



Universidad Politécnica  
de Madrid

**Escuela Técnica Superior de  
Ingenieros Informáticos**



Grado en Ciencia de Datos e Inteligencia Artificial

Trabajo Fin de Grado

**Previsión del Precio SPOT en el  
Mercado Eléctrico Español con Redes  
Neuronales**

Autor: Nicolás Vega Muñoz  
Tutor(a): Bojan Mihaljevic

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

*Trabajo Fin de Grado*

*Grado en Ciencia de Datos e Inteligencia Artificial*

**Título:** Previsión del Precio SPOT en el Mercado Eléctrico Español con Redes Neuronales

Junio 2024

**Autor:** Nicolás Vega Muñoz

**Tutor:** Bojan Mihaljevic

Departamento de Inteligencia Artificial

Escuela Técnica Superior de Ingenieros Informáticos

Universidad Politécnica de Madrid

# **Resumen**

El presente Trabajo de Fin de Grado tiene como objetivo desarrollar un modelo capaz de predecir el precio del MWh (precio SPOT del mercado energético) para cada hora del día siguiente, optimizando así la oferta en el mercado. Dada la volatilidad del mercado energético y la influencia de numerosos factores externos, la precisión en la predicción del precio SPOT es crucial para que las empresas puedan tomar decisiones informadas sobre a cuánto ofertar y maximizar ganancias sin el riesgo de no entrar en el mercado.

Para abordar este desafío, se estudiaron y compararon diversos modelos, incluyendo tanto modelos clásicos como de Deep Learning. Entre los modelos evaluados se destacan los modelos de estado del arte, como los Temporal Fusion Transformers (TFT), que permitieron obtener predicciones con un error medio absoluto (MAE) de 1,26€. La metodología aplicada abarca desde la obtención y preprocessamiento de datos hasta el modelado, la comparación de modelos y el análisis de resultados, proporcionando así una ventaja estratégica para operar en el mercado diario de electricidad.

La implementación de este modelo no solo busca mejorar la precisión en las predicciones, sino también optimizar la gestión de la oferta energética, ayudando a las empresas a ofertar de manera más competitiva y eficiente. Además, se destaca la importancia de tecnologías avanzadas y la interpretabilidad del modelo, permitiendo a las empresas (generadoras, comercializadoras...) entender cómo y por qué se generan las predicciones, lo cual es esencial para la toma de decisiones estratégicas en un mercado eléctrico dinámico.



# **Abstract**

The objective of this Final Degree Project is to develop a model capable of predicting the price of MWh (SPOT price of the energy market) for each hour of the following day, thus optimizing market offers. Given the volatility of the energy market and the influence of numerous external factors, accuracy in predicting the SPOT price is crucial for companies to make informed decisions on how to price their offers to maximize profits without the risk of exclusion from the market.

To address this challenge, various models were studied and compared, including both classical and Deep Learning models. Among the evaluated models, state-of-the-art models such as Temporal Fusion Transformers (TFT) stand out, achieving predictions with a mean absolute error (MAE) of €1.26. The applied methodology ranges from data acquisition and preprocessing to modeling, model comparison, and results analysis, thus providing a strategic advantage for operating in the daily electricity market.

The implementation of this model aims not only to improve prediction accuracy but also to optimize energy offer management, helping companies to bid more competitively and efficiently. Additionally, the importance of advanced technologies and model interpretability is highlighted, enabling companies (generators, marketers...) to understand how and why predictions are generated, which is essential for strategic decision-making in a dynamic electricity market.



# Tabla de contenidos

<b>1. Introducción</b>	<b>1</b>
1.1. Formación de precios en el mercado mayorista diario de electricidad	2
1.2. Tecnologías . . . . .	5
1.3. Características económicas de las tecnologías de generación . . . . .	7
1.4. Objetivos . . . . .	8
1.5. Metodología . . . . .	8
1.5.1. Obtención de datos . . . . .	8
1.5.2. Modelado . . . . .	9
1.5.3. Comparativa de modelos . . . . .	9
1.5.4. Análisis de resultados . . . . .	9
1.5.5. Restricciones técnicas . . . . .	9
<b>2. Fundamentos Teóricos</b>	<b>11</b>
2.1. Caracterización del dominio . . . . .	11
2.2. Serie Temporal . . . . .	11
2.3. Modelos Clásicos . . . . .	13
2.3.1. ARIMA . . . . .	13
2.3.2. SARIMA . . . . .	13
2.3.3. SARIMAX . . . . .	14
2.4. Modelos Neuronales . . . . .	14
2.4.1. Long Short-Term Memory (LSTM) . . . . .	15
2.4.2. Redes Neuronales Convolucionales (CNN) . . . . .	17
2.5. Transformers . . . . .	18
2.5.1. Temporal Fusion Transformers (TFT) . . . . .	20
2.6. Métricas para series temporales . . . . .	27
<b>3. Desarrollo</b>	<b>31</b>
3.1. Planteamiento . . . . .	31
3.2. Obtención de datos . . . . .	31
3.3. Análisis descriptivo de variables (EDA) . . . . .	32
3.4. Preprocesado . . . . .	36
3.4.1. Tratamiento de nulos . . . . .	37
3.4.2. Manejo de valores atípicos . . . . .	37
3.4.3. Selección de variables . . . . .	37
3.4.4. Normalización de datos . . . . .	37
3.5. Modelado . . . . .	38

## TABLA DE CONTENIDOS

---

3.5.1. (S)ARIMA(X) . . . . .	39
3.5.2. LSTM . . . . .	42
3.5.3. CNN-1D . . . . .	46
3.5.4. Temporal Fusion Transformers (TFT) . . . . .	49
<b>4. Comparativa y análisis de resultados</b>	<b>59</b>
4.1. Comparativa de modelos . . . . .	59
4.2. Interpretación del modelo TFT . . . . .	62
<b>5. Análisis de impacto</b>	<b>67</b>
<b>6. Conclusiones y trabajo futuro</b>	<b>69</b>
6.1. Conclusiones . . . . .	69
6.2. Trabajo Futuro . . . . .	70
<b>Bibliografía</b>	<b>73</b>
<b>Anexos</b>	<b>77</b>
<b>A. Primer anexo</b>	<b>77</b>
<b>B. Segundo anexo</b>	<b>99</b>
<b>C. Tercer anexo</b>	<b>101</b>

# **Capítulo 1**

## **Introducción**

El mercado mayorista energético es uno complicado en el que operar debido a la volatilidad que este presenta, además de la cantidad de factores externos que influyen en su comportamiento (demanda, producción de energías renovables, precio del gas...). Debido a cómo funciona este y a cómo se realiza la formación de precios, son muchas las empresas que tienen la incertidumbre sobre a qué precio pueden vender su energía.

En el mercado mayorista energético, algunas tecnologías tienen la capacidad de ajustar su producción según la demanda, como las centrales de agua embalsada que pueden optar por no generar energía en un momento dado y hacerlo en otro momento. Sin embargo, para otras tecnologías, como las energías renovables, la energía generada debe ser consumida en el momento de su producción, de lo contrario se desperdicia. Es precisamente esta problemática la que motiva la realización de este trabajo. Las empresas generadoras que buscan vender su energía deben encontrar un equilibrio delicado: ofertar al precio más alto posible para maximizar beneficios, pero sin arriesgarse a vender por encima del precio de casación y enfrentar pérdidas significativas por la generación de energía no consumida.

Por ello, el objetivo de este trabajo es aplicar los modelos más adecuados, tanto modelos clásicos estadísticos, como modelos de estado del arte como los Temporal Fusion Transformers (TFT), para ser capaces de predecir en el día 'D' el precio de venta de la energía (precio SPOT<sup>1</sup>) en el día 'D+1'.

En primer lugar, para comprender la motivación de este trabajo, debemos dar un contexto sobre el funcionamiento del mercado mayorista energético en España así como de las energías y tecnologías existentes.

---

<sup>1</sup>El precio SPOT es el precio de mercado actual al cual se puede comprar o vender un activo, en este caso, energía, para entrega inmediata. Este precio es lo contrario al precio futuro o forward price, donde los contratos se realizan ahora, pero la transacción y el pago ocurrirán en una fecha posterior.

## Capítulo 1. Introducción

### 1.1. Formación de precios en el mercado mayorista diario de electricidad

El mercado eléctrico es el conjunto de agentes y plataformas de negociación en las que se produce la compra y venta de energía eléctrica.

En España, al igual que en otros países, se tiene una secuencia de mercados que permite tanto a vendedores como a compradores intercambiar contratos con distintos horizontes temporales, desde el mercado intradiario a días, meses o años antes de que la energía sea tan siquiera generada. En esta secuencia de mercados actúan distintos gestores que orquestan las subastas de distintos productos. Este trabajo se desarrolla en el mercado diario, por lo que nuestro objetivo es obtener predicciones precisas del precio SPOT para el día siguiente, proporcionando así una ventaja estratégica para operar en él.

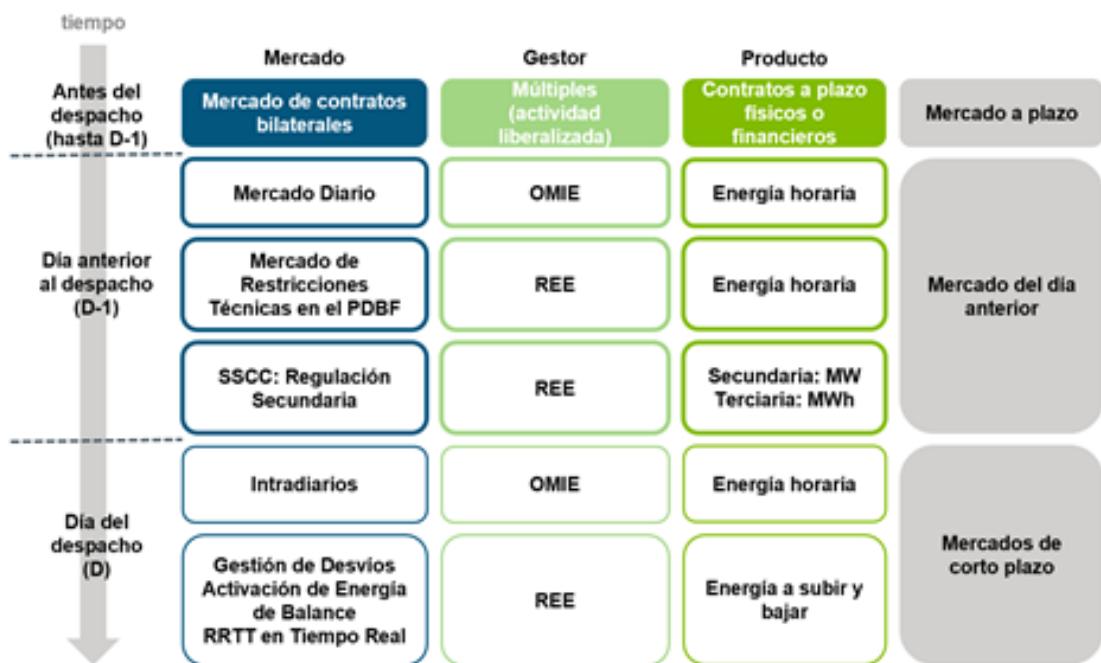


Figura 1.1: Secuencia de mercados en el mercado ibérico de electricidad (MIBEL). Figura de [1].

#### Mercado diario

Como explicamos previamente el objetivo del trabajo es predecir un día 'D' el precio de venta de la energía del día 'D+1', por ello explicaremos como funciona particularmente el mercado diario, gestionado por el Operador del Mercado Ibérico-Polo Español (OMIE), entidad privada encargada de garantizar transparencia, objetividad e independencia en la contratación de energías.

Contrario a lo que se pueda pensar en primera instancia, el mercado diario realmente no consta de un único producto, el precio energético del día siguiente,

### 1.1. Formación de precios en el mercado mayorista diario de electricidad

sino que se negocia el precio energético para cada hora del día siguiente. Por tanto, realmente son 24 productos los que se negocian en este mercado, y la formación del precio de cada uno de ellos se realiza de la misma forma, mediante el corte de las gráficas de oferta y demanda.

El mercado diario en España (y Europa) es marginalista, esto quiere decir que todos los generadores casados (que han sido comprometidos o contratados para suministrar energía) venden la energía al mismo precio, independientemente del precio individual que hayan ofertado. El mercado es así ya que de esta forma se puede garantizar el suministro a mínimo coste. Esto se logra ya que las generadoras reflejan en sus ofertas su coste de oportunidad, permitiendo ordenarlas en función de sus costes ascendenteamente y usar tan solo las más eficientes.

Todo esto se refleja a la hora de realizar la curva de oferta, en la que cada escalón corresponde a ofertas de centrales de la misma tecnología. En un primer instante nos puede sorprender que ciertas tecnologías como la nuclear se ubiquen en la parte baja de la curva. Debemos recordar que los precios ofertados reflejan el coste de oportunidad y no los totales o variables, y es por este motivo por el que las centrales nucleares, a pesar de tener altos costes fijos, se ubiquen en la parte baja de la curva debido a sus costes de oportunidad bajos, mientras que las centrales hidráulicas regulables se ubican en la parte alta ya que su coste de oportunidad es muy alto (tienen la opción de no consumir el agua ahora y hacerlo en un momento económico más ventajoso).

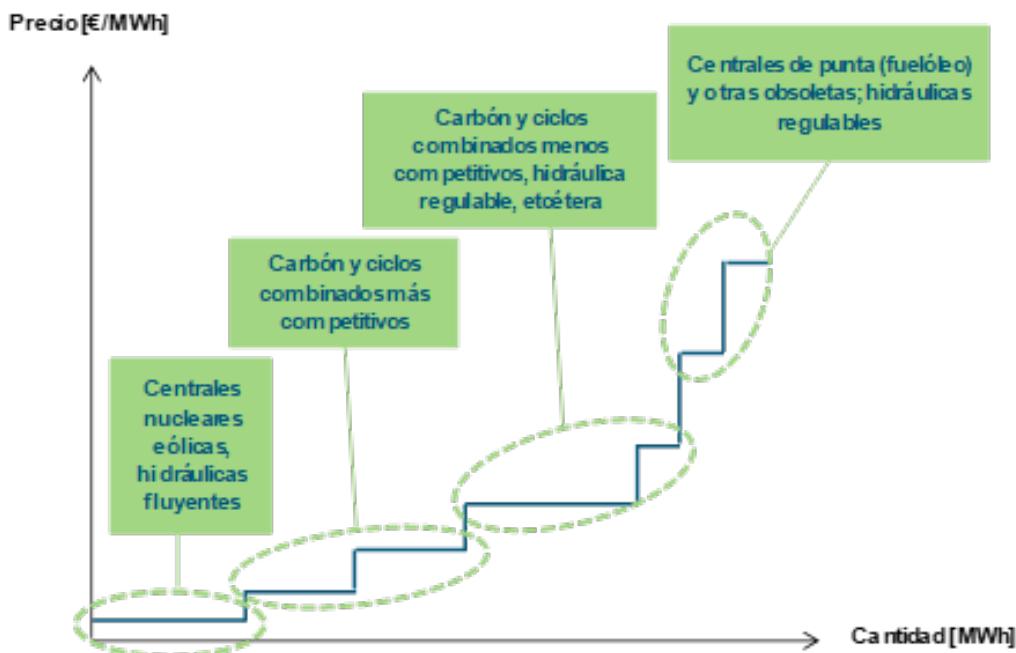


Figura 1.2: Curva de oferta de electricidad del mercado. Figura sacada de [1].

