013): Al peso de la arista de $v_i$ a $v_j$ lo llamamos $w_{i,j}$ y lo calculamos con la siguiente fórmula:

$$w_{i,j} = \exp\left(\frac{-d_{i,j}^2}{\sigma_{\text{distancias}}^2}\right)$$

4. Para hacer el grafo más sencillo y disperso, eliminamos aquellas aristas que tienen un peso $w_{i,j} < \kappa$, para algún $\kappa > 0$ que elegiremos tras algunas pruebas.

5. Transformamos el grafo al formato de la librería DGL: `dgl.data.GraphDataset`.

Para la visualización del grafo, utilizamos la librería networkx, Hagberg et al. (2008), con el algoritmo de visualización Kamada-Kawai, introducido en Kamada and Kawai (1989). Este método, a pesar de estar diseñado para grafos no dirigidos, funciona bien en nuestro caso.

Usando los 37 nodos seleccionados en 3.1.6, y utilizando $\kappa \in \{0.05, 0.1, 0.5\}$ tendríamos como resultado los tres grafos de la figura 6.15. Tras hacer algunas pruebas, vemos que los modelos convergen mejor y tienen mejores resultados cuando $\kappa = 0.1$.

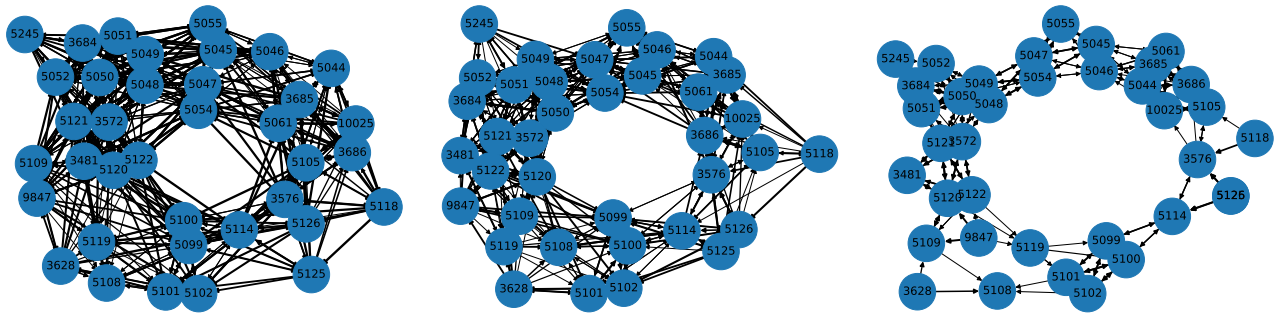


Figure 6.15: La red de puntos de medida en forma de grafo, utilizando tres valores de $\kappa$ diferentes
Los valores de $\kappa$ (el peso mínimo que debe tener una arista para no ser ignorada) son, respectivamente, 0.05, 0.1 y 0.5.

### 6.4.2 Deep Graph Library (DGL)

Para facilitar la investigación en el área de redes neuronales de grafos, en 2018 se comenzó a implementar la Deep Graph Library (Wang et al. (2019)) en código abierto.

Esta librería descompone las operaciones en tensores dispersos en un conjunto de primitivas de paso de mensajes (*message passing*) configurables por el usuario. Estas primitivas generalizan las operaciones en tensores y cubren tanto el paso hacia adelante (*forward*) como hacia atrás (*backward*). El paso de mensajes consiste en una de los siguientes casos:

- Agregar las características de las aristas adyacentes a un nodo para actualizar las características del nodo.
- Agregar las características de los nodos adyacentes a una arista para actualizar las características de la arista.

Además, la abstracción principal de la librería es el grafo, lo que permite optimizar los algoritmos de una manera sencilla.

La librería se puede utilizar como complemento de otras librerías de aprendizaje profundo, como PyTorch, Tensorflow y MXNet. La mayoría de ejemplos que se pueden encontrar en Internet utilizan PyTorch.

### 6.4.3 Observaciones sobre el modelo

Las condiciones en las que se aplica el modelo original y aquellas en las que lo estamos aplicando se diferencian en los siguientes puntos:

- El modelo original se usa para predecir la velocidad media en los nodos de la red, mientras que en este trabajo, por las razones mencionadas en 3.1.5, se está usando para predecir la ocupación (porcentaje del tiempo en que la vía está ocupada).

- Los datos del modelo original están agregados en períodos de cinco minutos, y, los nuestros, en períodos de 15 minutos.
- Las dos redes que usa modelo original tienen un mayor número de nodos (207 y 325), al contrario de los 37 que hemos seleccionado en 3.1.6.
- Los dos conjuntos de datos que utiliza el modelo original se extienden en períodos de 4 y 6 meses, respectivamente, mientras que el nuestro utiliza datos de tres años, y con una pandemia de por medio.
- Al contrario que en el modelo original, muchos de los sensores de nuestra red tienen períodos largos en los que no hay disponibilidad de datos, obligando a reducir en gran medida el conjunto total.
- El modelo original realiza predicciones a 15, 30 y 60 minutos. En este trabajo se van a evaluar períodos más largos de tiempo, de hasta 4 horas (16 períodos de 15 minutos).
- Por último, en este trabajo evaluamos el uso de otras variables diferentes al tráfico, mientras que el modelo original solo utiliza como características de los nodos las propias variables de tráfico.

Por estas razones, ha sido preciso, por una parte, adaptar el modelo en los siguientes aspectos:

- Para que la red neuronal use también características de los nodos, como las utilizadas en el modelo de regresión en 6.2.
- Se ha corregido la función que calculaba el error para adaptarla a nuestro caso.
- La implementación no utiliza las fórmulas originales de una red recurrente cerrada (GRU), que son, además las que se utilizan en el artículo, ya que intercambia en un caso la función sigmoide y la función de tangente hiperbólica. Esto se ha corregido.
- La implementación original tiene un error en la asignación de grafos a la red neuronal. Este error también se ha tenido que corregir.

El proceso hasta encontrar parámetros en los que el modelo funcione ha sido un proceso de prueba y error. Se han tomado los datos de entrenamiento y, sucesivamente, se ha:

1. Entrenado con los datos hasta febrero y evaluado en marzo.
2. Entrenado con los datos hasta marzo y evaluado en abril.
3. Entrenado con los datos hasta abril y evaluado en mayo.

| Parámetro | Config. 1 | Config. 2 | Config. 3 | Config. 4 | Config. 5 |
|---|---|---|---|---|---|
| Variables adicionales a utilizar | Ninguna | Ninguna | Ninguna | Ninguna | Hora y si es día laborable |
| Número de capas | 2 | 2 | 1 | 1 | 2 |
| Número de neuronas ocultas | 64 | 32 | 32 | 64 | 64 |

Table 6.2: Las 5 configuraciones diferentes de la red neuronal de grafos (GNN)

Este proceso ha sido manual, no automático, como en los casos de la regresión lineal y *XGBoost*, ya que a veces los resultados eran muy malos, y se ha interrumpido el entrenamiento y seguido por otra vía. De manera manual, se han ido recogiendo los errores para cada parámetro en una hoja de cálculo, para así encontrar en qué rangos el modelo funciona mejor.

### 6.4.4 Entrenamiento final

Con el último proceso descrito, se han concluido cinco configuraciones diferentes para este modelo, que evalúamos para secuencias de 2, 4, 8 y 16 períodos, igual que en los modelos de referencia o de predicción a corto plazo. Los parámetros que son fijos en todas las configuraciones son:

1. Pasos de difusión: 2
2. Tasa de aprendizaje inicial: 0.01
3. Tamaño de *batch*: 64
4. Tasa de aprendizaje mínima: 2e-6
5. Norma máxima de los gradientes: 5
6. Umbral de peso mínimo para considerar una arista en un grafo o $\kappa$: 0.1

Los parámetros que varían en las diferentes configuraciones son los indicados en la tabla 6.2. Es interesante (para luego analizar mejor los resultados) que podríamos ordenar estas configuraciones de menor a mayor complejidad como: Config. 3, (Config. 2, Config. 4), Config. 1, Config. 5, donde el paréntesis indica que las dos configuraciones en él tienen una complejidad similar.

Entrenamos un modelo para cada una de estas configuraciones, para las 4 longitudes de secuencia $T = T'$ (2, 4, 8 y 16, donde cada período son 15 minutos), siguiendo la notación de 1.2. Además, como seguimos el procedimiento de evaluación de modelos temporales con

validación cruzada explicado en 4.2.2, repetimos el proceso 7 veces (primero entrenamos en mayo y evaluamos en junio, luego entrenamos hasta junio y evaluamos en julio; así hasta entrenar hasta noviembre y evaluar en diciembre).

### 6.4.5 Interrupción del entrenamiento

En el entrenamiento de cada modelo, guardamos los resultados obtenidos en cada *epoch*. El entrenamiento de detiene si el error absoluto o cuadrático medio en la *epoch* es mayor que el de las cinco últimas *epochs*.

En la fase de pruebas se ha observado que el modelo en general no tiende al sobreajuste u *overfitting*. Es decir, que normalmente la *epoch* en la que el modelo tiene un mejor resultado para entrenamiento, tiene un resultado similar para los datos de test, Por ello, de entre los modelos guardados en cada iteración vamos a escoger aquel que tiene un menor error absoluto medio en los datos de entrenamiento. De esta manera, no estamos utilizando los datos de test para realizar la elección del modelo y, por lo tanto, la evaluación de los modelos va a ser correcta.

### 6.4.6 Comentarios sobre el entrenamiento

Como se ha explicado en 6.4.4, se ha asumido que el modelo no tiende al sobre ajuste u *overfitting* y, por ello, se ha seleccionado el modelo en el que el error absoluto medio en los datos de entrenamiento era menor. Sin embargo, el aprendizaje de los modelos ha sido mucho menor cuando $T = T'$ era menor. En las figuras 6.16, 6.17, 6.18, 6.19 y 6.20 vemos la evolución del error absoluto medio (MAE) en el entrenamiento (con los datos hasta mayo) en las cinco configuraciones para los diferentes valores de $T = T'$. También señalizamos aquella *epoch* cuyo modelo se ha seleccionado como el mejor. Vemos que, en algunos casos se han realizado más *epochs* que en otras. Esto es porque, como se ha explicado en 6.4.5, si el MAE en una *epoch* es mayor que el de las cinco anteriores, entonces el entrenamiento se ha interrumpido.