Como hemos explicado previamente, el mercado diario no consta de un único producto, sino de 24, uno para cada hora del día siguiente. A continuación ejemplificaremos como se forma el precio energético para una hora concreta,

## Capítulo 1. Introducción

---

siendo analogo para el resto de horas.

Antes de las 12h tanto vendedores como consumidores (las comercializadoras actúan tanto como vendedoras como compradoras) hacen sus ofertas al mercado. A las 12h el mercado cierra y OMIE se encarga de generar las curvas de oferta y demanda obteniendo los cortes de estas, y a las 14h se publican los precios de casación (SPOT) del día siguiente.

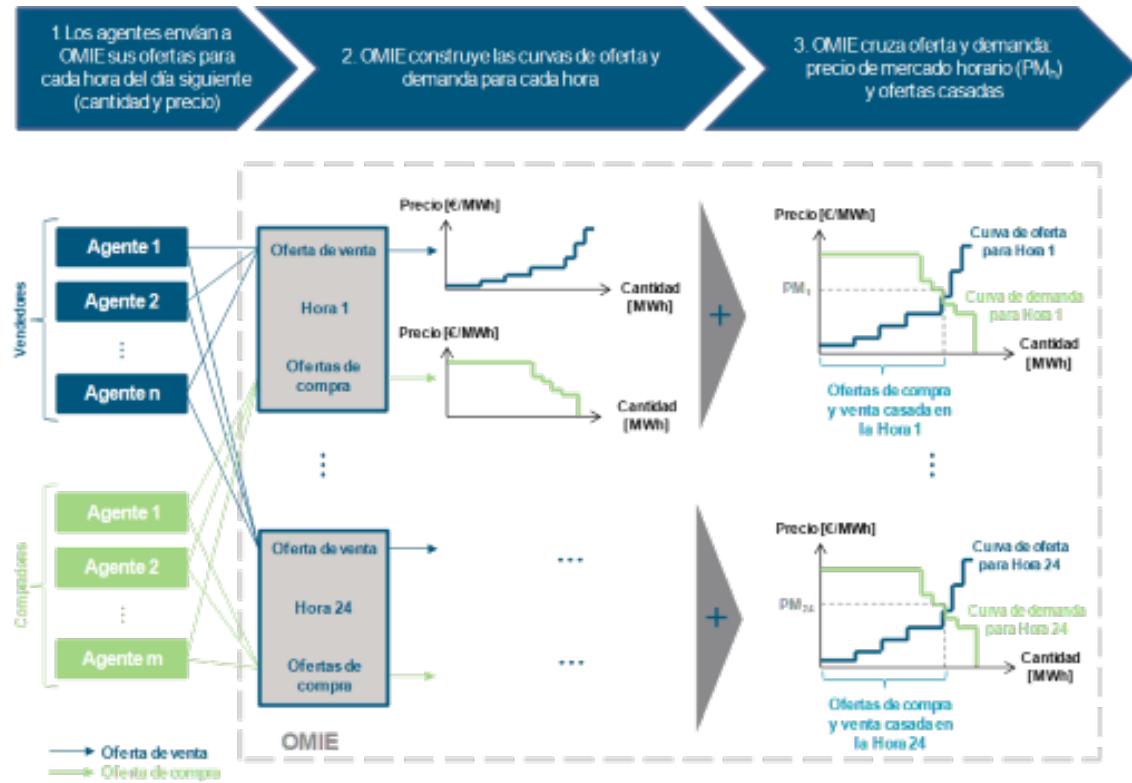


Figura 1.3: Esquema del funcionamiento del mercado diario de OMIE. Figura de [1].

Una vez se ha establecido un precio de casación se determina qué ofertas entran a mercado. Se van seleccionando las ofertas de menor a mayor hasta que la demanda sea cubierta, por lo que las ofertas que sean superiores al precio establecido pueden no llegar a ser casadas, y por tanto, estas no entrarán en mercado y no se venderán. Hay ciertas tecnologías a las que no entrar en mercado no les afecta drásticamente ya que tienen la opción de no generar hoy y hacerlo mañana (centrales de agua embalsada), mientras que hay otras en las que si no se consume la energía generada esta será desperdiciada (renovables).

Es esta última problemática que tienen las generadoras la que motiva la realización de este trabajo. Las generadoras con intención de vender deben ofertar al precio más alto posible para maximizar beneficios pero sin correr el riesgo de vender por encima del precio de casación y no entrar a mercado, e incluso pudiendo perder grandes sumas monetarias por la generación de energía que no será consumida.

Por ello predeciremos con la máxima precisión posible el precio SPOT de 'D+1', consiguiendo de esta forma realizar la mejor oferta posible al mercado.

### 1.2. Tecnologías

Dado que algunas formas de generar energía son más económicas que otras, el precio SPOT se verá influenciado por la fuente de generación de la energía ofertada.

Por ello deberemos comprender de qué tecnologías disponemos a la hora de generar electricidad, y cuando decimos 'generar' realmente nos referimos al proceso de transformar una energía primaria (nuclear, térmica, hidráulica, eólica, solar, etc.) en una eléctrica, ya que, como sabemos, la energía ni se crea ni se destruye, sólo se transforma.

Hay que tener en cuenta también que las transformaciones realizadas degradan la energía, por lo que no seremos capaces de aprovechar esa energía en su totalidad. Además, muchas transformaciones de energía producen emisiones perjudiciales al medio ambiente, por lo que la forma en la que se transforma la energía nos concierne y afecta directamente sobre la formación de precios. Por otra parte debemos comprender que el almacenamiento de grandes cantidades de energía aun no está plenamente desarrollado, por lo que toda energía generada deberá ser consumida. Una buena comprensión de las tecnologías disponibles nos ayudará a entender cómo se forman los precios energéticos que se ofertan.

Antes de explicar las distintas tecnologías existentes creemos importante remarcar que, en general, no hay como tal tecnologías mejores que otras, sino que todas las tecnologías son necesarias, ya que cada tecnología resulta especialmente adecuada técnica y económico para prestar un servicio concreto en relación con la cobertura de la demanda eléctrica y se complementan para suministrar de la forma más adecuada posible (en términos de coste y seguridad de suministro) la energía que demandan los consumidores en cada momento.

A modo de ejemplo, hay tecnologías con costes fijos muy altos (amortización de la inversión, parte fija del coste de operación y mantenimiento, etc.) pero con costes variables muy bajos. Estas tecnologías son las más adecuadas para producir un número de horas al año muy elevado. Por el contrario, hay tecnologías con costes fijos muy bajos pero con costes variables muy altos. Estas tecnologías son las más adecuadas para producir durante un número reducido de horas al año (aquellas en las que la demanda es más alta).

A continuación, presentaremos las distintas tecnologías de generación de energía eléctrica:

**Centrales nucleares:** Esta tecnología se basa en la fisión de los núcleos atómicos pesados, como el uranio-235, que se dividen en fragmentos más pequeños mediante la colisión con neutrones. Esta fisión libera una gran cantidad de energía en forma de calor que se usa para producir vapor de agua a alta presión, que será el encargado de generar electricidad al mover una turbina.

## **Capítulo 1. Introducción**

---

Desde un punto de vista económico, sus costes fijos son muy elevados y sus costes variables relativamente bajos. También se ven beneficiados al no verse afectado por los impuestos añadidos por emisiones de CO<sub>2</sub>. Además, en el contexto de la crisis energética desencadenada en 2021 por la subida de los precios del gas natural en el mercado europeo, el Parlamento comenzó la tramitación de un proyecto de Ley sobre la retribución a las centrales por el CO<sub>2</sub> no emitido.

**Centrales térmicas convencionales (carbón, gas natural y fuelóleo):** Al igual que en las centrales nucleares, la energía es producida por una turbina movida por vapor. Sin embargo, en estas centrales el calor generado para producir vapor se logra mediante la combustión de combustible fósiles. Estas centrales tienen un gran impacto ambiental debido a la emisión de gases de efecto invernadero.

**Centrales térmicas de ciclo combinado:** Estas centrales térmicas funcionan bajo el mismo principio que las centrales térmicas convencionales, sin embargo estas tienen una mayor eficiencia debido a que la energía térmica es transformada en electricidad mediante dos ciclos: primero una turbina de gas y después una turbina de vapor.

**Cogeneración:** La cogeneración es un proceso en el cual se produce simultáneamente energía eléctrica y térmica a partir de una misma fuente de energía. Generalmente, se aprovecha el calor residual generado durante la producción de electricidad para usos térmicos como calefacción o procesos industriales. Esta tecnología tiene un alto rendimiento (en torno al 80-90 %).

**Centrales hidráulicas:** Estas centrales aprovechan la energía cinética o potencial del agua situada en el cauce de un río o retenida en un embalse, convirtiéndola en energía eléctrica a través de un generador acoplado a una turbina. Existen 3 tipos principales: agua embalsada, hidráulicas fluyentes y de bombeo o reversibles.

**Generación eólica:** Mediante esta tecnología la electricidad es generada a partir de la energía cinética del viento.

**Generación solar:** La energía solar se puede aprovechar mediante:

- Tecnología solar fotovoltaica: Se usan células solares que generan electricidad cuando incide sobre ellos la radiación solar.
- Tecnología solar termoeléctrica: La energía eléctrica se obtiene mediante el calentamiento de un fluido.

**Biomasa:** Funcionan igual que las centrales térmicas convencionales, con la diferencia de que el combustible utilizado es de origen orgánico.

**Energías del mar:** Podemos aprovechar el mar para generar energía de diversas maneras:

- Energía mareomotriz: Se basa en el ascenso y descenso del agua del mar producido por la acción gravitatoria del Sol y la Luna.
- Energía de las corrientes: Se basa en la energía cinética de las corrientes de mar

### **1.3. Características económicas de las tecnologías de generación**

---

- Energía maremotérmica: Se basa en la diferencia de temperaturas entre la superficie y aguas profundas
- Energía de las olas: Se basa en el movimiento ondulatorio.
- Energía azul o potencia osmótica: Se basa en la diferencia en la concentración de sal entre el agua de mar y el agua dulce de los ríos mediante los procesos de ósmosis.

**Geotermia:** Esta energía renovable proviene del calor almacenado bajo la superficie terrestre.

### **1.3. Características económicas de las tecnologías de generación**

Como explicamos previamente, no es posible almacenar grandes cantidades de electricidad para su posterior consumo y por ello, se debe garantizar un abastecimiento eléctrico que satisfaga la cambiante demanda en cada instante sin desperdiciar energía. En esta sección entenderemos cómo se logra abordar este reto de la forma más eficiente, de modo que nunca sobre ni falte energía.

Las tecnologías descritas en la sección anterior tienen estructuras de costes y una capacidad para adaptarse a las variaciones de la demanda muy diferentes. Por ejemplo, si en un momento se produce un pico en la demanda una central nuclear no será capaz de variar su producción rápidamente ya que esta requiere una planificación meticulosa. Sin embargo, un embalse tan solo deberá dejar correr el agua necesaria para suplir esa demanda extra, y cuando esta sea satisfecha tan solo deberá cerrar de nuevo el fluente de agua.

En cuanto a la estructura de costes, las tecnologías se pueden clasificar de la siguiente manera:

- Tecnologías de **base** (nucleares, algunas de las centrales de carbón y ciclos combinados existentes), con costes fijos relativamente elevados y variables relativamente bajos.
- Tecnologías de **punta** (centrales de fuelóleo, turbinas de gas), con costes fijos bajos y variables altos.
- Tecnologías **intermedias** (algunas centrales de carbón y ciclos combinados existentes, centrales hidráulicas regulables), con costes fijos y variables intermedios respecto a los de las centrales de base y punta.

Debido a estas diferencias entre tecnologías es eficiente que la energía no provenga de una única tecnología, sino de la combinación de varias fuentes con diferentes capacidades. La combinación de fuentes concreta influirá en la formación del precio SPOT.

## Capítulo 1. Introducción

---

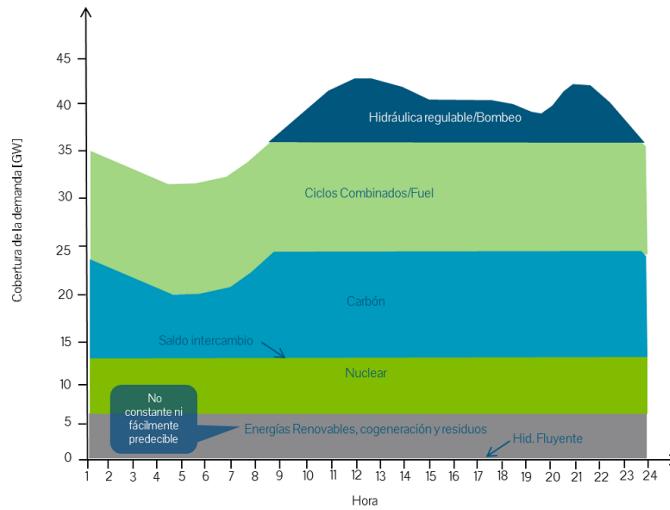


Figura 1.4: Cobertura ilustrativa de la curva de demanda horaria en un cierto día por las distintas tecnologías. Figura sacada de [2].

## 1.4. Objetivos

El objetivo principal de este trabajo es predecir con la máxima precisión el precio SPOT del MWh (precio casado) para cada una de las horas del día próximo con el fin de optimizar la oferta en el mercado.

Para ello se estudiaran diferentes modelos, tanto clásicos como más complejos y novedosos de Deep Learning y del estado del arte.

También evaluamos en este trabajo si los últimos modelos más complejos de Deep Learning (DL) otorgan notables mejoras frente a modelos clásicos, i.e., si la complejidad de estos modelos justifica su uso en este trabajo.

## 1.5. Metodología

En esta sección explicaremos como se va a lograr desarrollar el trabajo propuesto.