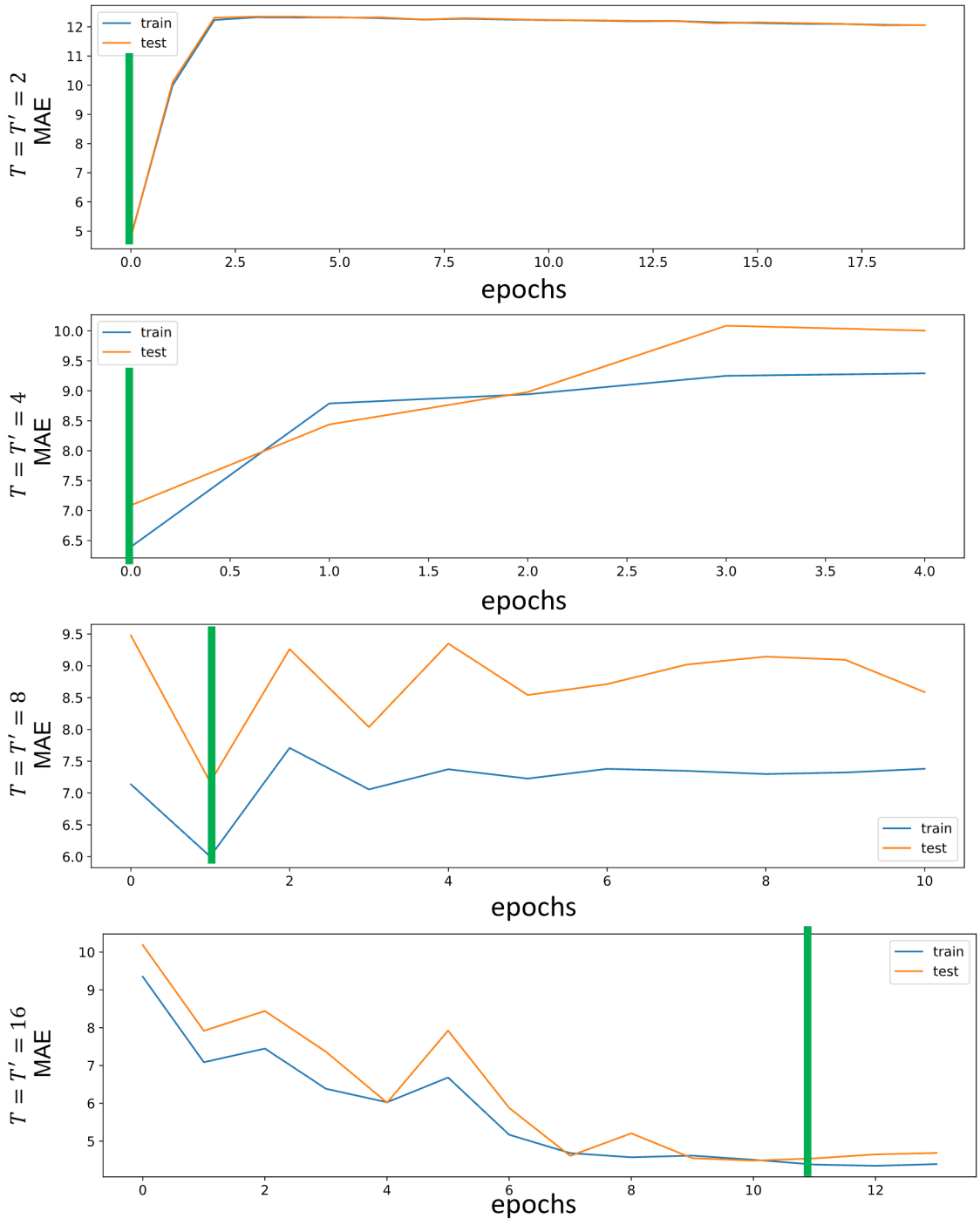


Figure 6.16: Evolución del MAE en el entrenamiento de la configuración 1 de la red neuronal recurrente convolucional de difusión en grafos (GNN)
Cada gráfico corresponde con un valor de $T$. El eje $x$ representa las *epochs*. La linea verde indica la *epoch* con menor MAE en los datos entrenamiento y cuyo modelo se ha seleccionado.

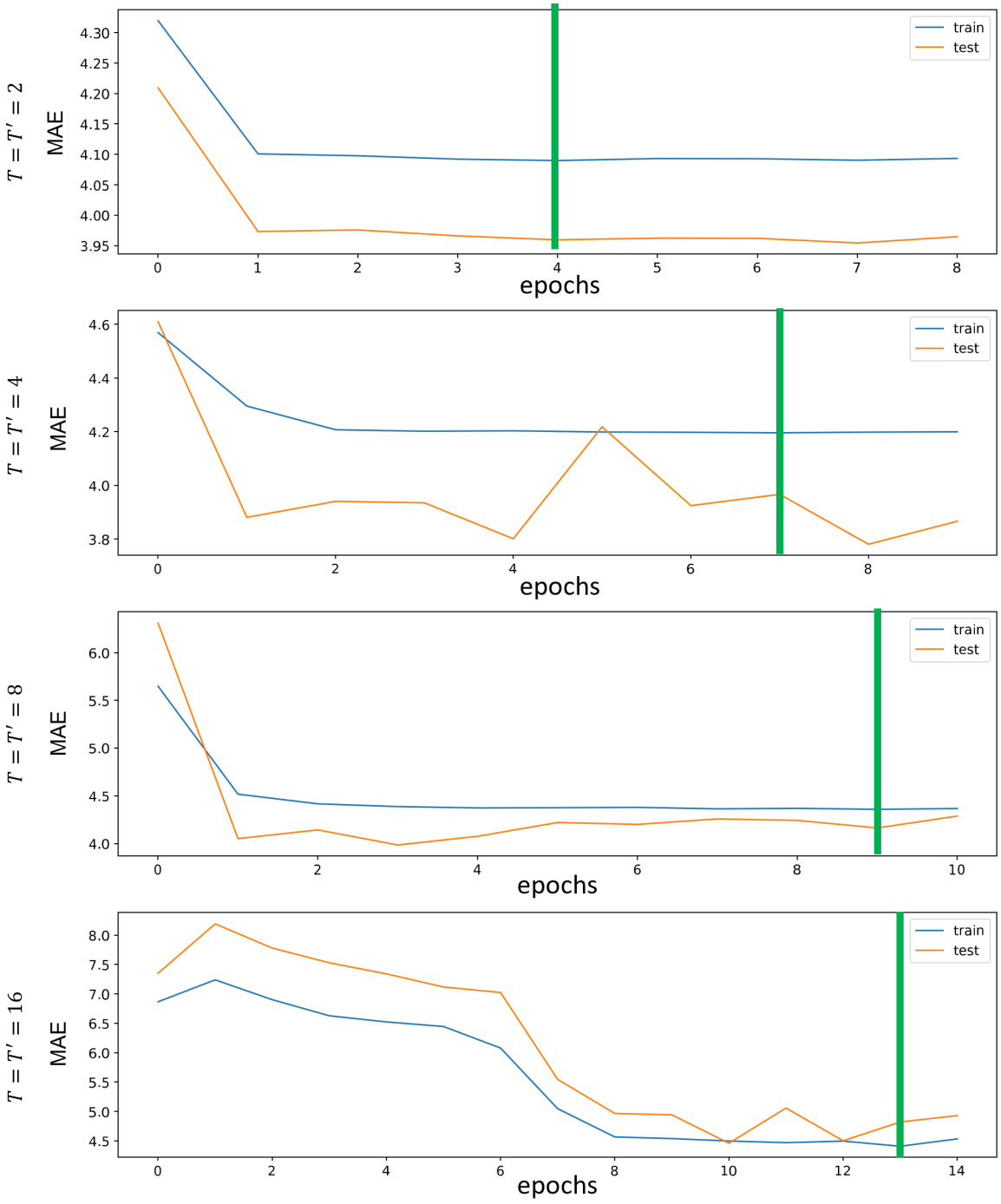


Figure 6.17: Evolución del MAE en el entrenamiento de la configuración 2 de la red neuronal recurrente convolucional de difusión en grafos (GNN)
Cada gráfico corresponde con un valor de $T$. El eje $x$ representa las *epochs*. La linea verde indica la *epoch* con menor MAE en los datos entrenamiento y cuyo modelo se ha seleccionado.

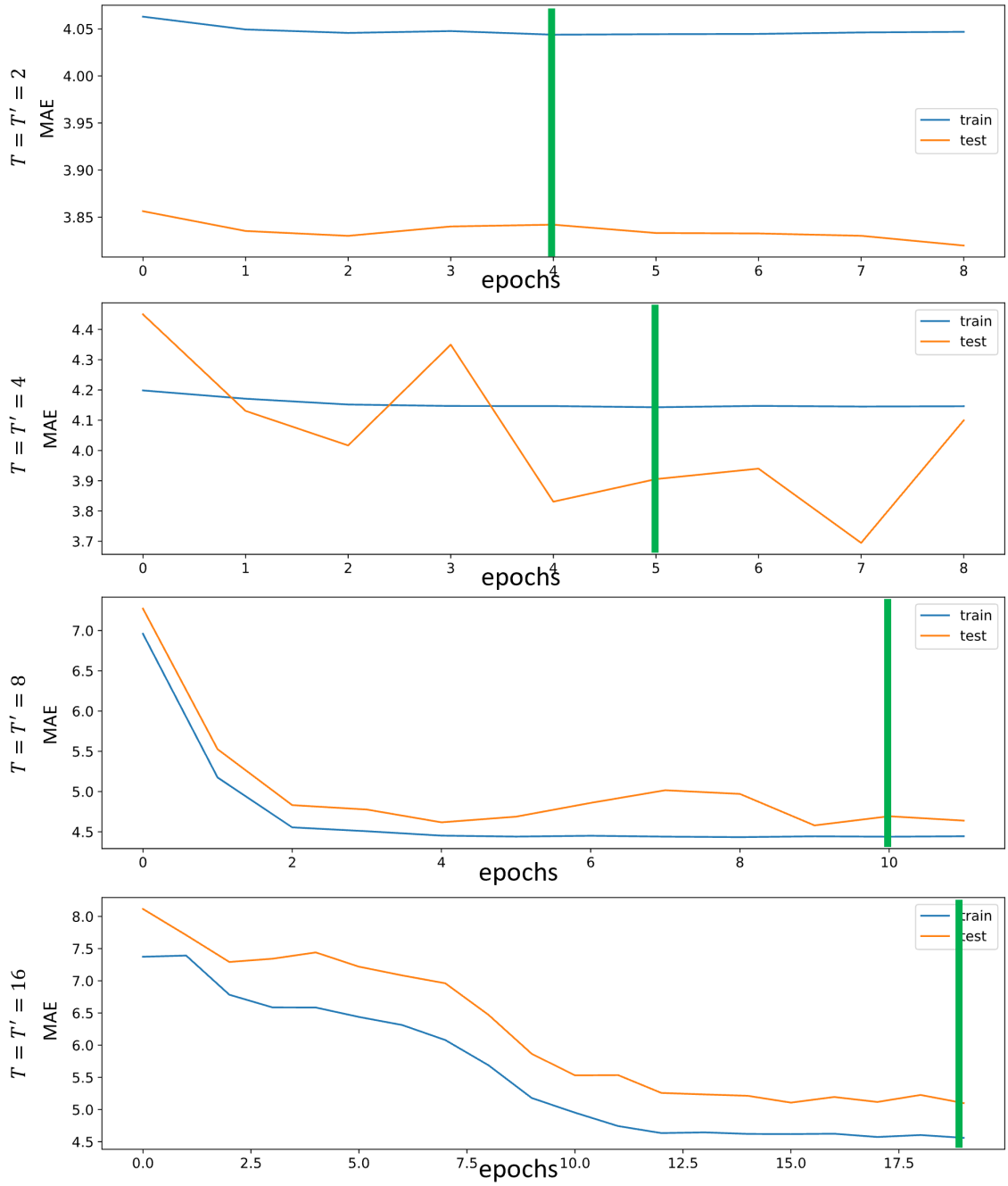


Figure 6.18: Evolución del MAE en el entrenamiento de la configuración 3 de la red neuronal recurrente convolucional de difusión en grafos (GNN)
Cada gráfico corresponde con un valor de $T$. El eje $x$ representa las *epochs*. La linea verde indica la *epoch* con menor MAE en los datos entrenamiento y cuyo modelo se ha seleccionado.