### 1.5.1. Obtención de datos

En primer lugar necesitamos realizar un estudio sobre qué factores influyen en la formación del precio SPOT, así como de qué fuentes existen para obtener los datos necesarios.

Una vez sepamos qué datos usar y cómo obtenerlos procederemos a elaborar un conjunto de datos (dataset) que usaremos para entrenar y validar los modelos.

### **1.5.2. Modelado**

Se realiza un estudio sobre los modelos matemáticos más adecuados y el estado del arte en predicción de series temporales. De entre los modelos estudiados, se seleccionan para su posterior implementación los más adecuados.

Para llevar a cabo una selección precisa del modelo óptimo, para cada modelo seleccionado realizamos una comparativa exhaustiva entre diferentes arquitecturas del mismo, ajustando sus hiperparámetros y evaluándolos con el fin de obtener la mejor configuración.

Basándonos en los resultados obtenidos en la comparativa previa, seleccionamos la mejor configuración del modelo usando datos de validación.

### **1.5.3. Comparativa de modelos**

Una vez halladas las configuraciones óptimas de cada modelo se procede a realizar una comparativa entre ellos con el objetivo de seleccionar un modelo ganador usando los datos de test.

Los datos de test corresponderán a las últimas 2 semanas disponibles del mes de abril de 2024. Dado que en nuestro trabajo predecimos con un horizonte temporal diario, las predicciones para estas 2 semanas se realizarán de manera secuencial. Esto significa que las predicciones diarias se llevarán a cabo una por una, día tras día, y posteriormente compararemos las predicciones acumuladas de estas 2 semanas con los valores reales.

### **1.5.4. Análisis de resultados**

Finalmente se evaluará el rendimiento del modelo ganador con el fin de determinar si se consiguen realizar unas buenas predicciones y valorar si el uso del modelo podría ayudar a realizar ofertas óptimas en el mercado.

Puesto que los TFT son modelos explicables procederemos a interpretarlo.

### **1.5.5. Restricciones técnicas**

- Puesto que el mercado cierra a las 12h y las predicciones deben realizarse antes, la obtención de datos debe realizarse lo más tarde posible para contar con los últimos datos disponibles, pero no demasiado tarde como para correr el riesgo de que no de tiempo a predecir y evaluar las predicciones. Por ello, la obtención de datos, así como la predicción deberá ser rápida para poder retrasar lo máximo posible la obtención de estos y contar con valores reales en vez de previsiones.
- El precio SPOT del día 'D+1' se publica a las 15h del día 'D'
- Las ofertas energéticas para el día 'D+1' se pueden realizar antes de las 12h del día 'D'.

## **Capítulo 1. Introducción**

---

- Existen dos fechas al año en las que se produce un cambio horario que afecta al número de horas de luz diarias, y por tanto, a la demanda energética. Por ello el modelo puede verse afectado por estos cambios, siendo el más reciente el cambio horario del 30 de marzo de 2024.

## **Capítulo 2**

# **Fundamentos Teóricos**

Una vez hemos obtenido una clara compresión sobre el sector y dominio en el que nos encontramos así como del problema al que se enfrentan las empresas, deberemos caracterizarlo en el dominio de Ciencia de Datos e Inteligencia Artificial.

En este capítulo además de caracterizar el problema en el dominio que nos concierne daremos una explicación teórica sobre las distintas técnicas y modelos y que usaremos en la realización de este trabajo.

### **2.1. Caracterización del dominio**

El objetivo principal de este trabajo es, dada una sucesión de valores ordenados en el tiempo, predecir los siguientes 24 valores en base a los valores anteriores, además de otras variables. El trabajo entra en el dominio de predicción de series temporales.

Debido a la naturaleza del dominio estudiaremos los siguientes modelos por su idoneidad para este:

- (S)ARIMA(X).
- Long Short-Term Memory (LSTM [3]).
- Convolutional Neural Networks (CNN) - 1D ([4]).
- Temporal Fusion Transformers (TFT [5])

En las siguientes secciones introduciremos el funcionamiento de cada uno de ellos.

### **2.2. Serie Temporal**

Una serie temporal [6] se compone de observaciones cuantitativas sobre una o más características medibles de una entidad individual y tomadas en múltiples puntos en el tiempo.

## Capítulo 2. Fundamentos Teóricos

El análisis de series temporales permite investigar patrones y tendencias en los datos a lo largo del tiempo. Esto incluye examinar la correlación temporal entre los valores pasados y presentes, evaluar cómo se influyen mutuamente y su relación con otras variables exógenas<sup>1</sup>. Además, se busca identificar la presencia de tendencias a largo plazo, patrones repetitivos como estacionalidades, así como también la variabilidad inherente en los datos.

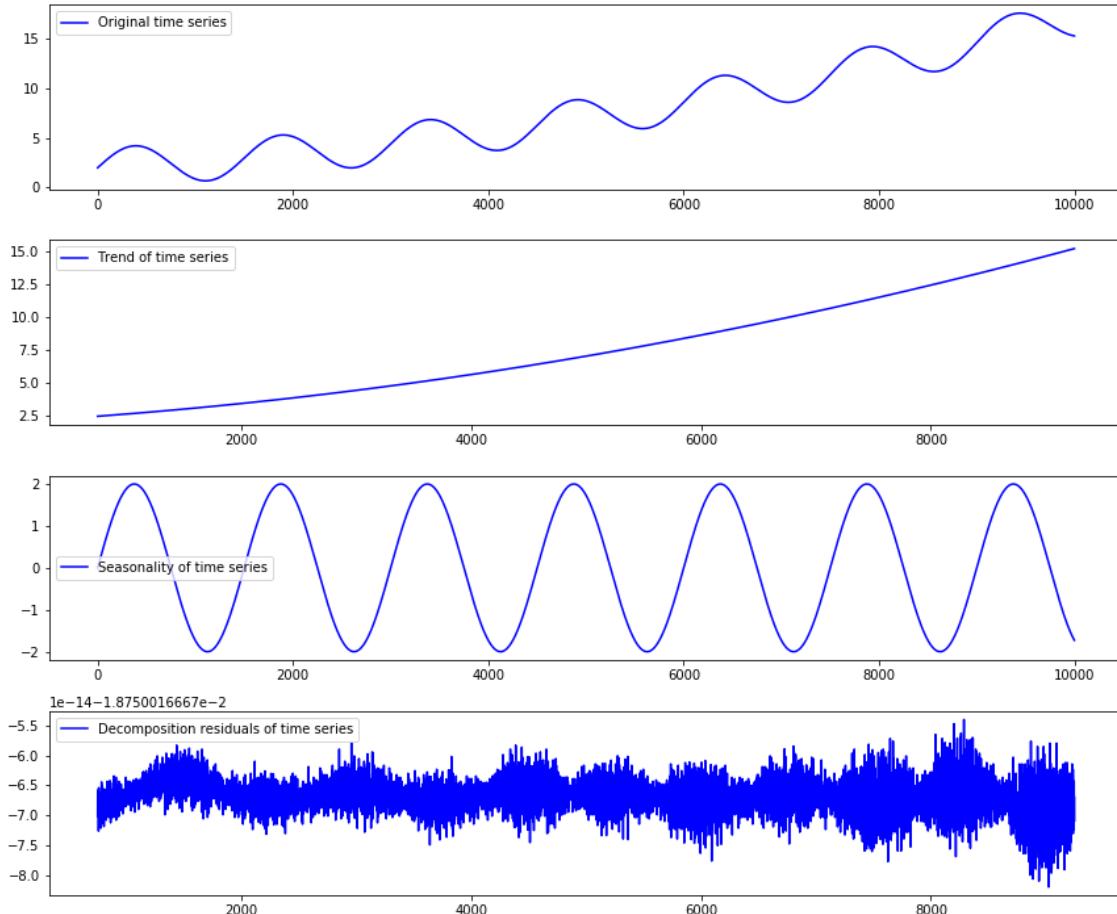


Figura 2.1: Descomposición de series temporales en componentes de tendencia y estacionalidad. La figura superior presenta la serie temporal original. La segunda figura desde la parte superior es el componente de tendencia estimado. La tercera figura desde la parte superior es la componente de estacionalidad estimada (periódica). La figura inferior es el residuo de la descomposición. Figura obtenida de [7].

<sup>1</sup>En nuestro contexto, las variables exógenas son factores externos al sistema de predicción que pueden influir en el precio de la energía. Ejemplos de estas variables incluyen condiciones meteorológicas, precios de combustibles, demanda energética, políticas gubernamentales entre otras.

## 2.3. Modelos Clásicos

### 2.3.1. ARIMA

Los modelos ARIMA (*Autoregressive Integrated Moving Average*) son herramientas fundamentales en el análisis y pronóstico de series temporales. Estos modelos combinan tres componentes principales:

- **Autorregresión (AR):** implica que los valores de la serie en un momento dado se relacionan linealmente con los valores previos de la misma serie. Matemáticamente, un modelo AR(p) se define como:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \dots + \phi_p Y_{t-p} + \varepsilon_t$$

donde  $Y_t$  es el valor de la serie en el tiempo  $t$ ,  $c$  es una constante,  $\phi_1, \phi_2, \dots, \phi_p$  son los coeficientes de autorregresión,  $\varepsilon_t$  es el término de error en el tiempo  $t$ .

- **Integración (I):** se refiere a la diferenciación de la serie para hacerla estacionaria, es decir, eliminar tendencias y estacionalidad, lo que simplifica el modelado. Matemáticamente, se aplica una diferencia  $d$  veces a la serie original, representada como  $\nabla^d Y_t$ .
- **Media móvil (MA):** considera la relación entre los errores de predicción sucesivos. Matemáticamente, un modelo MA(q) se define como:

$$Y_t = \mu + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

donde  $\mu$  es la media de la serie,  $\theta_1, \theta_2, \dots, \theta_q$  son los coeficientes de la media móvil y  $\varepsilon_t$  es el término de error en el tiempo  $t$ .

Los modelos ARIMA se representan como ARIMA(p, d, q), donde  $p$  denota el orden de la autorregresión,  $d$  la cantidad de veces que se ha diferenciado la serie para hacerla estacionaria y  $q$  el orden del promedio móvil. La elección adecuada de estos parámetros es crucial para un buen rendimiento del modelo. Para ello, se pueden utilizar técnicas como la inspección de la función de autocorrelación (ACF) y la función de autocorrelación parcial (PACF) para identificar los valores óptimos de  $p$  y  $q$ , mientras que  $d$  se determina mediante pruebas de estacionariedad.

Una vez identificados los parámetros del modelo ARIMA, se procede a la estimación de los coeficientes mediante métodos como el método de máxima verosimilitud. Posteriormente, el modelo se puede utilizar para realizar pronósticos de la serie temporal, permitiendo prever su comportamiento futuro con base en los datos históricos.

### 2.3.2. SARIMA

Similar a los modelos ARIMA, los modelos SARIMA combinan autorregresión, integración y media móvil, pero también incorporan componentes estacionales.

## Capítulo 2. Fundamentos Teóricos

---

Estos modelos son especialmente útiles cuando los datos exhiben patrones repetitivos a lo largo del tiempo, como estacionalidad anual en datos climáticos o diaria como en nuestro caso específico.

Los modelos SARIMA se representan como SARIMA(p, d, q)(P, D, Q)s, donde los parámetros  $p$ ,  $d$ ,  $q$  son análogos a los de ARIMA para los componentes no estacionales, mientras que  $P$ ,  $D$ ,  $Q$  representan los órdenes de la autorregresión estacional, la diferenciación estacional y la media móvil estacional, respectivamente. El parámetro 's' indica la periodicidad de la estacionalidad, es decir, el número de períodos en un ciclo estacional.

### 2.3.3. SARIMAX

Los modelos SARIMAX (*Seasonal Autoregressive Integrated Moving Average with Exogenous Regressors*) son una extensión de los modelos SARIMA que permiten incorporar variables exógenas en la modelización de series temporales. Estas variables exógenas pueden ser cualquier conjunto de datos que no formen parte de la serie temporal principal pero que se cree que tienen influencia en su comportamiento. Al incluir estas variables en el modelo, SARIMAX puede capturar mejor la complejidad de la serie temporal y mejorar la precisión de los pronósticos.

La representación matemática de los modelos SARIMAX es similar a la de los modelos SARIMA, pero con la inclusión de términos que capturan el efecto de las variables exógenas. Por ejemplo, un modelo SARIMAX(p, d, q)(P, D, Q)s con  $k$  variables exógenas puede expresarse como:

$$Y_t = c + \phi_1 Y_{t-1} + \dots + \phi_p Y_{t-p} + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q} + \sum_{i=1}^k \beta_i X_{it} + \varepsilon_t$$

donde  $X_{it}$  representa la  $i$ -ésima variable exógena en el tiempo  $t$ , y  $\beta_i$  son los coeficientes asociados con estas variables. .

## 2.4. Modelos Neuronales

### Introducción a las Redes Neuronales

Las redes neuronales son modelos computacionales inspirados en el funcionamiento del cerebro humano. Estas redes están compuestas por unidades básicas, llamadas neuronas, que están interconectadas en capas. Cada neurona procesa información y transmite señales a través de conexiones ponderadas hacia las neuronas de la capa siguiente. Estas tienen la capacidad de aprender y adaptarse a partir de datos les permite realizar tareas como clasificación, regresión, reconocimiento de patrones y más.

Un ejemplo fundamental de red neuronal es el perceptrón [8]. El perceptrón es un modelo de clasificación binaria que puede aprender a separar dos conjuntos de datos utilizando una función de activación. Visualmente, podemos imaginar

al perceptrón como una línea recta que divide el espacio de entrada en dos regiones: una región donde los puntos pertenecen a una clase y otra región donde pertenecen a la otra clase. En otras palabras, el perceptrón puede aprender a realizar una clasificación lineal.

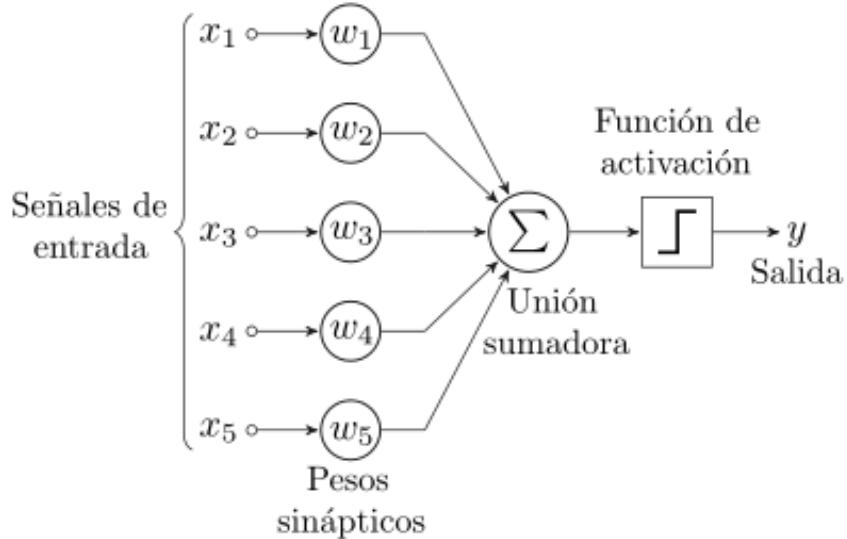


Figura 2.2: Esquema del funcionamiento de un perceptrón. Imagen obtenida de [9].