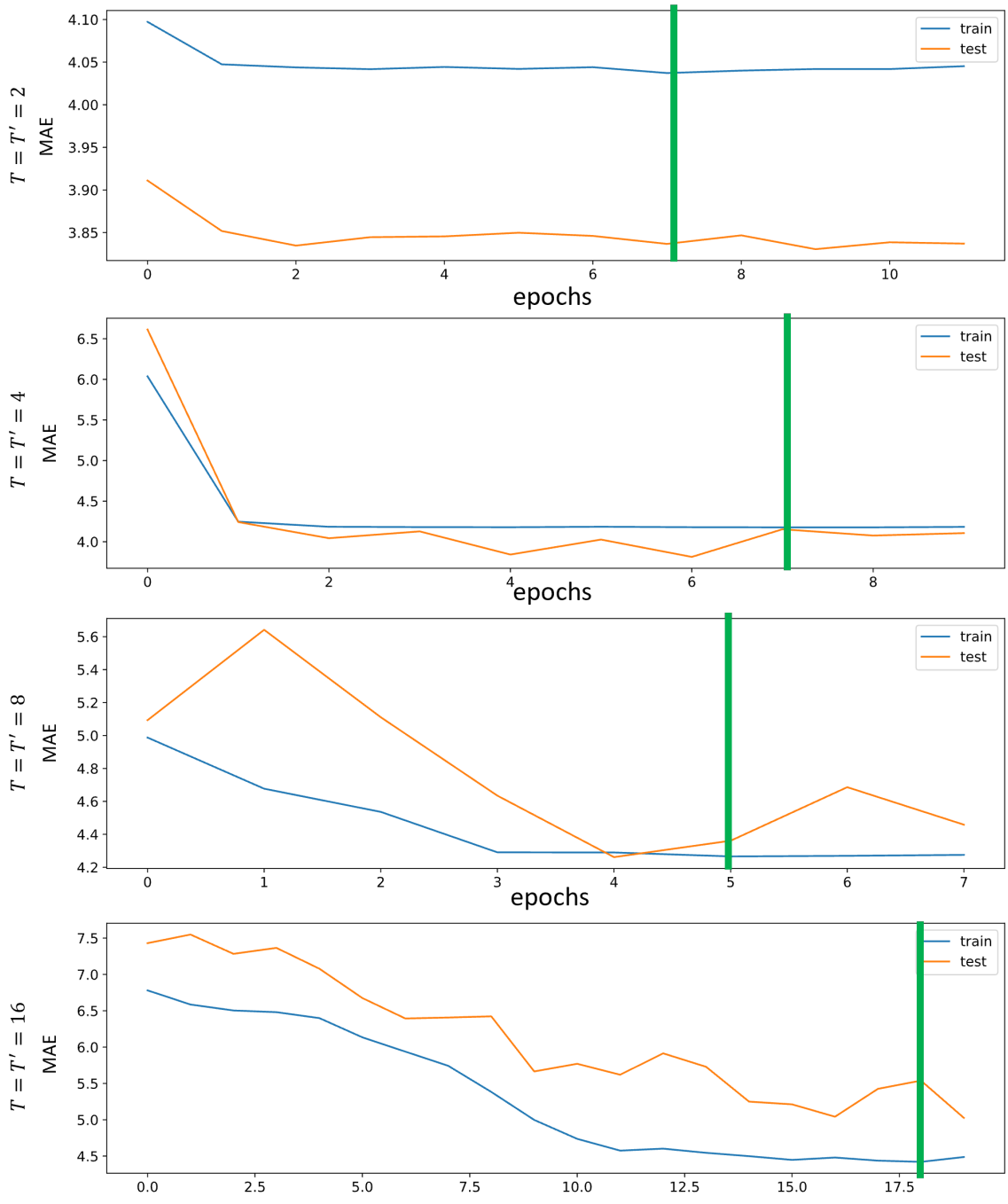


Figure 6.19: Evolución del MAE en el entrenamiento de la configuración 4 de la red neuronal recurrente convolucional de difusión en grafos (GNN)
Cada gráfico corresponde con un valor de $T$. El eje $x$ representa las *epochs*. La linea verde indica la *epoch* con menor MAE en los datos entrenamiento y cuyo modelo se ha seleccionado.

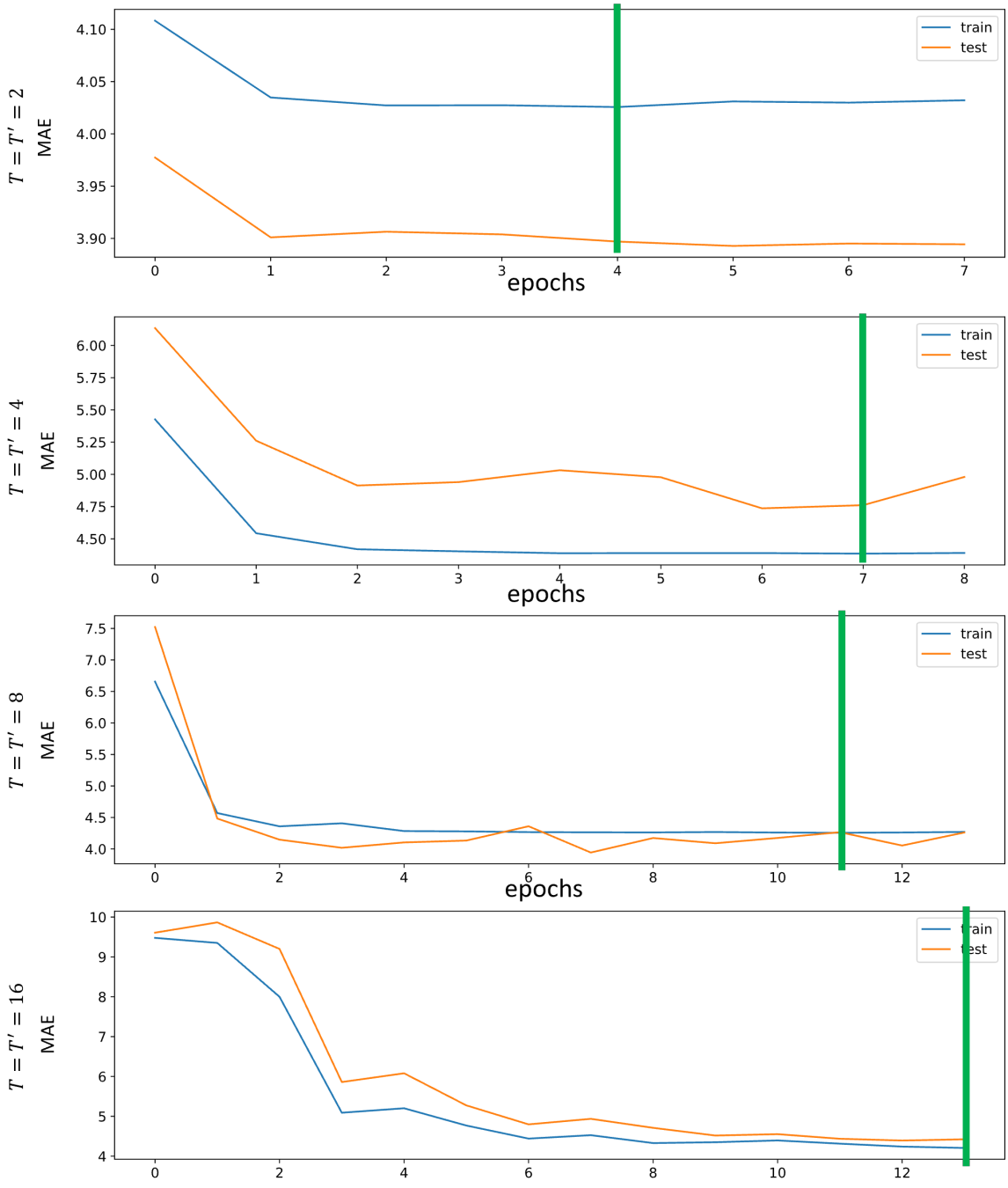


Figure 6.20: Evolución del MAE en el entrenamiento de la configuración 5 de la red neuronal recurrente convolucional de difusión en grafos (GNN)
Cada gráfico corresponde con un valor de $T$. El eje $x$ representa las *epochs*. La linea verde indica la *epoch* con menor MAE en los datos entrenamiento y cuyo modelo se ha seleccionado.

# Capítulo 7

# Resultados

Compararemos los modelos a corto y largo plazo por separado.

## 7.1 Análisis cuantitativo

Tomamos los sensores elegidos en 3.1.6 y entrenamos los modelos siguiendo el procedimiento indicado en 4.2.2:

1. Entrenar con todos los datos hasta mayo de 2021 y evaluar en junio del mismo año.
2. Entrenar con todos los datos hasta junio de 2021 y evaluar en julio del mismo año.
3. Repetir el mismo paso hasta entrenar con los datos hasta noviembre de 2021 y evaluar en diciembre del mismo año.

Así tenemos 8 cifras de error absoluto medio y error cuadrático medio, para cada modelo y estimador. Calculamos la media de cada modelo en cada estimador. Las tablas de errores no están incluidas en este documento, pero se pueden consultar, en formato *CSV* en el repositorio Sobrini García (2022), en el apartado `results`. Un resumen de los resultados lo podemos ver a continuación (**recordamos que los nombres de los modelos se puede consultar al principio del capítulo 6**).

### 7.1.1 Modelos a largo plazo

En la tabla 7.1 se pueden ver los errores absoluto y cuadrático medio para cada modelo a largo plazo. En la tabla 7.2 vemos para cuántos sensores el modelo correspondiente es el mejor modelo a largo plazo.

Los modelos a largo plazo que consiguen los errores promedios más bajos son:

- Para el MAE, el regresor de **mediana** por hora del día y según si es día laborable o no (6.1.2.2, punto 2).

| Modelo | MAE | MSE |
|---|---|---|
| mean | 7.0975 | 194.3968 |
| median | 6.8177 | 210.3351 |
| daytime_mean | 5.511 | 152.7654 |
| daytime_mean_ww | 5.0176 | 141.3132 |
| daytime_median | 5.6788 | 178.4711 |
| daytime_median_ww | 4.9121 | 162.4776 |
| lin-reg1 | 5.5013 | 143.3149 |
| lin-reg2 | 5.4948 | 143.3022 |
| xgboost | 5.1732 | 156.8019 |

Table 7.1: MAE y MSE de cada modelo a largo plazo.
La media del error se ha calculado a partir del error en ocho entrenamientos y predicciones diferentes.

| Modelo | Mejor modelo según MAE | Mejor modelo según MSE |
|---|---|---|
| mean | 0 | 0 |
| median | 0 | 0 |
| daytime_mean | 0 | 1 |
| daytime_mean_ww | 3 | 13 |
| daytime_median | 0 | 0 |
| daytime_median_ww | 20 | 6 |
| lin-reg1 | 0 | 1 |
| lin-reg2 | 0 | 1 |
| xgboost | 14 | 15 |
| best_model | 0 | 0 |

Table 7.2: Mejores modelos a largo plazo según MAE y MSE.
El valor de cada celda indica la cantidad de sensores que tienen con el modelo correspondiente el mínimo error (MAE o MSE).

| Modelo | MAE | MSE |
|---|---|---|
| repeat | 3.6618 | 58.0303 |
| repeat_last | 3.4518 | 49.657 |
| drift | 4.9806 | 105.8268 |
| gnn_config1 | 4.372 | 71.7795 |
| gnn_config2 | 4.3881 | 72.1211 |
| gnn_config3 | 4.4743 | 74.1041 |
| gnn_config4 | 4.37 | 73.126 |
| gnn_config5 | 4.4563 | 73.4416 |

Table 7.3: MAE y MSE de cada modelo a corto plazo con $T = T' = 2$.
La media del error se ha calculado a partir del error en ocho entrenamientos y predicciones diferentes.

- Para el MSE,el regresor de **media** por hora del día y según si es día laborable o no (6.1.2.2, punto 1).

Es decir, tanto el MAE como el MSE han *elegido* un modelo con el mismo planteamiento, pero usando mediana y media respectivamente. Si revisamos la literatura, vemos en Hyndman and Athanasopoulos (2021b) que optimizar el MAE lleva a pronósticos cercanos a la mediana, mientras que optimitar el MSE lleva a pronósticos cercanos a la media.

Sin embargo, en la tabla 7.2 vemos que los modelos de XGBoost han conseguido también los mejores MAE y MSE en 14 y 15 sensores, respectivamente. Los dos modelos de regresión lineal también han conseguido destacar, cada uno en uno de los sensores.

### 7.1.2 Modelos a corto plazo

En las tablas 7.3, 7.5, 7.7 y 7.9 se pueden ver los errores MAE y MSE para los modelos a corto plazo, con 2, 4 8 y 16 períodos (es decir, media hora, una hora, dos horas y cuatro horas respectivamente). En las tablas 7.4, 7.6, 7.8 y 7.10 vemos los el número de sensores para los cuales cada modelos ha sido el mejor.

#### 7.1.2.1 Cuando $T = T' = 2$

Vemos los resultados en 7.3 y 7.4.

Cuando $T = T' = 2$, el mejor modelo, según ambos errores, es aquel que repite siempre el último valor. Además hay bastante diferencia entre el error cuadrático medio de este modelo y los demás. Nuestra intuición nos dice que, para un período de tiempo tan corto (media ahora), los demás modelos (más complejos) no consiguen aprender información más valiosa que la única información que tiene el regresor de repetición: que el tráfico suele ser parecido al del instante anterior observado.

| Modelo | Mejor modelo según MAE | Mejor modelo según MSE |
|---|---|---|
| repeat | 3 | 2 |
| repeat_last | 30 | 19 |
| drift | 0 | 0 |
| gnn_config1 | 1 | 5 |
| gnn_config2 | 2 | 3 |
| gnn_config3 | 1 | 1 |
| gnn_config4 | 0 | 6 |
| gnn_config5 | 0 | 1 |
| best_model | 0 | 0 |

Table 7.4: Mejores modelos a corto plazo, para $T = T' = 2$ según MAE y MSE.
El valor de cada celda indica la cantidad de sensores que tienen con el modelo correspondiente el mínimo error (MAE o MSE).

| Modelo | MAE | MSE |
|---|---|---|
| repeat | 3.8952 | 62.8005 |
| repeat_last | 3.6209 | 55.1738 |
| drift | 4.8517 | 100.1884 |
| gnn_config1 | 4.307 | 64.8145 |
| gnn_config2 | 3.8596 | 56.7963 |
| gnn_config3 | 3.762 | 54.8122 |
| gnn_config4 | 3.8699 | 57.9633 |
| gnn_config5 | 4.0193 | 63.232 |

Table 7.5: MAE y MSE de cada modelo a corto plazo con $T = T' = 4$.
La media del error se ha calculado a partir del error en ocho entrenamientos y predicciones diferentes.

En la tabla 7.4 vemos que todas las configuraciones de GNN han conseguido sobreponerse en alguno de los sensores para MSE.

#### 7.1.2.2 Cuando $T = T' = 4$

Cuando $T = T' = 4$, los regresores de repetición pierden poder, mientras que los de GNN adquieren información sobre más observaciones que antes y, por lo tanto, mejores resultados. Los mejores modelos son:

- Según el MAE, el regresor de repetición del último valor.
- Según el MSE, la tercera configuración de GNN. Es interesante observar que, como se ha indicado en 6.4.4, la tercera configuración es la más sencilla de todas las de GNN que hemos probado.

En la tabla 7.6 vemos que tres de las configuraciones GNN han conseguido los mejores resultados en 31 de los 37 sensores para MSE. Sin embargo, para MAE, la mayoría de sensores obtienen una mejor predicción con el regresor de repetición.

| Modelo | Mejor modelo según MAE | Mejor modelo según MSE |
|---|---|---|
| repeat | 0 | 0 |
| repeat_last | 19 | 6 |
| drift | 0 | 0 |
| gnn_config1 | 0 | 0 |
| gnn_config2 | 2 | 5 |
| gnn_config3 | 5 | 14 |
| gnn_config4 | 5 | 3 |
| gnn_config5 | 6 | 9 |
| best_model | 0 | 0 |

Table 7.6: Mejores modelos a corto plazo, para $T = T' = 4$ según MAE y MSE.
El valor de cada celda indica la cantidad de sensores que tienen con el modelo correspondiente el mínimo error (MAE o MSE).

| Modelo | MAE | MSE |
|---|---|---|
| repeat | 4.533 | 75.9262 |
| repeat_last | 3.9508 | 62.0473 |
| drift | 4.9635 | 103.4157 |
| gnn_config1 | 4.9058 | 81.7841 |
| gnn_config2 | 4.1973 | 65.7112 |
| gnn_config3 | 4.4847 | 70.1129 |
| gnn_config4 | 4.2991 | 72.2839 |
| gnn_config5 | 4.5359 | 74.6174 |

Table 7.7: MAE y MSE de cada modelo a corto plazo con $T = T' = 8$.
La media del error se ha calculado a partir del error en ocho entrenamientos y predicciones diferentes.

#### 7.1.2.3 Cuando $T = T' = 8$

Cuando $T = T' = 8$, en contra de lo que esperábamos, el regresor de repetición es el que consigue los mejores resultados, tanto para MAE, como para MSE.

Sin embargo, en la tabla 7.8 vemos que la mayoría de los sensores consiguen mejores resultados con modelos GNN, sobre todo para MSE.

#### 7.1.2.4 Cuando $T = T' = 16$

Es en este último caso es cuando algunos de los modelos de GNN tienen más ventaja, ya que pueden aprender más información de los pasos anteriores. Los mejores resultados los da, tanto para el MAE, como para el MSE, la segunda configuración de GNN. Aún sigue habiendo algunos sensores (7 para MAE, 6 para MSE) que consiguen mejores resultados con el predictor de repetición del último valor.

| Modelo | Mejor modelo según MAE | Mejor modelo según MSE |
|---|---|---|
| repeat | 0 | 0 |
| repeat_last | 14 | 8 |
| drift | 0 | 0 |
| gnn_config1 | 1 | 1 |
| gnn_config2 | 9 | 7 |
| gnn_config3 | 0 | 5 |
| gnn_config4 | 6 | 8 |
| gnn_config5 | 7 | 8 |
| best_model | 0 | 0 |

Table 7.8: Mejores modelos a corto plazo, para $T = T' = 8$ según MAE y MSE.
El valor de cada celda indica la cantidad de sensores que tienen con el modelo correspondiente el mínimo error (MAE o MSE).

| Modelo | MAE | MSE |
|---|---|---|
| repeat | 5.5595 | 101.7854 |
| repeat_last | 4.5492 | 76.5235 |
| drift | 5.6791 | 127.2319 |
| gnn_config1 | 4.2826 | 70.0539 |
| gnn_config2 | 4.2006 | 65.9451 |
| gnn_config3 | 4.5236 | 72.4112 |
| gnn_config4 | 4.6311 | 78.4796 |
| gnn_config5 | 4.8077 | 86.6361 |

Table 7.9: MAE y MSE de cada modelo a corto plazo con $T = T' = 16$.
La media del error se ha calculado a partir del error en ocho entrenamientos y predicciones diferentes.

| Modelo | Mejor modelo según MAE | Mejor modelo según MSE |
|---|---|---|
| repeat | 0 | 0 |
| repeat_last | 7 | 6 |
| drift | 0 | 0 |
| gnn_config1 | 11 | 10 |
| gnn_config2 | 3 | 3 |
| gnn_config3 | 9 | 13 |
| gnn_config4 | 3 | 1 |
| gnn_config5 | 4 | 4 |
| best_model | 0 | 0 |

Table 7.10: Mejores modelos a corto plazo, para $T = T' = 16$ según MAE y MSE.
El valor de cada celda indica la cantidad de sensores que tienen con el modelo correspondiente el mínimo error (MAE o MSE).

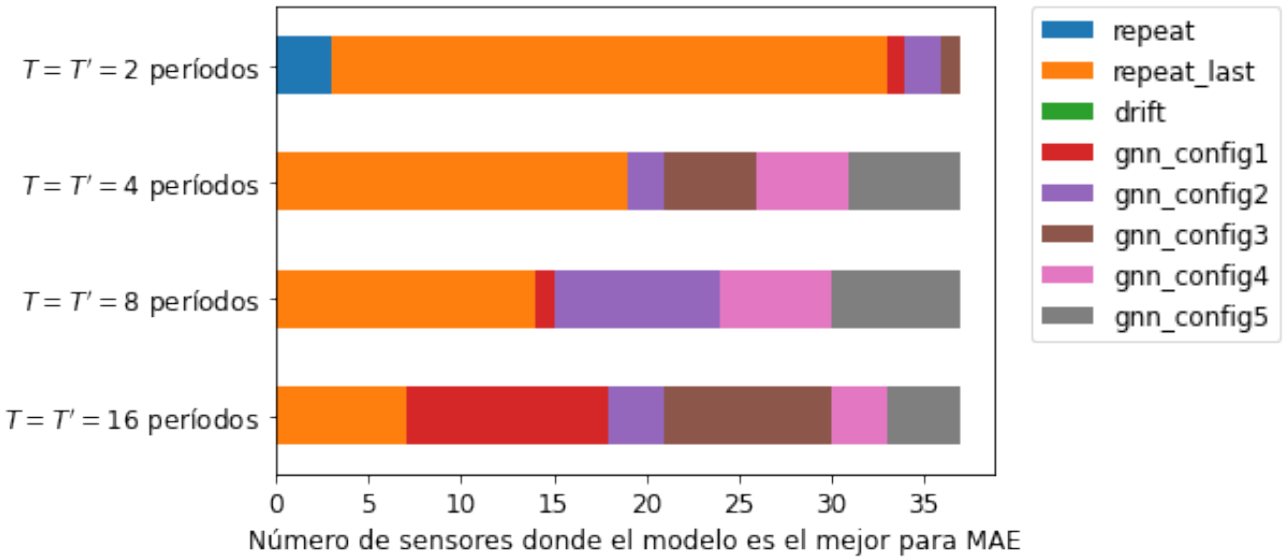


Figure 7.1: Mejores modelos a corto plazo para MAE
Cada barra representa un tipo de problema (predicción con dos, cuatro, ocho o dieciséis períodos de adelanto). Los colores representar un tipo de modelo y, el tamaño de las barras, el número de veces que dicho modelo ha sido el mejor en el problema correspondiente.

#### 7.1.2.5 Resumen de los mejores modelos a corto plazo

De una manera más visual, podemos observar cómo evolucionan los mejores modelos entre $T = T' = 2$ y $T = T' = 16$, para MAE (figura 7.1) y para MSE (figura 7.2). Vemos que, para MSE, los modelos GNN tienen un papel muy importante para todos los $T$, mientra que, para MAE, $T$ tiene que ser al menos 8 para que los modelos GNN destaquen ante el regresor de repetición del último valor,

## 7.2 Análisis cualitativo

Ahora vemos algunos ejemplos de los resultados de los modelos en la práctica.

En las figuras 7.3 y 7.4 vemos dos ejemplos de las predicciones de las dos configuraciones de regresión lineal y de XGBoost en dos sensores diferentes. Recordamos que la regresión lineal utiliza la variable de las horas en forma de splines y que, por lo tanto, la curva es más suave.

Predicciones a largo plazo.

Predicciones a largo plazo.