La formulación matemática del perceptrón es simple. Dado un conjunto de datos de entrada  $X$  y un vector de pesos  $W$  y un sesgo  $b$ , el perceptrón calcula la salida  $y$  como:

$$y = \begin{cases} 1 & \text{si } W \cdot X + b > 0 \\ 0 & \text{en otro caso} \end{cases}$$

Donde  $W \cdot X$  representa el producto entre el vector de pesos  $W$  y el vector de entrada  $X$ , y  $b$  es el sesgo.

Aunque el perceptrón es un modelo simple, sentó las bases para redes neuronales más complejas y poderosas. Con el desarrollo de algoritmos de aprendizaje profundo y la disponibilidad de conjuntos de datos másivos, las redes neuronales han demostrado ser capaces de resolver problemas altamente complejos en una variedad de campos.

#### 2.4.1. Long Short-Term Memory (LSTM)

Las redes neuronales LSTM (Long Short-Term Memory) son un tipo especializado de redes neuronales recurrentes (RNN [10]) que han demostrado ser altamente efectivas en el modelado de secuencias de datos, como las series temporales. A diferencia de las RNN tradicionales, las LSTM están diseñadas para manejar

## Capítulo 2. Fundamentos Teóricos

---

problemás de largo alcance de dependencia temporal al mantener y actualizar un estado de memoria interno a lo largo del tiempo.

Matemáticamente, una capa LSTM consiste en una serie de puertas que controlan el flujo de información en la red. La formulación matemática de una unidad LSTM se compone de varias ecuaciones que describen cómo se actualiza y se modula el estado de la celda y la salida de la unidad en cada paso de tiempo. La ecuación básica de una unidad LSTM se puede expresar como sigue:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\ \hat{C}_t &= \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \hat{C}_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned}$$

donde:

- $x_t$  es la entrada en el tiempo  $t$ ,
- $h_t$  es la salida en el tiempo  $t$ ,
- $c_t$  es el estado de la celda en el tiempo  $t$ ,
- $i_t, f_t, \hat{C}_t, o_t$  son las puertas de entrada, olvido, actualización de celda y salida respectivamente,
- $\sigma$  es la función de activación sigmoide,
- $\odot$  representa la multiplicación elemento por elemento, y
- $W$  y  $b$  son los pesos y sesgos de la red respectivamente.

En la figura 2.3 se puede ver una representación visual de la formulación matemática recién descrita.

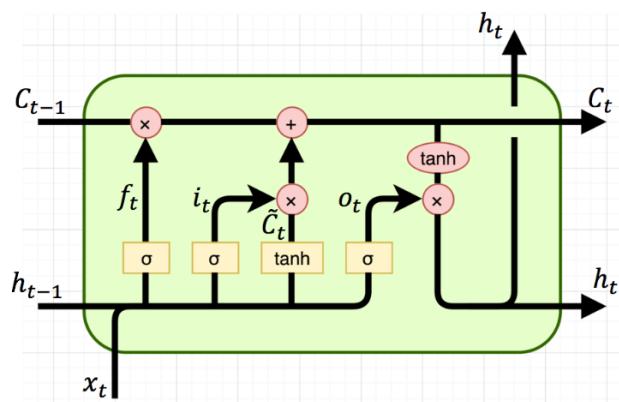


Figura 2.3: Diagrama de una unidad LSTM. Figura sacada de [11].

Las LSTM son capaces de aprender dependencias a largo plazo en los datos, lo que las hace adecuadas para una amplia gama de aplicaciones de predicción y modelado de secuencias, como el análisis de series temporales, el procesamiento de lenguaje natural y la generación de texto, entre otros.

### 2.4.2. Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales (CNN) [12] son un tipo de arquitectura de red neuronal profunda que ha demostrado ser altamente efectiva en tareas de visión por computadora, como la clasificación de imágenes y la detección de objetos. Las CNN están diseñadas para extraer automáticamente características relevantes de los datos de entrada, lo que las hace especialmente útiles para procesar datos de imágenes.

Matemáticamente, una CNN está compuesta por capas convolucionales, capas de agrupamiento (pooling) y capas completamente conectadas.

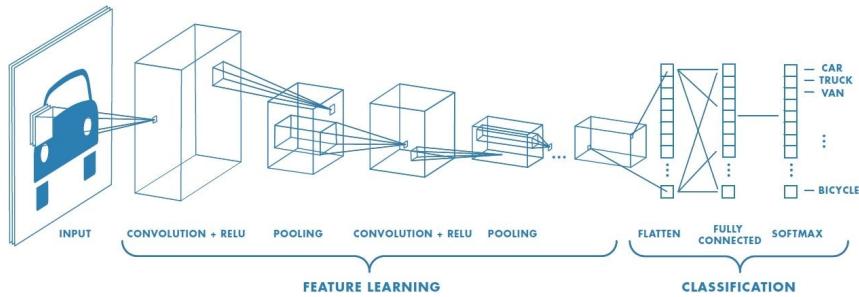


Figura 2.4: Esquema de la arquitectura de CNN. Figura sacada de [13].

La capa convolucional es el componente central de una CNN y consiste en aplicar filtros convolucionales a la entrada para extraer características relevantes. Cada filtro convolucional se desliza a través de la entrada y realiza operaciones de convolución para producir mapas de características. Estos mapas de características capturan diferentes aspectos de la entrada, como bordes, texturas y patrones locales.

La formulación matemática de una operación de convolución en una CNN se expresa como sigue:

$$\text{Salida}(i, j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \text{Entrada}(i+m, j+n) \times \text{Filtro}(m, n)$$

donde:

- Entrada( $i, j$ ) es el valor de píxel en la entrada en la posición ( $i, j$ ),
- Filtro( $m, n$ ) es el valor del filtro en la posición ( $m, n$ ),
- Salida( $i, j$ ) es el valor de píxel en la salida en la posición ( $i, j$ ),
- $M$  y  $N$  son las dimensiones del filtro.

## Capítulo 2. Fundamentos Teóricos

Las CNN son capaces de aprender automáticamente las características importantes de los datos de entrada, lo que las hace muy versátiles y efectivas en una amplia gama de aplicaciones de visión por computadora.

### Redes Neuronales Convolucionales 1D (CNN 1D) para Series Temporales

Las Redes Neuronales Convolucionales unidimensionales (CNN 1D) son una variante de las CNN diseñadas específicamente para modelar datos unidimensionales, como las series temporales. Aunque las CNN 1D son comúnmente asociadas con tareas de procesamiento de señales, como el procesamiento de audio y la clasificación de texto, también son altamente efectivas en el análisis de series temporales debido a su capacidad para capturar patrones locales en los datos.

Una de las principales razones para utilizar CNN 1D en el análisis de series temporales es su capacidad para aprender automáticamente características relevantes de los datos en diferentes escalas temporales. Las capas convolucionales en una CNN 1D aplican filtros a lo largo de la dimensión temporal de la serie, lo que les permite capturar patrones locales en diferentes ventanas de tiempo. Esto es especialmente útil en el análisis de series temporales, donde los patrones de interés pueden variar en escala temporal.

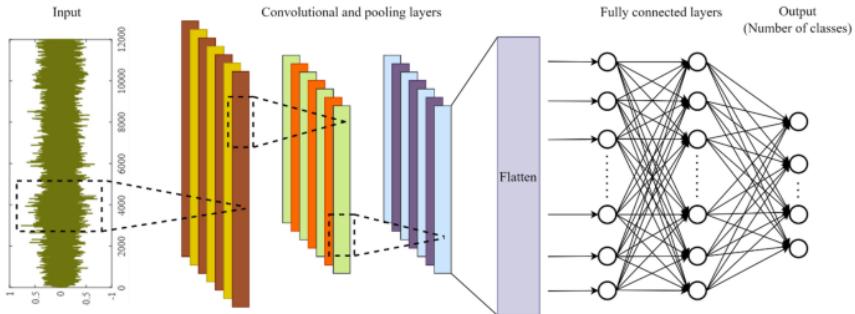


Figura 2.5: Esquema de arquitectura CNN 1-D. Figura sacada de [4].

La formulación matemática de una operación de convolución en una CNN 1D es similar a la de una CNN convencional, pero aplicada a una dimensión unidimensional de los datos. Esto permite que la red aprenda automáticamente las características relevantes de la serie temporal, como tendencias, patrones estacionales y cambios abruptos, sin requerir una ingeniería manual de características.

En resumen, las CNN 1D son una opción poderosa y eficaz para el análisis de series temporales debido a su capacidad para capturar patrones locales en diferentes escalas temporales de manera automática.

## 2.5. Transformers

Los Transformers [14] son una arquitectura que, a diferencia de las tradicionales como las redes neuronales recurrentes (RNN [10]) y las redes neuronales

les convolucionales (CNN) que dependen en gran medida de la recurrencia y la convolución para capturar las dependencias temporales en los datos, los Transformers utilizan mecanismos de atención para aprender las relaciones entre las diferentes posiciones en una secuencia de datos.

La particularidad de los Transformers es su mecanismo de atención, que permite que cada elemento de la secuencia se relacione con todos los demás elementos, capturando así las dependencias a largo y corto plazo de manera más efectiva. Esto les otorga una ventaja significativa en el modelado de series temporales, ya que son capaces de capturar relaciones complejas y dependencias temporales no lineales en los datos.

La arquitectura básica de un Transformer consta de bloques de atención, capas de normalización y capas de alimentación hacia adelante.

Cada bloque de atención calcula una matriz de atención que pondera la importancia de cada elemento de la secuencia con respecto a los demás, mientras que las capas de normalización y las capas de alimentación hacia adelante ayudan a estabilizar y transformar la salida de cada bloque de atención.

La formulación matemática del mecanismo de atención en un Transformer se puede expresar como sigue:

Dado un conjunto de vectores de entrada  $X = [x_1, x_2, \dots, x_n]$ , donde  $x_i$  representa el vector de características del elemento en la posición  $i$ , la salida del mecanismo de atención se calcula como:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

Donde:

- $Q$ ,  $K$  y  $V$  son las matrices de consultas, claves y valores respectivamente, que se calculan proyectando los vectores de entrada  $X$  a través de capas lineales.
- $d_k$  es la dimensión de las claves y se utiliza para escalar el producto entre las consultas y las claves.
- softmax es la función de activación softmax que normaliza los pesos de atención para que sumen 1.

Una vez calculada la matriz de atención, se utiliza para ponderar los vectores de valores  $V$ , lo que produce la salida final del mecanismo de atención.

Además, los Transformers se componen típicamente de un conjunto de bloques de codificador y un conjunto de bloques de decodificador.

**Encoder (codificador):** El bloque de codificador en un Transformer se encarga de procesar la entrada y extraer representaciones de características relevantes. Cada bloque de codificador contiene capas de atención y capas de alimentación hacia adelante, que permiten al modelo aprender representaciones contextualizadas de la entrada.

## Capítulo 2. Fundamentos Teóricos

**Decoder (decodificador):** El bloque de decodificador en un Transformer se utiliza en tareas de generación de secuencias, como la traducción automática. Recibe la salida del codificador y utiliza una estructura similar al codificador, pero con la adición de una capa de atención adicional que permite al modelo centrarse en diferentes partes de la entrada y la salida durante la generación de la secuencia de salida.

La arquitectura Transformer también incluye conexiones residuales y capas de normalización después de cada bloque de atención, lo que ayuda a mitigar el problema de la desaparición del gradiente y acelerar el entrenamiento del modelo.

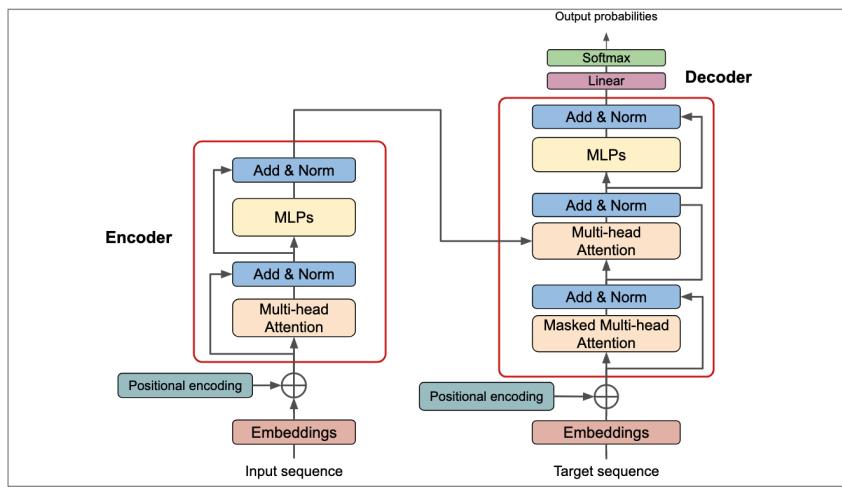


Figura 2.6: Esquema de arquitectura Transformer. MLPs hace referencia a Multilayer Perceptron. Figura sacada de [15].

### 2.5.1. Temporal Fusion Transformers (TFT)

Temporal Fusion Transformers (TFT) [5] es una arquitectura de red neuronal del estado del arte desarrollada específicamente para abordar problemas de series temporales con múltiples variables de entrada. Combina la flexibilidad de las Redes Neuronales Recurrentes (RNN) [10] con la capacidad de modelado de atención de los Transformers [14], lo que lo convierte en una opción poderosa para el análisis y la predicción de series temporales multivariable, con la ventaja además de ser un modelo explicable.

En problemas reales de forecasting, nos encontramos con datos que son conocidos en diferentes intervalos de tiempo. Algunos datos futuros son desconocidos, como la variable objetivo que intentamos predecir, mientras que otros pueden ser conocidos, como si un día será festivo o no.

Sin embargo, esta aproximación puede ser limitada, ya que no aprovecha la complejidad y riqueza de las distintas formas de datos disponibles en problemas reales.

Hasta ahora, los modelos existentes tienden a manejar los datos de forma me-

nos flexibles, centrándose solo en datos históricos (conocidos) o en futuros (desconocidos). En contraste, los TFT permiten la inclusión de múltiples tipos de variables, tanto conocidas como desconocidas, a lo largo de diferentes horizontes temporales. Esto ofrece una mayor flexibilidad y precisión en la modelización de problemas reales.