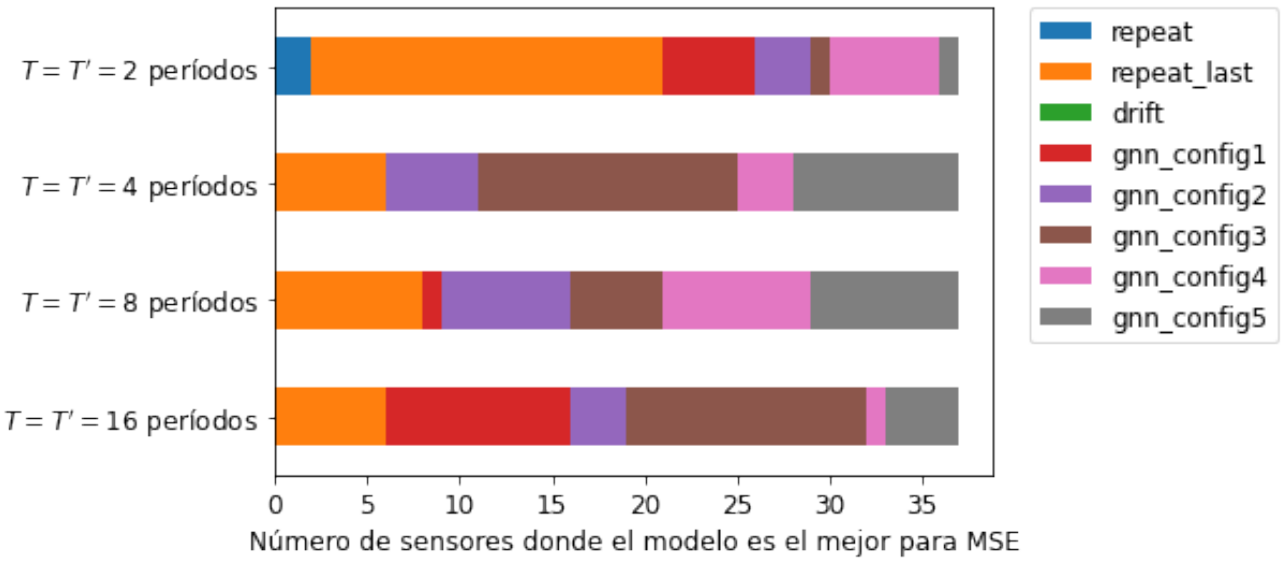


Figure 7.2: Mejores modelos a corto plazo para MSE
Cada barra representa un tipo de problema (predicción con dos, cuatro, ocho o dieciséis períodos de adelanto). Los colores representar un tipo de modelo y, el tamaño de las barras, el número de veces que dicho modelo ha sido el mejor en el problema correspondiente.

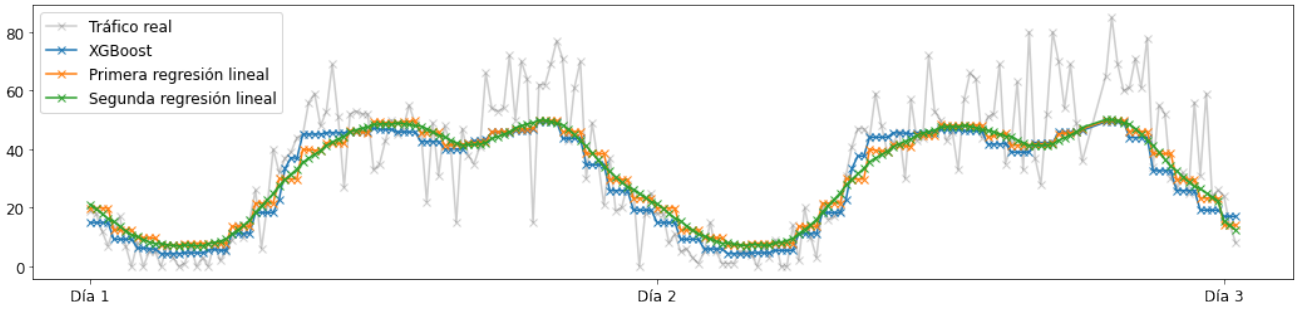


Figure 7.3: Los dos modelos de regresión lineal y XGBoost en un ejemplo

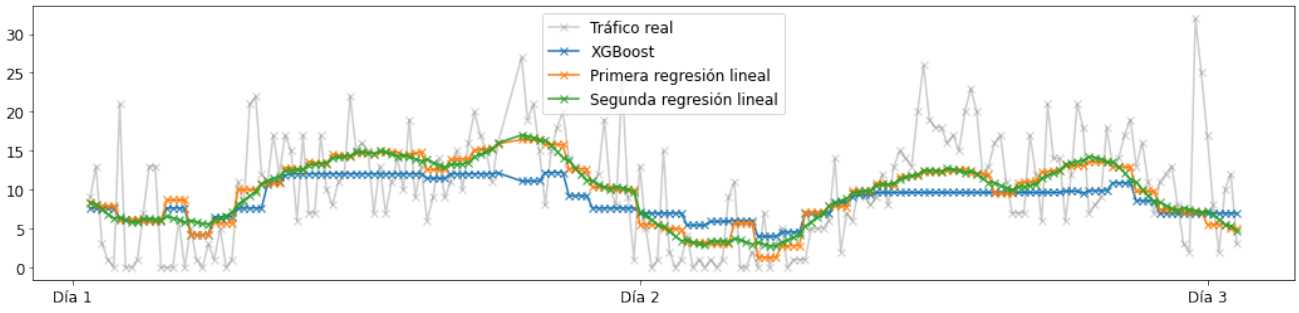


Figure 7.4: Los dos modelos de regresión lineal y XGBoost en un segundo ejemplo

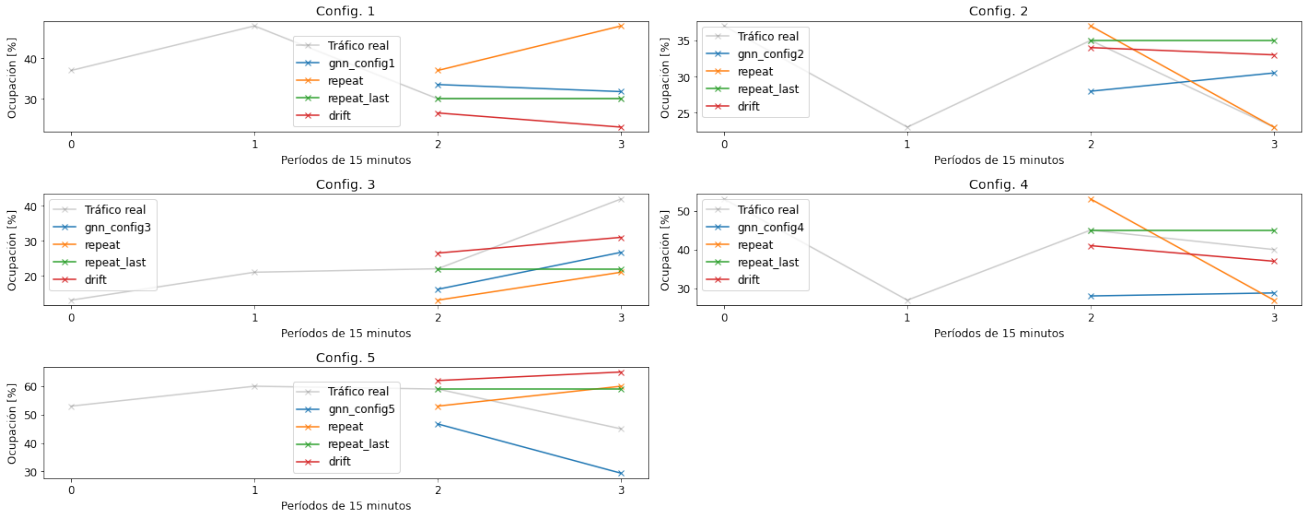


Figure 7.5: Las cinco configuraciones de GNN en ejemplos en un mismo sensor cuando $T = T' = 2$
La línea gris representa los valores reales de tráfico. La línea azul representa las predicciones, que se realizan con los datos de la primera mitad.

En cuanto a los modelos a corto plazo, en las figuras 7.5, 7.6, 7.7 y 7.8 vemos ejemplos de predicciones con las cinco configuraciones diferentes, con cada uno de los valores de $T = T'$, comparados con los modelos de referencia de repetición completa (`repeat`, 6.1.3.1), repetición del último valor (`repeat_last`, 6.1.3.2) y variación (`drift`, 6.1.4).

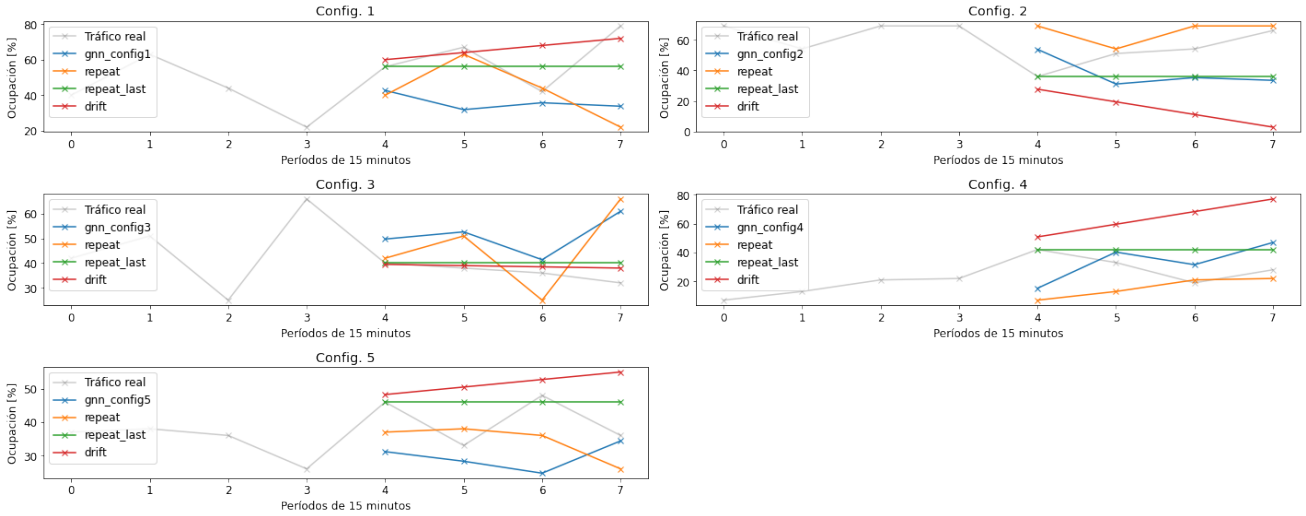


Figure 7.6: Las cinco configuraciones de GNN en ejemplos en un mismo sensor cuando $T = T' = 4$
La línea gris representa los valores reales de tráfico. La línea azul representa las predicciones, que se realizan con los datos de la primera mitad.

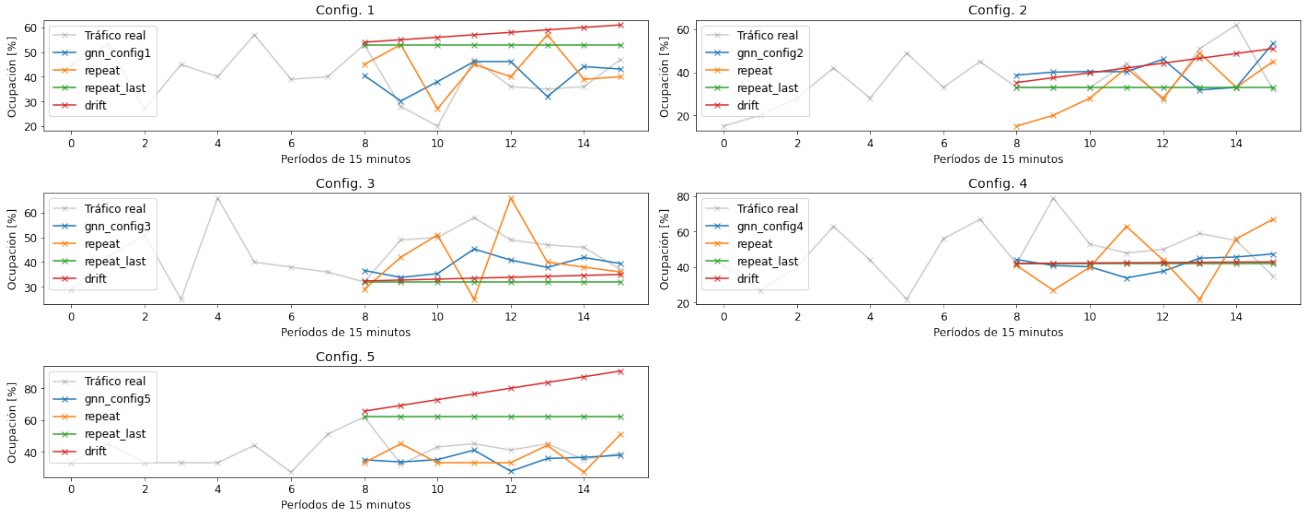


Figure 7.7: Las cinco configuraciones de GNN en ejemplos en un mismo sensor cuando $T = T' = 8$
´La línea gris representa los valores reales de tráfico. La línea azul representa las predicciones, que se realizan con los datos de la primera mitad.

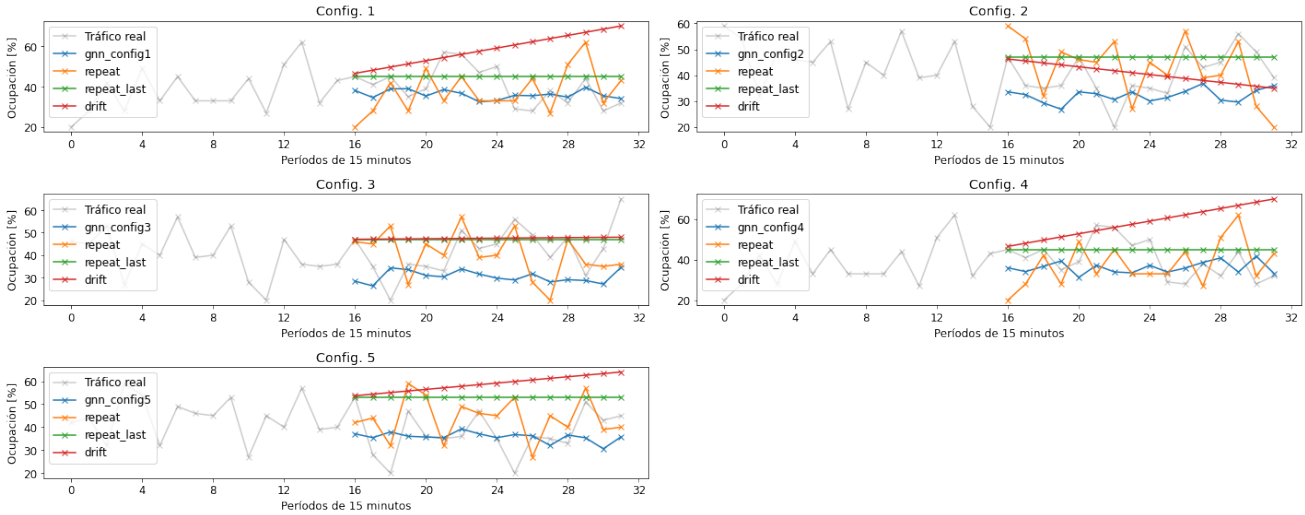


Figure 7.8: Las cinco configuraciones de GNN en ejemplos en un mismo sensor cuando $T = T' = 16$

La línea gris representa los valores reales de tráfico. La línea azul representa las predicciones, que se realizan con los datos de la primera mitad.

# Capítulo 8

# Conclusiones y trabajos futuros

## 8.1 Conclusiones

El objetivo de este trabajo era comparar modelos con diferentes grados de dificultad y que utilizan diferentes tipos de variables, para resolver dos problemas: la predicción del tráfico a corto y a largo plazo. Los datos que se han utilizado son los del Portal de datos abiertos del Ayuntamiento de Madrid, para los años 2019, 2020 y 2021.

### 8.1.1 Predicción a largo plazo

Debido a que los modelos a largo plazo no pueden utilizar observaciones recientes para realizar las predicciones, obtienen resultados peores que los de a corto plazo. Esto es algo que podíamos esperar, ya que su objetivo consiste más en encontrar y entender los factores que regulan el tráfico. Dentro de estos modelos, los modelos de referencia que calculan la media o mediana por hora y tipo de día (laborable o no), para luego realizar sus predicciones en base a simplemente estas dos variables, han conseguido muy buenos resultados, en términos de errores absolutos y cuadráticos medios promedio sobre todos los sensores. Sin embargo, si miramos los sensores uno a uno, vemos que algo menos de la mitad han conseguido los mejores resultados con XGBoost.

La razón por la que estos modelos tan sencillos han conseguido, en general, mejores resultados que modelos mucho más complejos, es probablemente que los más complejos utilizan variables que no están dando información importante para realizar las predicciones en datos nuevos para el modelo, propiciando el sobreajuste u *overfitting*.

### 8.1.2 Predicción a corto plazo

Los modelos a corto plazo han tenido resultados muy diferentes, dependiendo del número de observaciones que se utilicen para realizar la predicción. A medida que este número ha aumentado (desde 2 períodos hasta 16), los modelos con arquitectura de red neuronal de grafos han

ido ganando importancia, ya que han tenido más información para poder realizar sus predicciones. Sin embargo, al realizar el estudio cualitativo de los resultados, hemos visto patrones de predicción que habría que estudiar más en detalle, ya que no siguen una lógica que podamos reconocer a simple vista.

## 8.2 Trabajos futuros

Hay múltiples aspectos de los modelos con los que se ha trabajado que podrían estudiarse con más detalle, con el objetivo de introducir modificaciones que mejoren el resultado:

### 8.2.1 Variables explicativas

En este trabajo se han realizado múltiples pruebas para obtener las variables con las que los resultados son los mejores. Sin embargo, estas variables obtenidas son en ocasiones de dudosa relevancia. En los modelos de regresión lineal, hemos visto que, de hecho, estas variables tienen coeficientes casi despreciables. Sin embargo, en XGBoost hemos comprobado que el modelo le da una importancia sorprendentemente grande a las variables meteorológicas. Dado que estas tienen cierta correlación con las variables temporales (de día, la temperatura es más alta, por ejemplo), es lógico pensar que el modelo XGBoost está aprendiendo la información temporal a partir de la meteorológica. Esto está muy probablemente degradando los resultados y es preciso realizar un análisis de las variables para elegir las que dan más información a XGBoost.

### 8.2.2 Red neuronal de grafos con difusión convolucional

En este trabajo se ha adaptado la red de Ibe (2013) para que pueda usar otro tipo de variables diferentes al tráfico. Sin embargo, con la arquitectura de la red, el modelo no ha sabido extraer la información relevante para obtener mejores resultados. Será interesante adaptar la red y los parámetros para mejorar los resultados.

### 8.2.3 Tratamiento de los valores nulos

El planteamiento en este trabajo respecto a los valores nulos ha sido ignorarlos, lo que ha causado, en ocasiones, problemas de disponibilidad de datos. Sería interesante utilizar otros enfoques. Por ejemplo, una interpolación de *K-nearest-neighbors*. En XGBoost hay una opción para entrenar con datos que contienen nulos (Chen and Guestrin (2016)). También sería interesante explorar esta opción.

# Bibliografía


Belhadi, A., Djenouri, Y., Djenouri, D., and Lin, J. C.-W. (2020). A recurrent neural network for urban long-term traffic flow forecasting. *Applied Intelligence*, 50(10):3252–3265.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on knowledge discovery and data mining*, volume 13-17- of *KDD '16*, pages 785–794. ACM.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches.

Dong, H., Ding, F., Tan, H., and Zhang, H. (2022). Laplacian integration of graph convolutional network with tensor completion for traffic prediction with missing data in inter-city highway network. *Physica A*, 586:126474.

Dong, X., Lei, T., Jin, S., and Hou, Z. (2018). Short-term traffic flow prediction based on xgboost. In *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 854–859.

Dougherty, M. S. and Cobbett, M. R. (1997). Short-term inter-urban traffic forecasts using neural networks. *International journal of forecasting*, 13(1):21–31.

Eric, C. S. (2021). Discrete temporal dynamic graph with recurrent structure. `https://github.com/dmlc/dgl/tree/master/examples/pytorch/dtgrnn`.

Fisher, C. (2020). Google maps is improving travel etas with deepmind ai. *Engadget*.

Freedman, D. (2009). Multiple regression. In *Statistical models : theory and practice*, chapter 4. Second edition. edition.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Special invited paper. additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–374.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of statistics*, 29(5):1189–1232.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Hanson, S. J. (1990). A stochastic version of the delta rule. *Physica D: Nonlinear Phenomena*, 42(1):265–272.

Hyndman, R. and Athanasopoulos, G. (2021a). Arima models. In *Forecasting: Principles and Practice*, chapter 9. OTexts, Australia, 3rd edition.

Hyndman, R. and Athanasopoulos, G. (2021b). Evaluating point forecast accuracy. In *Forecasting: Principles and Practice*, chapter 5.8. OTexts, Australia, 3rd edition.

Hyndman, R. and Athanasopoulos, G. (2021c). Time series cross-validation. In *Forecasting: Principles and Practice*, chapter 5. OTexts, Australia, 3rd edition.

Ibe, O. C. O. C. (2013). Diffusion processes. In *Elements of random walk and diffusion processes*, Wiley series in operations research and management science, chapter 7. John Wiley & Sons, Inc., Hoboken, N.J.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. `https://arxiv.org/abs/1502.03167`.

Jain, A. (2016). Xgboost parameters: Xgboost parameter tuning. `https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/`.

Jordahl, K., den Bossche, J. V., Fleischmann, M., Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D., Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur, A., Rey, S., Ren, C., Arribas-Bel, D., Wasser, L., Wolf, L. J., Journois, M., Wilson, J., Greenhall, A., Holdgraf, C., Filipe, and Leblanc, F. (2020). geopandas/geopandas: v0.8.1. `https://doi.org/10.5281/zenodo.3946761`.

Kamada, T. and Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15.

Karim, A., Abdellah, A., and Hamid, S. (2019). Long-term traffic flow forecasting based on an artificial neural network. *Advances in Science, Technology and Engineering Systems Journal*, 4.

Karlaftis, M. and Vlahogianni, E. (2011). Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation research. Part C, Emerging technologies*, 19(3):387–399.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

Lange, O. and Perez, L. (2020). Traffic prediction with advanced graph neural networks. `https://www.deepmind.com/blog/traffic-prediction-with-advanced-graph-neural-networks`.

Lee, S. and Fambro, D. B. (1999). Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. *Transportation Research Record*, 1678(1):179–188.

Li, C.-S. and Chen, M.-C. (2012). Identifying important variables for predicting travel time of freeway with non-recurrent congestion with neural networks. *Neural computing & applications*, 23(6):1611–1629.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting.