En lugar de simplemente dividir las variables en conocidas y desconocidas, los TFT clasifican las variables en categorías más específicas: variables pasadas estáticas, variables pasadas conocidas que varían en el tiempo, variables futuras desconocidas que varían en el tiempo, así como variables futuras conocidas que varían en el tiempo (Fig. 2.7 y Fig. 2.9). Esta clasificación permite capturar de manera más efectiva las relaciones y patrones subyacentes en los datos, mejorando tanto la precisión de las predicciones como la interpretabilidad del modelo.

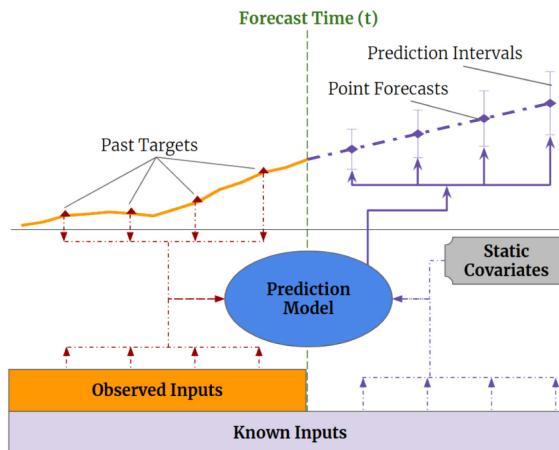


Figura 2.7: Ilustración de la predicción con múltiples horizontes utilizando variables pasadas estáticas, variables pasadas conocidas y desconocidas que varían en el tiempo. Figura obtenida de [5].

Procedemos a detallar las diferencias entre estas variables, dividiendo a su vez en variables reales y categóricas:

- **Static reals:** Son variables numéricas que no cambian a lo largo del tiempo. Ejemplos incluyen la altura de un edificio o la capacidad de un tanque de almacenamiento.
- **Static categoricals:** Son variables categóricas que permanecen constantes con el tiempo. Un ejemplo podría ser el tipo de industria de una empresa o el país de origen de un producto.
- **Time varying known reals:** Son variables numéricas que varían a lo largo del tiempo y cuyos valores son conocidos en todos los puntos temporales. Por ejemplo, los horarios programados de vuelos o los precios regulados de la electricidad en diferentes horas del día.
- **Time varying known categoricals:** Son variables categóricas que cambian con el tiempo y cuyos valores también son conocidos. Un ejemplo sería

## Capítulo 2. Fundamentos Teóricos

---

el día de la semana (lunes, martes), días festivos o el turno de trabajo (mañana, tarde, noche).

- **Time varying unknown real:** Son variables numéricas que varían con el tiempo pero cuyos valores no son conocidos de antemano y deben ser predichos. En nuestro caso será el precio SPOT.
- **Time varying unknown categoricals:** Son variables categóricas que cambian con el tiempo y cuyos valores no son conocidos previamente. Un ejemplo sería el tipo de incidente que podría ocurrir en una central eléctrica (fallo mecánico, fallo eléctrico).

La arquitectura completa de un TFT (fig. 2.9) es bastante compleja, por lo que nos limitaremos a describir y conceptualizar los principales componentes de los TFT:

1. **Gating mechanisms** (Mecanismos de control) para saltarse cualquier componente no necesario de la arquitectura, proporcionando complejidad y profundidad adaptable. Es decir, es el componente que permite al modelo ajustarse tanto a problemas sencillos como a problemas complejos ajustando autónomamente la propia profundidad del modelo.
2. **Variable selection networks** (Redes de selección de variables) para seleccionar variables de entrada relevantes en cada paso de tiempo.
3. **Static covariate encoders** (Codificadores de covariables estáticas) para integrar características estáticas en la red, a través de la codificación de vectores de contexto para condicionar la dinámica temporal. Estos serán los encargados de introducir en el modelo nuestras variables estáticas.
4. **Temporal processing** (Procesamiento temporal) para aprender relaciones temporales a largo y corto plazo tanto de entradas observadas, como conocidas que varían en el tiempo. Se emplea una capa de secuencia a secuencia para el procesamiento local, mientras que las dependencias a largo plazo se capturan utilizando un novedoso bloque de multi-head attention interpretable.
5. **Prediction intervals** (Intervalos de predicción) mediante pronósticos de cuantiles para determinar el rango de valores objetivo probable en cada horizonte de predicción.

Explicamos más a fondo los componentes recién mencionados y el por qué de su implementación:

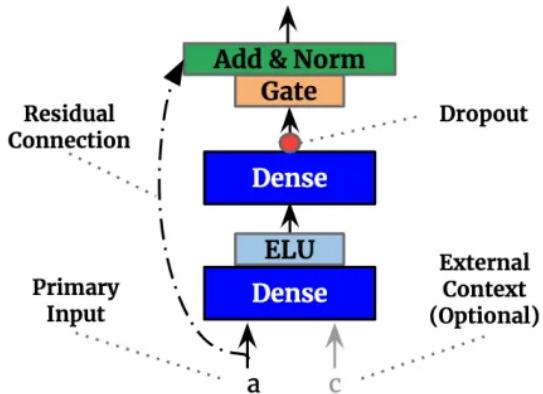
### **Gating mechanisms**

---

Si nuestro problema es uno sencillo pero tratamos de modelizarlo con una arquitectura muy compleja y profunda estaremos forzando al modelo a aprender relaciones inexistentes o a modelizar el ruido en los datos.

En general es complicado determinar el grado de procesamiento no lineal necesario para lograr modelizar correctamente nuestro problema. Por ello, los TFTs,

con el objetivo de proporcionar al modelo la flexibilidad de aplicar procesamiento no lineal solo cuando sea necesario, se implementan las Redes Residuales con Puerta (Gated Residual Network, GRN) como un bloque en la arquitectura TFT.



**Gated Residual Network (GRN)**

Figura 2.8: Gated Residual Network (GRN). Imagen de [5]

### Variable selection networks

---

A la hora de entrenar nuestros modelos disponemos de múltiples variables. Sin embargo, no todas ellas tienen por qué ser relevantes y su contribución específica a la salida suele ser desconocida.

TFT está diseñado para realizar una selección de variables a nivel de instancia (para cada predicción) a través del uso de redes de selección de variables.

Este componente tiene 2 ventajas:

- Eliminar cualquier entrada ruidosa innecesaria que podría afectar negativamente el rendimiento.
- Interpretabilidad: Proporcionar información sobre qué variables son más significativas para realizar predicciones.

La mayoría de los conjuntos de datos de series temporales del mundo real contienen características con menos poder predictivo, por lo que la selección de variables puede ayudar enormemente al rendimiento del modelo mediante la utilización de tan solo en las variables más relevantes.

### Static covariate encoders

---

A diferencia de otras arquitecturas de predicción de series temporales, el TFT está diseñado para integrar la información de nuestras variables estáticas.

Estos encoders están diseñados para transformar estas variables estáticas en representaciones contextuales que el modelo puede utilizar a lo largo del proceso

## Capítulo 2. Fundamentos Teóricos

---

de predicción. Al incorporar esta información, los static covariate encoders mejoran la capacidad del TFT para hacer predicciones más precisas y relevantes, ya que proporcionan un contexto adicional que puede influir en el comportamiento de las series temporales.

### Temporal processing

---

#### Interpretable Multi-Head Attention

El TFT utiliza un mecanismo de autoatención (self-attention) para aprender relaciones a largo plazo entre diferentes momentos en el tiempo. Este mecanismo está basado en la atención multi-cabeza de las arquitecturas de transformadores (2.5), pero ha sido modificado para mejorar la explicabilidad. Cada cabeza de atención puede aprender diferentes patrones temporales mientras analiza un conjunto común de características de entrada.

#### Temporal Fusion Decoder

El 'temporal fusion decoder' utiliza las siguientes capas para aprender las relaciones temporales presentes en el conjunto de datos:

- **Locality Enhancement with Sequence-to-Sequence Layer:** En los datos de series temporales, los valores significativos a menudo se identifican en relación con sus valores vecinos, como anomalías, puntos de inflexión o patrones cílicos.

Aprovechar el contexto local, construyendo características que utilicen información sobre patrones en lugar de solo valores puntuales, puede mejorar el rendimiento de las arquitecturas basadas en la atención.

Las arquitecturas basadas en la atención a menudo no consideran situaciones donde el número de entradas pasadas y futuras difiere. Por ello, el TFT utiliza un modelo secuencia a secuencia para manejar estas diferencias de manera natural, introduciendo los datos pasados en el codificador y los datos futuros conocidos (known-futures) en el decodificador.

Además, se considera el uso de un codificador-decodificador LSTM, aunque también se pueden adoptar otros modelos.

Esto también actúa como un sustituto de la codificación posicional estandar, proporcionando un sesgo inductivo adecuado para el orden temporal de las entradas.

- **Static Enrichment Layer:** Dado que las variables estáticas suelen tener un impacto importante en la dinámica temporal (por ejemplo, días festivos), se agrega una capa de enriquecimiento estático. Esta capa mejora las características temporales al incluir metadatos estáticos.
- **Temporal Self-Attention Layer:** Despues de enriquecer las características estáticas se aplica una capa de autoatención (self-attention). Primero, todas las características temporales enriquecidas estáticamente se agrupan

en una única matriz, sobre la cual se aplica atención multi-cabeza interpretable en cada momento de pronóstico. Para garantizar que cada dimensión temporal solo pueda atender a características que la preceden, se aplica un enmáscaramiento del decodificador a la capa de atención multi-cabeza.

Además de preservar el flujo de información causal mediante el enmáscaramiento, la capa de autoatención permite que TFT capture dependencias a largo plazo, algo que puede ser desafiante para arquitecturas basadas en RNN.

Después de la capa de autoatención, también aplicamos una capa de control adicional para facilitar el entrenamiento.

- **Position-wise Feed-forward Layer:** Se aplica un procesamiento no lineal adicional a las salidas de la capa de autoatención. Al igual que la capa de enriquecimiento estático, esto hace uso de GRNs, donde los pesos de GNR se comparten en toda la capa.

Además, aplicamos una conexión residual con compuerta que abarca todo el bloque del transformador. Esto proporciona un camino directo a la capa de secuencia a secuencia, lo que resulta en un modelo más simple cuando no se necesita complejidad adicional.

- **Quantile Outputs:** TFT también genera intervalos de predicción sobre las predicciones puntuales. Esto se logra mediante la predicción simultánea de varios percentiles (por ejemplo, el 10, 50 y 90) en cada paso de tiempo. Los pronósticos de cuantiles se generan utilizando la transformación lineal de la salida del decodificador de fusión temporal.

Como podemos ver los modelos TFT pueden ser muy poderosos en la predicción de series temporales, pero además estos modelos tienen la ventaja de ser interpretables, lo cual es un aspecto muy positivo frente a los modelos de caja negra (black-box).

Existen tres casos de uso de interpretabilidad:

- Examinar la importancia de cada variable de entrada en la predicción.
- Visualizar patrones temporales persistentes
- Identificar regímenes o eventos que conducen a cambios significativos en la dinámica temporal.

Tras esta introducción a los TFT podemos concluir que son modelos muy potentes y adaptables a múltiples dominios y casos de uso. Aun así, también hay que mencionar que al ser una arquitectura tan nueva aun no hay mucha documentación al respecto y la curva de aprendizaje puede ser pronunciada.

Toda la formulación matemática puede ser consultada en [5].

La arquitectura de un TFT, con todos los componentes recien descritos, puede ser consultada en la fig. 2.9.

## Capítulo 2. Fundamentos Teóricos

---

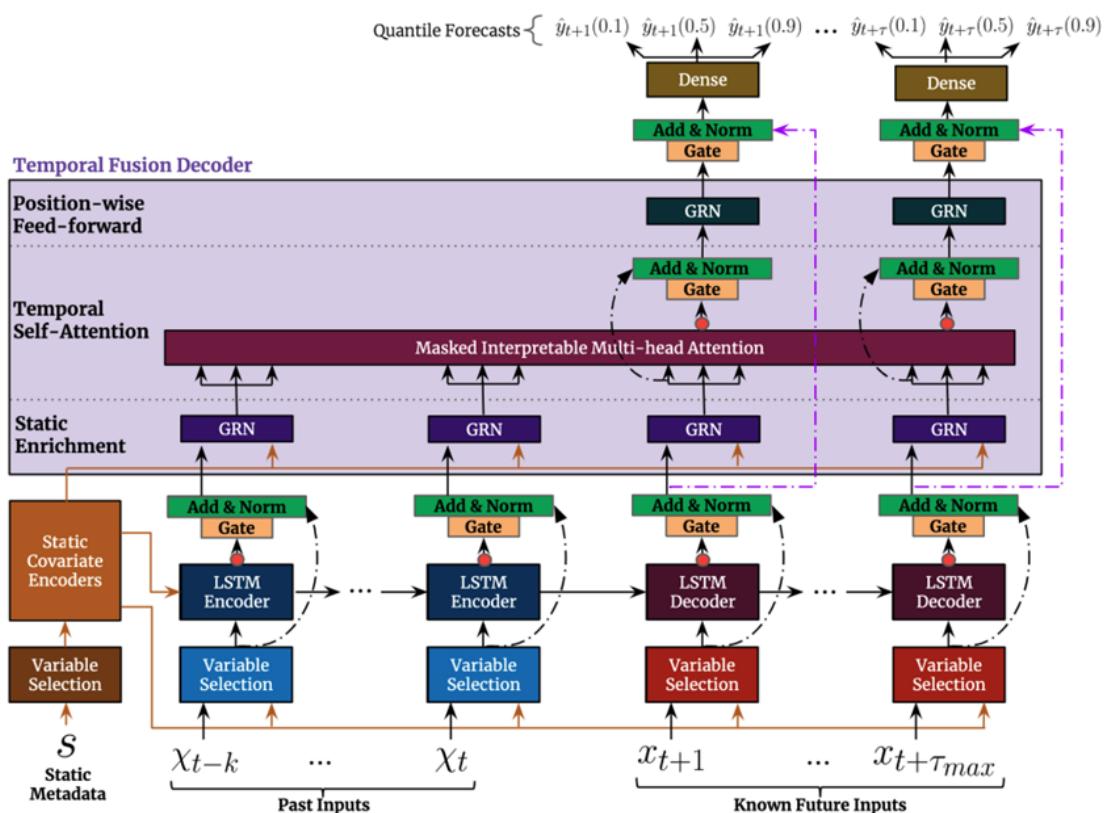


Figura 2.9: Arquitectura TFT. Figura sacada de [5].