Liu, P., Zhang, Y., Kong, D., and Yin, B. (2019). Improved spatio-temporal residual networks for bus traffic flow prediction. *Applied Sciences*, 9(4).

Lu, Z., Xia, J., Wang, M., Nie, Q., and Ou, J. (2020). Short-term traffic flow forecasting via multi-regime modeling and ensemble learning. *Applied Sciences*, 10(1).

Manibardo, E. L., Laña, I., and Ser, J. D. (2020). Deep learning for road traffic forecasting: Does it make a difference? *CoRR*, abs/2012.02260.

OpenStreetMap contributors (2017). Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`.

Osipov, V., Nikiforov, V., Zhukova, N., and Miloserdov, D. (2020). Urban traffic flows forecasting by recurrent neural networks with spiral structures of layers. *Neural computing & applications*, 32(18):14885–14897.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

python visualization (2020). Folium. `https://python-visualization.github.io/folium/`.

Rifkin, R. and Lippert, R. (2007). Notes on regularized least squares.

Samoili, S. and Dumont, A.-G. (2012). Framework for real-time traffic forecasting methodology under exogenous parameters.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98.

Sobrini García, E. (2022). Repositorio del tfm. `https://github.com/elena-sg/madrid-traffic`.

Sun, S., Wu, H., and Xiang, L. (2020). City-wide traffic flow forecasting using a deep convolutional neural network. *Sensors*, 20(2).

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. `https://arxiv.org/abs/1409.3215`.

Tampubolon, H. and Hsiung, P.-A. (2018). Supervised deep learning based for traffic flow prediction. In *2018 International Conference on Smart Green Technology in Electrical and Information Systems (ICSGTEIS)*, pages 95–100.

Van Der Voort, M., Dougherty, M., and Watson, S. (1996). Combining kohonen maps with arima time series models to forecast traffic flow. *Transportation research. Part C, Emerging technologies*, 4(5):307–318.

van Lint, H. and van Hinsbergen, C. (2012). Short-term traffic and travel time prediction models. In *Artificial Intelligence Applications to Critical Transportation Issues*, pages 22–41. Washington, D.C.: Transportation Research Board, USA.

Wang, M., Yang, S., Sun, Y., and Gao, J. (2017). Discovering urban mobility patterns with pagerank based traffic modeling and prediction. *Physica A*, 485:23–34.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. (2019). Deep graph library: A graph-centric, highly-performant package for graph neural networks. `https://arxiv.org/abs/1909.01315`.

Wang, Z., Su, X., and Ding, Z. (2021). Long-term traffic prediction based on lstm encoder-decoder architecture. *IEEE Transactions on Intelligent Transportation Systems*, 22(10):6561–6571.

Williams, B. M. and Hoel, L. A. (2003). Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672.

Yang, B., Sun, S., Li, J., Lin, X., and Tian, Y. (2019). Traffic flow prediction using lstm with feature enhancement. *Neurocomputing*, 332:320–327.

Yeo, I. and Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959.

Zang, D., Fang, Y., Wang, D., Wei, Z., Tang, K., and Li, X. (2018). Long term traffic flow prediction using residual net and deconvolutional neural network. In Lai, J.-H., Liu, C.-L., Chen, X., Zhou, J., Tan, T., Zheng, N., and Zha, H., editors, *Pattern Recognition and Computer Vision*, pages 62–74, Cham. Springer International Publishing.

Zhang, Y. and Haghani, A. (2015). A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies*, 58:308–324. Big Data in Transportation and Traffic Engineering.

# Anexo A

# Transformación de las variables temporales

Las variables temporales, como el mes o la hora del día, tienen un rol muy importante en los resultados de algunos de los modelos propuestos. Por ello, es importante la manera en que transformemos estas variables o el *feature engineering*. Vamos a ver los diferentes métodos que utilizaremos, aplicados sobre el ejemplo de la hora del día, pero estos los podremos utilizar sobre otras múltiples variables.

## A.1 Mantener la variable es su forma original

La manera más directa de usar las variables temporales periódicas es usarlas tal y como están en nuestros datos originales. De esta manera, no tenemos que realizar ninguna transformación especial, ni aumentamos el número de variables explicativas en nuestro modelo.

Por ejemplo, a lo largo de dos días, la variable de las horas tiene la forma de la figura A.1.

El problema con este tipo de variable es que no estamos utilizando la naturaleza periódica de las variable. Por ejemplo, podemos suponer que no hay grandes diferencias entre la situación en una carretera a las 23:55 y a las 00:05. Sin embargo, nuestra variable no refleja que estos dos instante del día sean cercanos en el tiempo.

Por esta razón, vamos a ver otros métodos que nos permitan explotar la naturaleza temporal y periódica de algunas de nuestras variables.

## A.2 Transformación *one-hot*

En este tipo de transformación, creamos una columna para cada valor distinto dentro de una clase. El valor de la columna será 1 si el valor de la variable original es el de la columna y 0 en

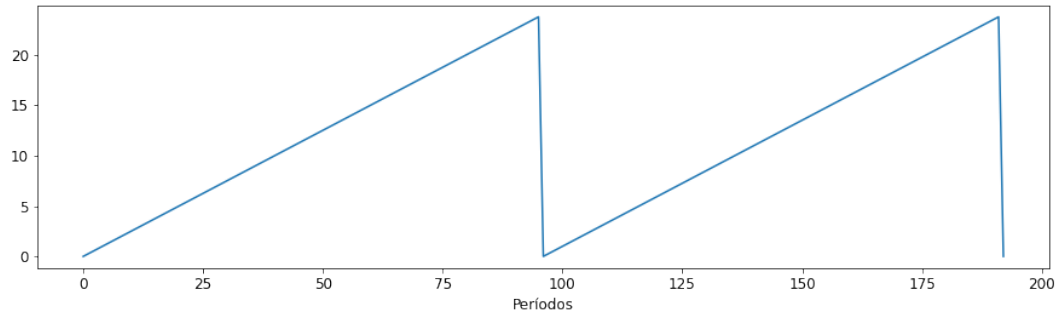


Figure A.1: Forma original de la variable hora durante dos días.

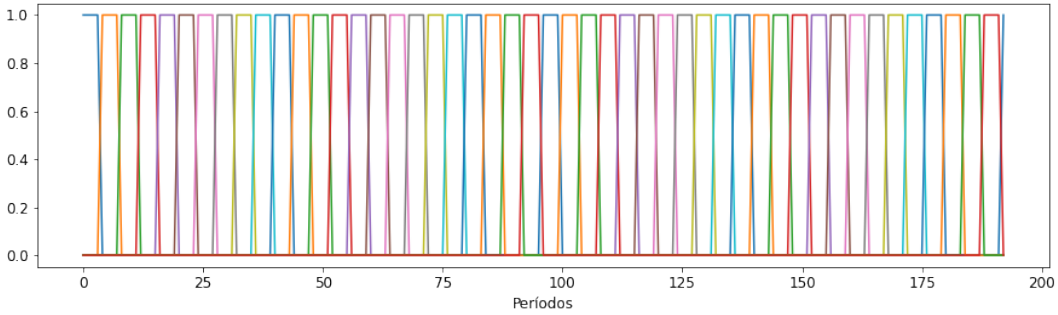


Figure A.2: Transformación one-hot sobre las horas de un día.
Cada una de las 24 líneas representa una hora.

caso contrario.

En nuestros datos, utilizamos las horas con precisión de 15 minutos, pero, antes de utilizar este tipo de transformación, truncamos el valor y mantenemos solo las horas, de tal manera que solo creamos 24 columnas adicionales. La señal de la hora durante los 192 registros de dos días (4 registros por hora $\times$ 24 horas en un día $\times$ 2 días $=$ 192 registros en dos días) tendría entonces la forma de la figura A.2.

## A.3 Transformada de Fourier

Jean-Basptiste Fourier demostró que una series de términos seno y coseno con las frecuencias adecuadas pueden aproximar cualquier función periódica. Si $m$ es el período de una variable periódica, entonces los términos de Fourier vienen dados por

$$a_{i,t} = \sin\left(\frac{2i\pi t}{m}\right), \qquad b_{i,t} = \cos\left(\frac{2i\pi t}{m}\right)$$

Por ejemplo, para las horas, tendríamos un período de $m = 24$. Si decidimos tener $2 \times 2 = 4$

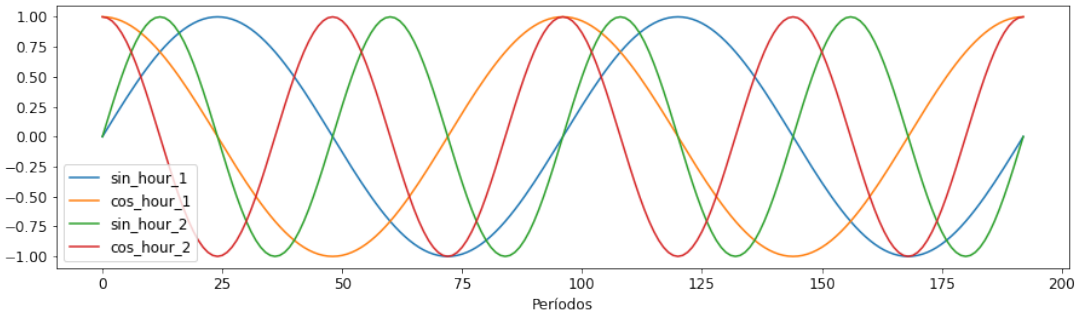


Figure A.3: Cuatro términos de la transformada de Fourier sobre las horas de dos días. Cada línea es uno de los términos de Fourier.

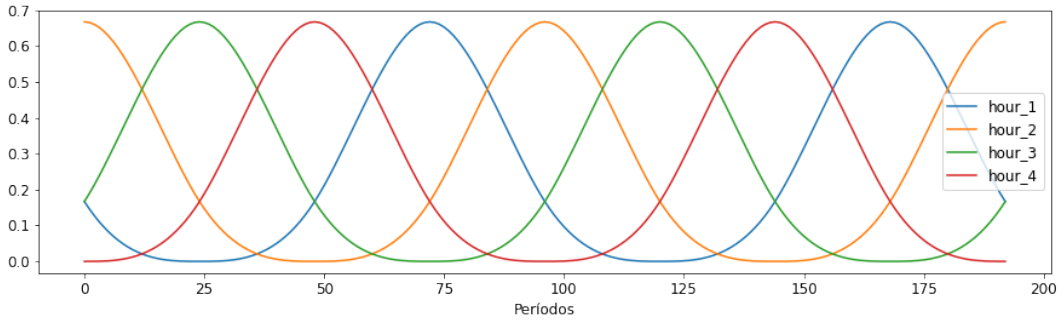


Figure A.4: Cuatro splines codificando la variable hora durante dos días.

términos para codificar la variable de las horas, tendríamos como resultado durante dos días (192 registros) la señal de la figura A.3. En comparación con el método one-hot propuesto en A.2, nos ahorraríamos $24 - 4 = 20$ variables.

En nuestro caso, vamos a utilizar la transformada de Fourier siempre con $1 \times 2$ términos.

## A.4 Splines periódicos

Este es el último tipo de transformación que vemos para las variables temporales periódicas.

Por ejemplo, si seleccionamos, igual que en A.3, cuatro variables, entonces el resultado será el de la figura A.4.

En nuestro caso, vamos a utilizar siempre un número de splines igual a la mitad del período de la variable de referencia que estemos transformando. Por ejemplo, para las horas, 12 y, para los meses, 6.

# Anexo B

# Transformación de las variables meteorológicas

Las variables meteorológicas que utilizamos son: precipitación, velocidad del viento, dirección del viento, temperatura, humedad relativa, presión barométrica y radiación solar.

De ellas, las únicas en las que vamos a probar algunas transformaciones son la precipitación y la velocidad y dirección del viento. Las otras tienen distribuciones más uniformes y, por ello, vamos a utilizarlas tal y como están.

## B.1 Precipitación

En más del 95% de las observaciones disponibles en 2019, 2020 y 2021, la precipitación es 0, y en el 99%, el valor es menor que 1.1. Sin embargo, en ocasiones nos encontramos con valores de alrededor de 20. Por ello, vamos a evaluar algunas transformaciones que uniformicen estas cantidades.

Veremos las transformaciones aplicadas a los valores en la estación 28079024, en Casa de Campo, en los días 25 y 26 de agosto de 2019, un caso en el que se observan picos en la precipitación mayores a los habituales.

Recordamos que, como las datos meteorológicos tienen periodicidad horaria, están interpoladas linealmente, para conseguir periodicidad de 15 minutos, como se ha explicado en 3.3.

### B.1.1 Mantener las variables en su forma original

Véase el ejemplo en la figura B.1. Observamos dos picos que, en comparación con el resto de los datos, están desproporcionados.

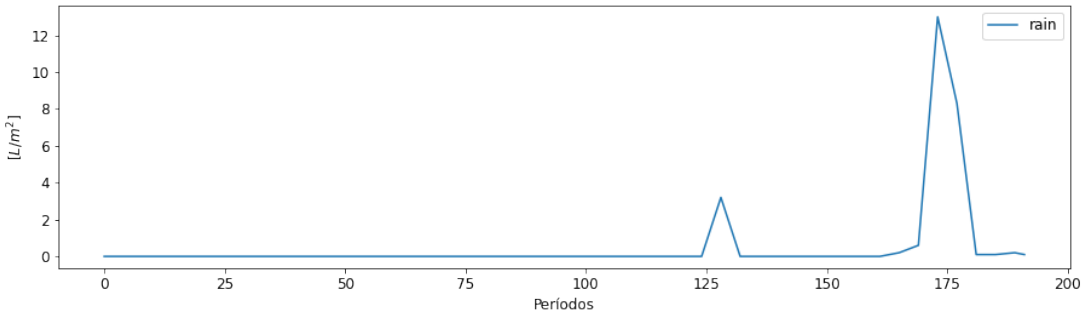


Figure B.1: Forma original de la variable precipitación durante dos días.

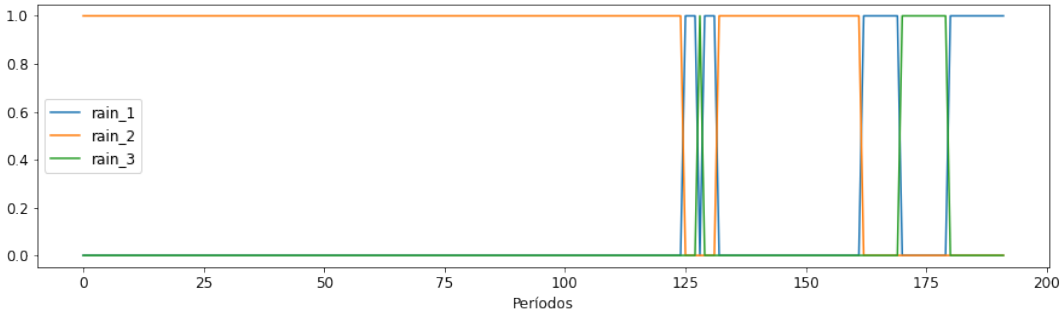


Figure B.2: Transformación *one-hot* de la variable precipitación durante dos días.

### B.1.2 Transformación *one-hot*

La primera transformación que probamos consiste en asignar una categoría a cada rango de valores y, en base a ellas, realizar transformación *one-hot*. Las categorías serán: 0 $L/m^2$, entre 0 y 2.5 $L/m^2$ y más de $2.5L/m^2$. Así, obtenemos el resultado de la figura B.2.

### B.1.3 Transformación ordinal

Podemos utilizar las mismas categorías que en B.1.2, y darles los valores 0, 1 y 2, en una sola variable. Entonces nos quedaría como resultado la figura en B.3.

### B.1.4 Transformación de potencia: Yeo-Johnson

Las transformaciones de potencia es una familia de transformaciones paramétricas y monótonas que sirven para que la correspondiente distribución se asemeje a una distribución normal. Dos de las más populares son Box-Cox, que no admite valores nulos ni negativos, y Yeo-Johnson (Yeo and Johnson (2000)), que sí los admite. Vamos a utilizar la segunda, ya que tenemos valores nulos.

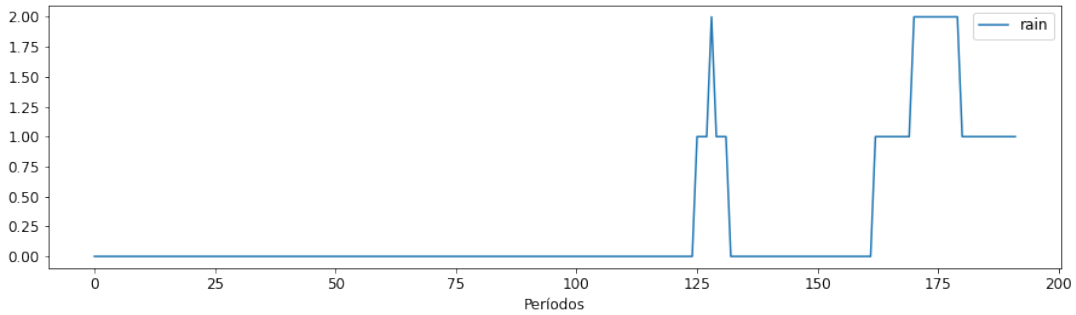


Figure B.3: Transformación ordinal de la variable precipitación durante dos días.

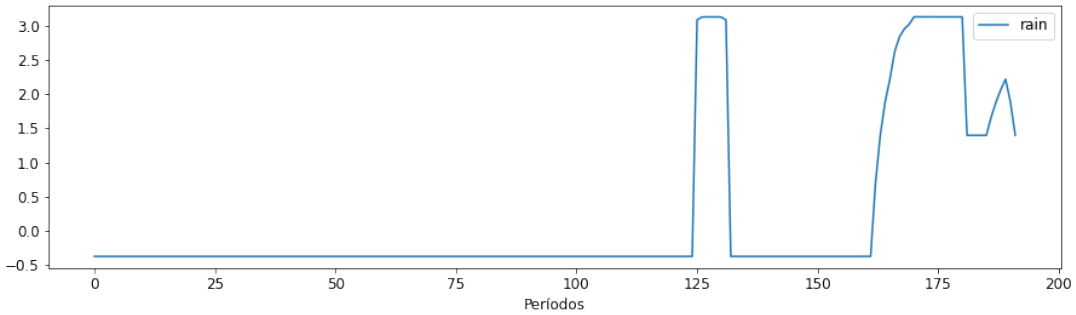


Figure B.4: Transformación de potencia de la variable precipitación durante dos días.

Esta transformación está dada por

$$x_i^{(\lambda)} = \begin{cases} [(x_i+1)^{\lambda}-1]/\lambda & \text{if } \lambda \neq 0, x_i \geq 0, \\ \ln(x_i+1) & \text{if } \lambda = 0, x_i \geq 0 \\ -[(-x_i+1)^{2-\lambda}-1]/(2-\lambda) & \text{if } \lambda \neq 2, x_i \leq 0, \\ -\ln(-x_i+1) & \text{if } \lambda = 2, x_i < 0 \end{cases}$$

donde $\lambda$ se estima por máxima verosimilitud.

El resultado de aplicar esta transformación en el mismo ejemplo se puede ver en B.4.

### B.1.5 Transformaciones de cuantiles

Este tipo de transformaciones son no paramétricas y utilizan información de los cuantiles. Primero se utiliza una estimación de la función de distribución para convertir los valores a una distribución uniforme. Utilizando la función cuantil, se transforma la variable en la dis-

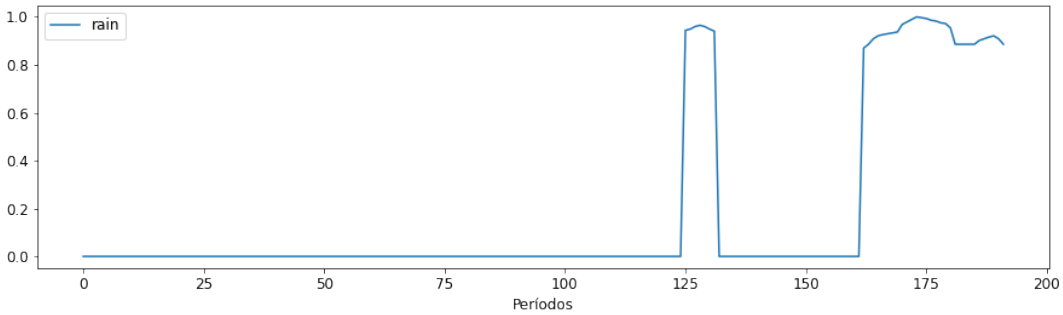


Figure B.5: Transformación cuantil con distribución uniforme de la variable precipitación durante dos días.

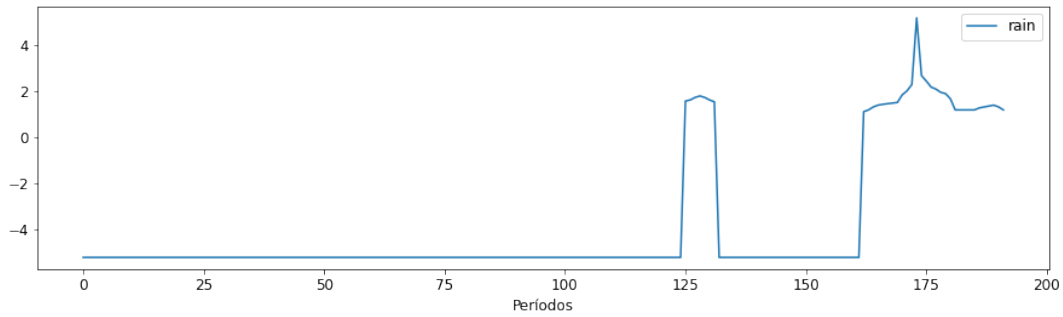


Figure B.6: Transformación cuantil con distribución normal de la variable precipitación durante dos días.

tribución objetivo. Los valores que están por encima o por debajo del rango de los datos de entrenamiento se transformará en el máximo o mínimo, respectivamente.

Utilizamos esta transformación poniendo como objetivo una transformación uniforme (figura B.5), así como una normal (figura B.6).

## B.2 Viento

En el viento no tenemos el problema que teníamos para la precipitación, ya que los valores no son tan extremos. Por ello, no aplicaremos transformaciones como las descritas en B.1.

Sin embargo, vamos a realizar otro tipo de transformación, en la que extraemos los valores

del viento en cada una de las coordenadas: $x$ e $y$. Esto lo hacemos de la siguiente manera:

$$\text{viento}_x = \text{velocidad del viento} \times \cos\left(\text{dirección del viento}\right)$$

$$\text{viento}_y = \text{velocidad del viento} \times \sin\left(\text{dirección del viento}\right)$$

A la hora de elegir el modelo, vamos a evaluar los resultados usando la variable original de la velocidad del viento, por una parte y, por otra, usando $\text{viento}_x$ y $\text{viento}_y$.