### 2.6. Métricas para series temporales

Para evaluar los resultados y comparar los modelos deberemos hacer uso de métricas de error. Estas también podrán ser usadas a la hora de entrenar los modelos con el objetivo de ser minimizadas, a modo de funciones de pérdida (loss). Introduciremos las métricas que usaremos:

Sean:

$$\begin{aligned} y_i &= \text{Valor real} \\ \hat{y}_i &= \text{Predicción} \end{aligned}$$

Definimos:

- **Criterio de Información de Akaike (AIC):** El Criterio de Información de Akaike (AIC) es una medida estadística que evalúa la calidad relativa de un modelo estadístico. Se utiliza comúnmente en la selección de modelos, especialmente en el contexto de series temporales. El AIC proporciona un equilibrio entre la bondad del ajuste del modelo y su complejidad, permitiendo comparar diferentes modelos y seleccionar el más adecuado.

Para un modelo ARIMA, el AIC se calcula utilizando la siguiente fórmula:

$$\text{AIC} = -2 \log(L) + 2p$$

donde:

- $L$  es la verosimilitud<sup>2</sup> del modelo, que mide qué tan bien el modelo se ajusta a los datos observados.
- $p$  es el número de parámetros en el modelo. En el caso de un modelo ARIMA, esto incluiría los coeficientes de autorregresión, los coeficientes de medias móviles, el orden de diferenciación y cualquier constante o tendencia.

El AIC penaliza la inclusión de parámetros adicionales en el modelo, lo que ayuda a evitar el sobreajuste. En la práctica, se prefiere un valor de AIC más bajo, lo que indica un mejor equilibrio entre la bondad del ajuste y la complejidad del modelo.

El AIC se utiliza comúnmente junto con otras métricas de evaluación de modelos, como el Error Absoluto Medio (MAE) y el Error Cuadrático Medio (MSE). En nuestro caso usaremos tan solo el AIC en la evaluación de modelos ARIMA.

- **Error Absoluto Medio (MAE):** El MAE es una métrica que calcula el promedio de las diferencias absolutas entre las predicciones del modelo y los

---

<sup>2</sup>Si denotamos los datos observados como  $y_1, y_2, \dots, y_n$  y los pronósticos del modelo como  $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$ , entonces la función de verosimilitud del modelo ARIMA se calcula como la probabilidad conjunta de observar los datos  $y_1, y_2, \dots, y_n$  dada la especificación del modelo y los parámetros estimados.

## Capítulo 2. Fundamentos Teóricos

---

valores reales de la serie temporal. Será una métrica muy importante en nuestro trabajo pues en nuestro caso refleja en cuántos € erramos de media en un periodo de tiempo. Se expresa matemáticamente como:

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

- **Error Porcentual Absoluto Medio (MAPE):** El MAPE es una métrica que calcula el promedio de los errores porcentuales absolutos entre las predicciones del modelo y los valores reales de la serie temporal. El MAPE es útil para evaluar el rendimiento del modelo en términos de porcentaje de error en relación con los valores reales. Aun así, para el tramo temporal en el que entrenamos y evaluamos el modelo, el MAPE no es posible calcularlo por su formulación matemática:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\%$$

Como podemos ver se divide la diferencia absoluta entre el valor predecido y el real entre el valor real, por lo que no es posible aplicarlo a un escenario como el nuestro en el que hay presentes valores iguales a 0.

La introducimos aun así en este apartado pues en series temporales tan volátiles como la nuestra el MAE puede no reflejar bien el rendimiento del modelo al haber precios tan dispares. Sin embargo, en un tramo pasado o futuro de nuestra serie temporal esta métrica sería valida, y posiblemente, la más adecuada.

- **Error Cuadrático Medio (MSE):** El MSE es una métrica que calcula el promedio de los cuadrados de las diferencias entre las predicciones del modelo y los valores reales de la serie temporal. El MSE penaliza de manera significativa los errores grandes. En nuestra serie temporal volátil, el MSE puede proporcionar una medida más robusta del rendimiento del modelo al tener en cuenta la magnitud de los errores. Se expresa matemáticamente como:

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

donde  $y_i$  es el valor real en el tiempo  $i$ ,  $\hat{y}_i$  es la predicción del modelo en el tiempo  $i$ , y  $n$  es el número total de observaciones.

- **Métrica Ad Hoc:** Como hemos mencionado previamente el MAPE no puede ser usado en nuestro trabajo, el MAE puede ser útil sobre todo para evaluar el modelo, pero no para entrenarlo, pues tiende a predecir valores intermedios entre los mínimos y máximos de la secuencia de entrenamiento. El

## 2.6. Métricas para series temporales

---

resto de métricas experimentalmente se ha comprobado que ni servían para evaluar el modelo ni para entrenarlo correctamente.

Por todo ello en este trabajo se desarrolla una métrica ad hoc para el problema al que nos enfrentamos. Como vimos en la fig. 3.4 la evolución horaria del precio SPOT consta principalmente de 2 picos, uno entorno a las 7:00 y otro entorno a las 20:00, y entre ellos se presenta un valle. Puesto que la dificultad se halla principalmente en predecir la magnitud de los 2 picos (y la progresiva ascensión hasta ellos en el entorno horario) definimos una métrica que penalice a diferentes niveles de exponencialidad el error a unas horas u otras.

Definimos matemáticamente nuestra métrica de la siguiente forma:

$$\text{custom\_loss} = \sum_{i=1}^{24} |y_i - \hat{y}_i|^{w_i}$$

donde:

$$\mathbf{w} = [2, 1, 1, 1, 1, 1, 3, 3, 3, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 3, 3, 2, 2]$$

$y_i$  = Valor real

$\hat{y}_i$  = Predicción

Esta métrica se usa como función de perdida a la hora de entrenar los modelos. **No nos sirve como métrica de evaluación** ya que por el distinto factor exponencial al que se miden los errores en cada hora puede haber modelos con buena capacidad predictiva que se ven muy penalizados por una única mala predicción en una hora con alto peso

- **Quantile Loss:** Esta métrica tan solo sera usada como función de perdida al entrenar los Temporal Fusion Transformers ya que estos modelos, como explicamos en 2.5.1 realizan predicciones en diferentes cuantiles. La quantile loss es una métrica utilizada principalmente en modelos de regresión cuantílica, donde el objetivo es predecir un cuantil específico de la distribución de la variable de interés. Esta métrica es particularmente útil en situaciones donde se desea capturar la incertidumbre y variabilidad de las predicciones. La Quantile Loss se define matemáticamente como:

$$\text{Quantile Loss} = \sum_{i=1}^n (q \cdot \max(y_i - \hat{y}_i, 0) + (1 - q) \cdot \max(\hat{y}_i - y_i, 0))$$

donde  $n$  es el número de observaciones y  $q$  es el cuantil que se desea predecir, un valor entre 0 y 1. Por ejemplo,  $q = 0,5$  corresponde a la mediana.

Esta métrica permite ajustar el modelo de manera que se minimicen los errores asimétricamente, lo cual es útil cuando se desea dar más peso a sobreestimaciones o subestimaciones, dependiendo del valor del cuantil  $q$ . En nuestro trabajo, utilizamos la Quantile Loss para capturar mejor la

## **Capítulo 2. Fundamentos Teóricos**

---

variabilidad de los precios, realizar predicciones más robustas en diferentes escenarios y tener una mejor interpretabilidad del modelo.

A modo de conclusión, en este trabajo utilizaremos el AIC junto al MAE para evaluar los modelos ARIMA. Para los modelos LSTM y CNN, emplearemos nuestra métrica Ad Hoc durante el entrenamiento y el MAE (y el MSE en menor medida) para evaluarlos. Finalmente, el modelo TFT será entrenado y evaluado utilizando la Quantile Loss. La comparación final de todos los modelos se realizará con el MAE, siendo considerado el mejor modelo aquel que tenga el MAE más bajo.

# **Capítulo 3**

## **Desarrollo**

En este capítulo se mostrará que herramientas y metodologías fueron usadas para lograr el desarrollo del trabajo.

### **3.1. Planteamiento**

Primero de todo debemos conceptualizar el trabajo. Como explicamos en capítulos anteriores el objetivo de este trabajo es ser capaz de predecir un día 'D' el precio SPOT (por horas) de la energía en el mercado mayorista diario español del día 'D+1'.

### **3.2. Obtención de datos**

Para comenzar, es fundamental establecer el método de adquisición de datos para el entrenamiento del modelo. Además de basarnos en estudios previamente realizados ([16] y [17]), hemos explorado otras variables mediante la formulación de hipótesis y análisis empíricos, con el fin de enriquecer nuestra investigación y ampliar nuestro enfoque analítico. Como resultado de este proceso, hemos identificado las variables más influyentes sobre el precio energético:

- **Precio del gas:** Precio del gas en el Punto Virtual de Balance (PVP) (MWh). [18]
- **Derechos de emisión de CO<sub>2</sub> (EUA):** Los EUA (European Union Allowance) son los derechos que otorgan a una entidad o empresa la capacidad de emitir una cantidad específica de gases de efecto invernadero, se mide en €/tonelada. [19]
- **Generación eólica:** Generación medida en tiempo real del tipo de producción eólica (MW) [20].
- **Generación solar:** Generación medida en tiempo real del tipo de producción solar (MW) [20].
- **Demandado:** valor real de la demanda de energía eléctrica en tiempo real (MW) [20].

## Capítulo 3. Desarrollo

---

- **Demanda residual:** Previsión resultante de restar a la previsión de la demanda diaria las previsiones solares y eólicas (MW) [20].
- **Rampa:** Se crea esta variable que mide la diferencia en demanda la residual entre un período y el período anterior. Esto es útil para identificar cambios bruscos, tendencias y patrones en la demanda residual. (MW)

Estas serán las variables de las que dispondremos, así como sus previsiones, para generar el dataset de entrenamiento.

En cuanto a las fuentes decidimos hacer uso de fuentes oficiales como ESIOS (Sistema de Información del Operador del Sistema), Sendeco2 (Sistema europeo de negociación de CO2) o MIBGAS (Mercado Ibérico del Gas), así como de grandes empresas como ICE (Intercontinental Exchange) para minimizar el riesgo de no poder obtener los datos u obtener datos erróneos.

La mayoría de datos, para obtenerlos gratuitamente, deberán ser obtenidos de las páginas web oficiales, por lo que haremos web scraping para obtenerlos. Alternativamente, y si el servicio es gratuito, optaremos por hacer consultas a APIs para obtener la información. Por tanto, nuestras fuentes serán:

- **Web Scraping:**
  - MIBGAS: Precio y previsión de gas
  - Sendeco2: Historico de precios de CO2
  - ICE: Precio y previsión de CO2
- **API:**
  - ESIOS: Precio SPOT, producción y previsión de renovables (solar y eólica), demanda (y previsión) y demanda residual.

Cabe mencionar que todos los datos usados en este trabajo son de dominio público. El código usado en esta sección es propiedad de NTT DATA EMEAL - Business Analytics, por lo que este no será publicado por acuerdos de confidencialidad con la empresa.

### 3.3. Análisis descriptivo de variables (EDA)

Antes de construir nuestro dataset de entrenamiento deberemos realizar un análisis descriptivo de las variables que tenemos para identificar posibles anomalías o errores en los datos.

Se llevará a cabo el EDA sobre la variable objetivo además de las variables exógenas:

- Precio SPOT
- Demanda
- Precio gas
- Prod. eólica

### 3.3. Análisis descriptivo de variables (EDA)

- Prod. solar
- Demanda residual
- Rampa

En primera instancia visualizamos la serie temporal completa de la variable objetivo, el precio SPOT, de la que disponemos desde el 1 de enero de 2019. Debido a la gran volatilidad diaria añadiremos una media móvil diaria, una semanal y otra mensual para entender mejor la evolución de la serie temporal.



Figura 3.1: Serie Temporal de precio SPOT

Lo que primero llama la atención de la gráfica, además de su volatilidad diaria, son los valores centrales de la misma. Dicho tramo temporal corresponde al año 2022, en el que el mercado energético se vio fuertemente afectado por la guerra de Ucrania debido al corte de suministro de gas en Rusia. Previo a este año apreciamos como el precio se mantuvo más o menos constante hasta 2021, momento en el que comienza un paulatino aumento del precio energético hasta que alcanza su máximo el día 8 de marzo de 2022 a las 19:00 con un valor de 700 €/MVh.

Puesto que el mercado actual no se ve tan afectado por el contexto geo-político descrito consideramos conveniente descartar este tramo temporal con el fin de no confundir a los modelos con relaciones ahora inexistentes entre variables.

La decisión de fijar a partir de cuando considerar la serie temporal no es trivial. Con ayuda de gráficas decidimos fijar el corte 365 días atrás, quedándonos por tanto con los datos del ultimo año, desde el 22 de abril de 2023 hasta el 22 de abril de 2024. Puede que un modelo use más o menos días que estos, pero las distribuciones de las variables serán prácticamente las mismas. Por ello podemos con total confianza realizar el análisis estadístico de variables a partir del 22 de abril de 2023 sin perdida de generalidad alguna.

Observamos la gráfica del ultimo año.

## Capítulo 3. Desarrollo

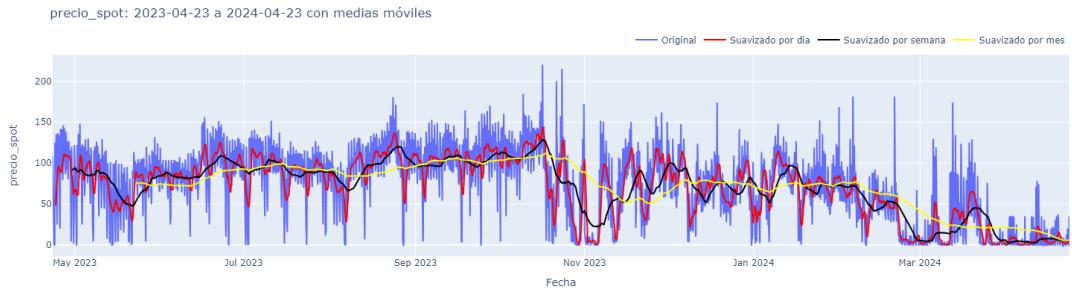


Figura 3.2: Ultimo 365 días

Observamos como ahora la serie temporal es algo más uniforme pese a su alta volatilidad. Procedemos a visualizar la distribución del precio energético por mes.

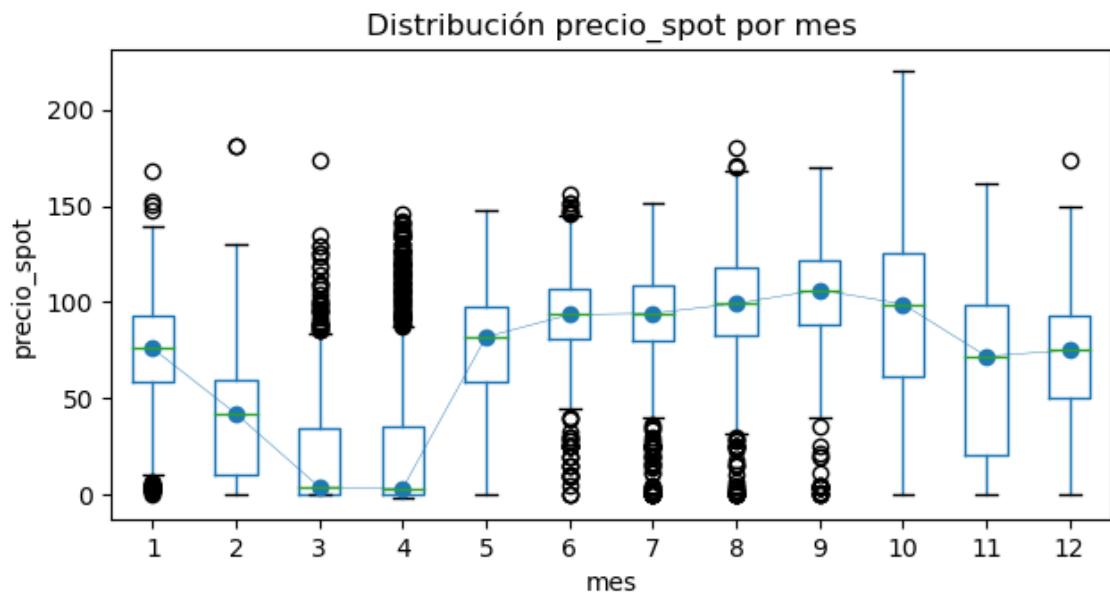


Figura 3.3: Diagrama de cajas precio SPOT por mes

En él podemos observar como efectivamente marzo y abril son los meses con valores más bajos. Esto es debido a que en 2024 en estos meses han habido muchos valores iguales a 0. Por otro lado podemos ver como en los meses de verano (de junio a septiembre) es cuando más valores atípicos hay. También ocurre en los meses de marzo y abril pero es debido a lo comentado anteriormente, hay muchos 0 que hacen que los picos (normales en la serie temporal) parezcan atípicos.

Ahora procedemos a visualizar la distribución por horas:

### 3.3. Análisis descriptivo de variables (EDA)

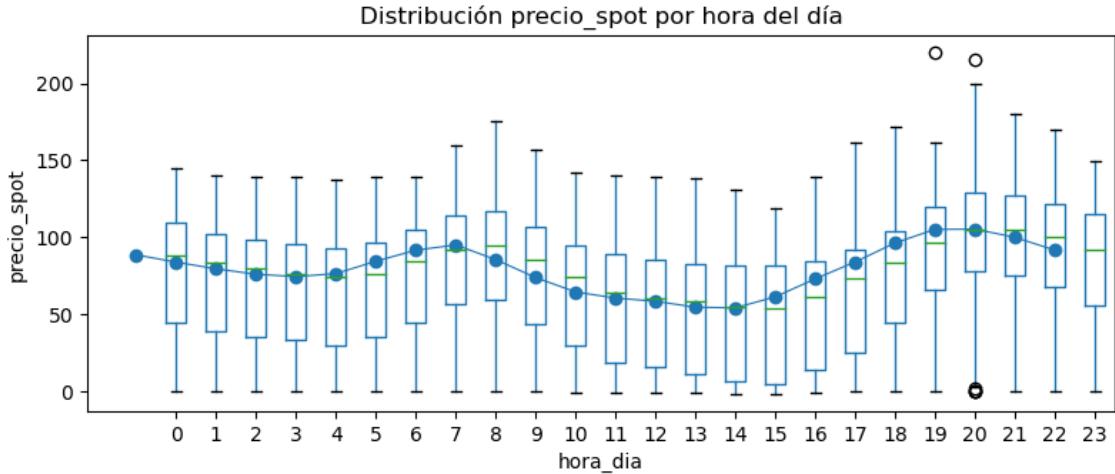


Figura 3.4: Diagrama de cajas precio SPOT por hora

Visualizando la distribución diaria podemos hacernos una idea de la forma general de la serie temporal diaria, esto es, la forma que tratamos de predecir. Como se puede observar, y como veremos próximamente, la serie temporal consta de 2 tramos horarios (alrededor de las 7h y alrededor de las 20h) en los que generalmente los precios son más elevados (picos), y entre estos tramos el precio suele ser más bajo y estable (valles).

Con ayuda de la función de autocorrelación (ACF) y autocorrelación parcial (PACF) estudiamos la estacionalidad de la serie:

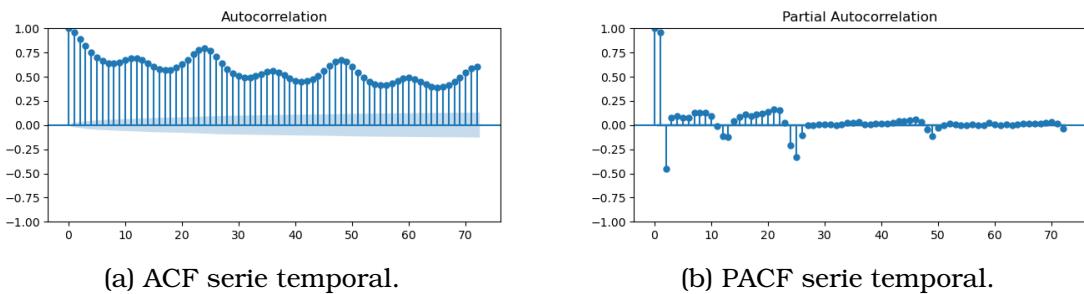


Figura 3.5: (a) ACF, (b) PACF de la serie temporal.

Como podemos ver tanto en la ACF como en la PACF los retardos múltiples de 24 son significativos, por ello podemos concluir que nuestra serie temporal tiene estacionalidad diaria. Esto tiene sentido puesto que la serie temporal diaria tiene una forma que se repite día a día (valle entre 2 picos).

Procedemos ahora a visualizar las relaciones entre variables y sus distribuciones mediante gráficos de densidad univariada y gráficos de dispersión.

## Capítulo 3. Desarrollo

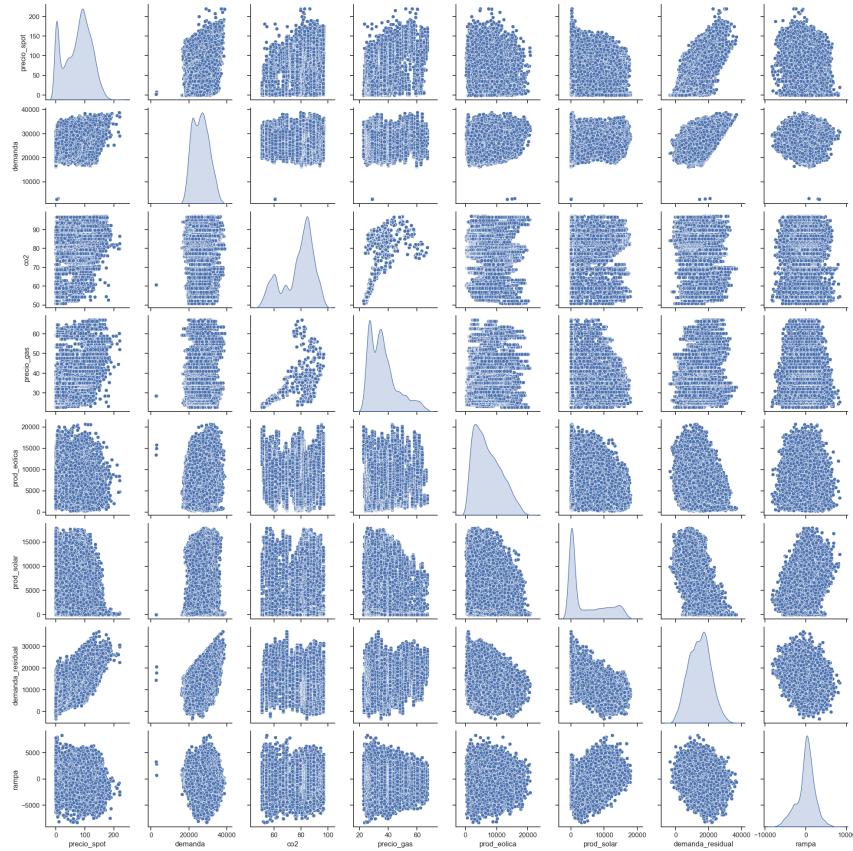


Figura 3.6: Gráficos de densidad univariada junto con gráficos de dispersión

Como podemos ver en general no siguen una distribución normal, aun así hay alguna como la demanda, demanda residual o rampa que vagamente la imitan. Pese a que la distribución de producción solar pueda parecer extraña o con valores erróneos en un primer momento, esta sí tiene sentido puesto que la mayoría de los valores se agrupan en los valores más bajos, indicando las horas de noche o poca luz que ocurren todos los días mientras que horas con mucho sol solo se dan a ciertas horas ciertos días del año.

Las variables en general no se ven muy correlacionadas entre si. El precio SPOT parece correlacionarse con la demanda residual en mayor medida . También se aprecia una correlación positiva entre los precios del CO2 y del gas, y entre la demanda residual y la producción eólica. Posteriormente corroboramos estas conclusiones en la matriz de correlación (fig. 3.7).

El EDA completo para cada una de las variables puede ser consultado en el repositorio de Github.

### 3.4. Preprocesado

Una vez tenemos nuestro dataset elaborado y hemos realizado el EDA procedemos a realizar el preprocesado necesario.

### **3.4.1. Tratamiento de nulos**

Este caso no se da puesto que los datos se obtienen de fuentes oficiales fiables. Aun así como cabe la posibilidad de que alguna fuente no este disponible en un momento concreto decidimos tratar estos valores 'nulos' rellenando los valores faltantes con los últimos dados. Por ejemplo si un día no podemos obtener el precio del CO<sub>2</sub> lo que haríamos seria suponer que el precio de hoy es el mismo del que fue ayer. Optamos por esta estrategia en todas las variables pues la variabilidad en ellas de un día al siguiente es baja.

Pese a que esta imputación no se haga posterior a la elaboración del dataset sería equivalente a llenar los valores faltantes con los anteriores (forward fill).

### **3.4.2. Manejo de valores atípicos**

Al realizar el EDA de las variables nos encontramos con ciertos valores atípicos primeramente en la columna 'demanda', donde tomaba valor 0 en unas horas de septiembre de 2023 y otros en unas horas de marzo de 2024. Al observar esas filas vimos como también tomaban valor 0 la producción solar y eólica simultáneamente. Estos 2 últimos valores en sí no parecen ser atípicos ni faltantes puesto que son valores que sí pueden tomar, sin embargo al tratarse de horas centrales del día (con luz) sumado a los valores iguales a 0 también en demanda concluimos que esos datos eran erróneos.

Al tratarse de tan solo 5 filas del histórico decidimos imputar manualmente los valores con valores similares a los del día anterior y el día próximo a esa misma hora.

### **3.4.3. Selección de variables**

Puesto que contamos con pocas variables, en principio, no descartaremos ninguna de ellas a no ser que sea estrictamente necesario. Observamos la matriz de correlación (fig. 3.7) para observar si hay variables muy correlacionadas, lo cual perjudicaría al modelo ya que habría varias variables que contienen la misma información, y por tanto mayor influencia que el resto de variables al estar repetidas.

Observando la matriz de correlación (fig. 3.7) confirmamos las conclusiones que sacamos a priori observando la fig. 3.6. La mayor correlación existente es entre nuestra variable objetivo 'precio spot' y la demanda residual con valor 0.66, seguido del co2 con valor 0.55.

En cuanto a las variables exógenas la correlación más alta es de 0.55 entre la producción solar y la demanda residual, por tanto no creemos oportuno descartar ninguna variable exógena.

### **3.4.4. Normalización de datos**

La normalización de datos generalmente deberá ser tratada dentro de cada modelo específico puesto que hay modelos que no necesitan normalización. Sin

## Capítulo 3. Desarrollo

---

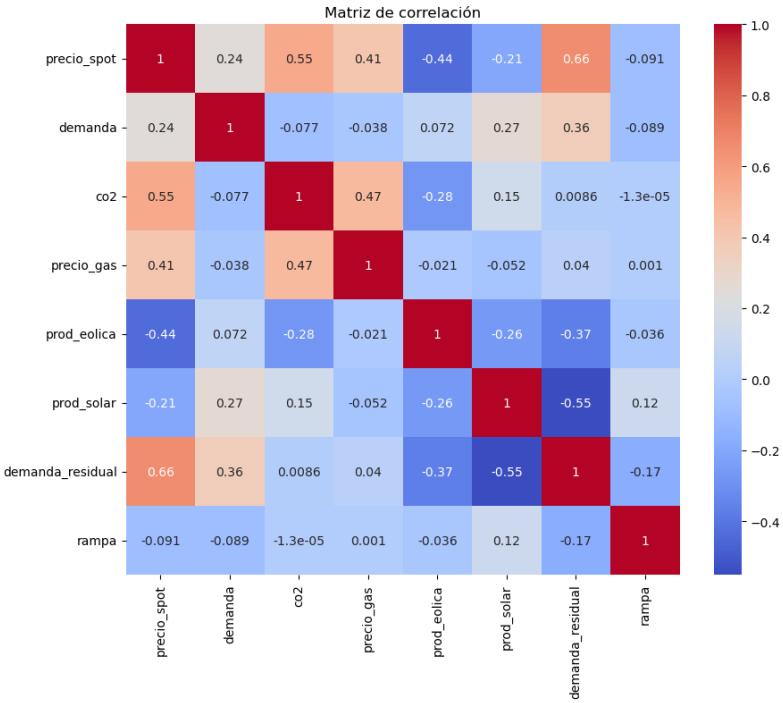


Figura 3.7: Matriz de correlación

embargo, en nuestro caso todos los modelos se benefician de una normalización de datos.

Para normalizar los datos visualizamos la distribución de cada variable (fig. 3.6) para así hallar aquella normalización más adecuada.

Optamos por llevar a cabo una estandarización de los datos siempre que sea posible puesto que hay variables como la producción solar y el precio del gas que cuentan con valores atípicos muy alejados de su mediana, lo cual produciría que la mayoría de los valores de esas variables al normalizarlas mediante MinMax quedasen próximas a 0. Dicho esto, y como se puede ver en el código, cada modelo experimentalmente puede dar mejores resultados con otras normalizaciones.

### 3.5. Modelado

Tras haber realizado un análisis de los diferentes modelos existentes decidimos implementar los siguientes por su idoneidad en el dominio de series temporales:

- **Modelos clásicos:**

- (S)ARIMA(X)

- **Modelos neuronales:**

- Long Short-Term Memory (LSTM)

- Convolutional Neural Networks (CNN) - 1D
- Temporal Fusion Transformers (TFT)

#### 3.5.1. (S)ARIMA(X)

En primer lugar, antes de pasar a los modelos más complejos y novedosos, comenzamos con un enfoque más clásico, (S)ARIMA(X). Empíricamente vimos que obteníamos resultados ligeramente mejores si tan solo entrenábamos usando los datos desde el 1 de enero de 2024 en vez de desde abril de 2023.

Decidimos inicialmente elaborar 2 modelos SARIMAX, uno siguiendo la metodología de Box-Jenkins descrita en [21], y otro realizando una búsqueda de hiperparámetros.

Como describimos en el apartado de Fundamentos Teóricos el modelo ARIMA tiene como supuesto que nuestra serie temporal es estacionaria en media y en varianza, i.e, que la media y la varianza son estables en el tiempo.

Al descomponer la serie temporal (fig. 3.8) vemos rápidamente que no es estacionaria (tanto la media como la varianza disminuye con el tiempo), por lo que procedemos a transformar nuestra serie temporal con el objetivo de tener una serie estacionaria siguiendo los supuestos de ARIMA.

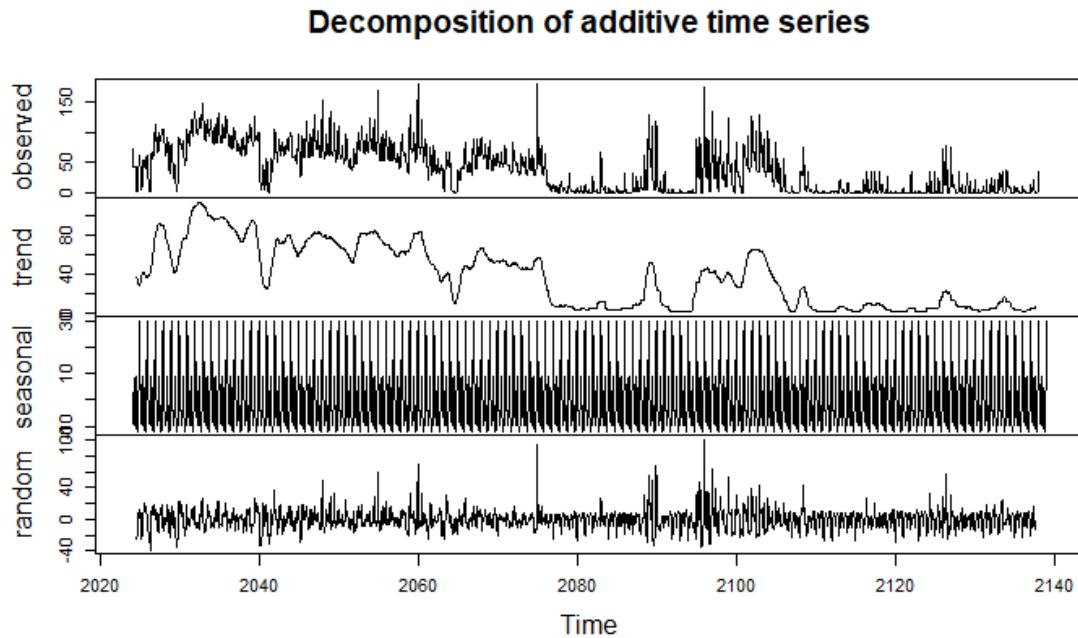


Figura 3.8: Descomposición de la serie temporal

Una vez se realizan las transformaciones necesarias se establecen los parámetros del modelo. Observando las ACF y PACF construimos un modelo SA-

## Capítulo 3. Desarrollo

---

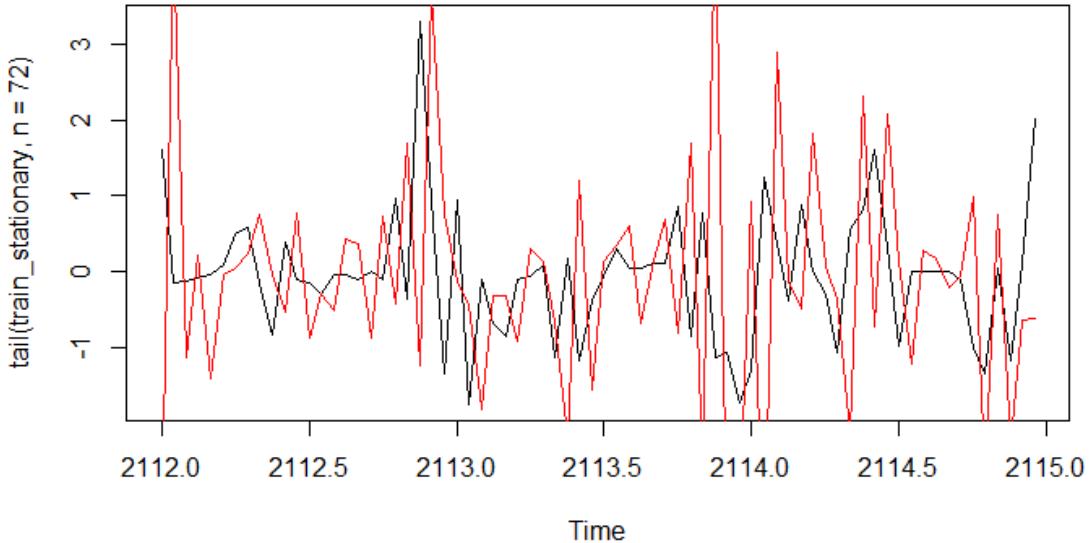


Figura 3.9: Modelo SARIMAX(0,1,0)x(0,1,0)24

RIMAX(0,1,0)x(0,1,0)24. Con una búsqueda de hiperparámetros obtenemos un modelo SARIMAX(2,0,0)(2,0,0)24

Una vez se entrena los modelos y se realiza su diagnosis, procedemos a realizar predicciones en un conjunto de validación para ver la capacidad predictiva de los modelos. Ambos modelos, dieron unos resultados insatisfactorios. Podemos visualizar los ajustes de los modelos SARIMAX(0,1,0)x(0,1,0)24 y SARIMAX(2,0,0)(2,0,0)24 a nuestra serie temporal (fig. 3.9 y fig. 3.10).

Tanto el estudio siguiendo la metodología de Box-Jenkins [21] como las predicciones puede ser consultado en los anexos.

Puesto que los modelos se ajustaban francamente mal a la serie temporal decidimos probar suerte elaborando un modelo SARIMAX obviando toda la base teórica (supuestos de ARIMA).

Entrenamos un modelo tras una búsqueda de hiperparámetros con la serie temporal directamente, sin tratar la estacionariedad ni realizando diagnosis del modelo. El modelo obtenido es un SARIMAX(1,1,4)(1,0,0)[24].

Sorprendentemente, como podemos ver en fig. 3.11, este modelo se ajusta mejor que los otros 2 anteriores a nuestros datos.

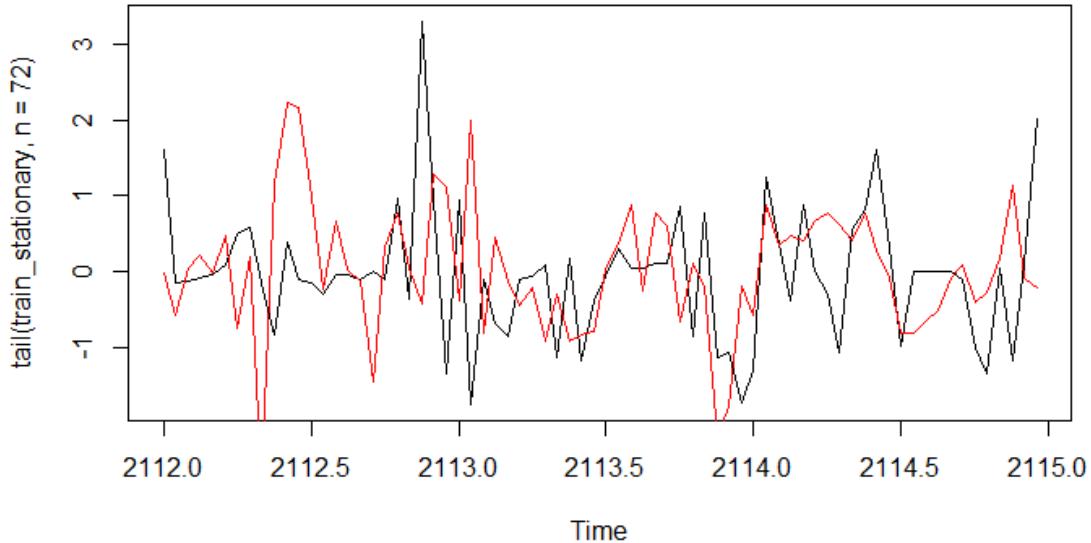


Figura 3.10: Modelo obtenido con búsqueda de hiperparámetros: SARIMAX(2,0,0)(2,0,0)[24]

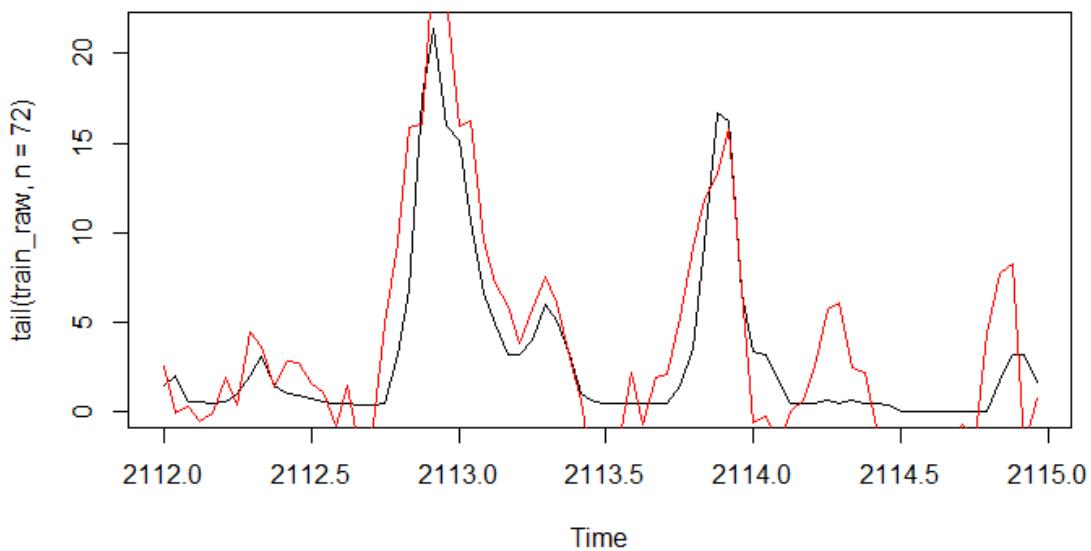


Figura 3.11: Ajuste del modelo (rojo) vs valores reales (negro). SARIMAX(1,1,4)(1,0,0)[24]

## Capítulo 3. Desarrollo

Este modelo claramente supera a los 2 anteriores. Procedemos a evaluarlo con los datos de validación.

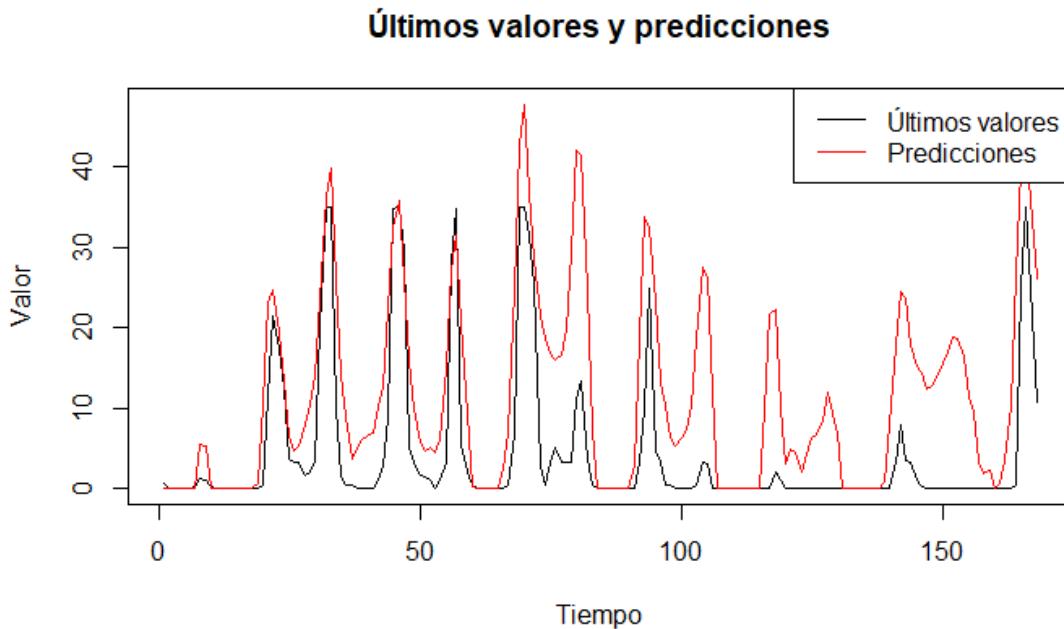


Figura 3.12: Predicciones (rojo) vs valores reales (negro) en datos de validación.  
SARIMAX(1,1,4)(1,0,0)[24]

Como podemos ver este modelo clásico da unos buenos resultados, obteniendo en validación un MAE de 7.13, lo cual implica que en estos 7 días de predicciones el modelo se equivoca de media en tan solo 7.13€.

Aunque los resultados hayan sido buenos hay un aspecto que no nos gusta al visualizar las predicciones. Tenemos la impresión de que el modelo siempre predice la misma forma, pero en distinta magnitud, lo cual puede no dar buenos resultados a largo plazo pese a que la métrica no sea muy alta.

### 3.5.2. LSTM

Ahora procedemos con modelos de redes neuronales, primeramente con un tipo de Red Neuronal Recurrente [10], las LSTM [3].

La modelización sera sencilla puesto que las LSTM no tienen muchos hiperparámetros que ajustar, principalmente serán el numero de unidades LSTM en la capa (a mayor numero permite al modelo capturar patrones más complejos en los datos, a costa de aumentar el coste computacional).

Además también se deben ajustar los hiperparámetros comunes a cualquier modelo de aprendizaje automático en series temporales como la longitud de las secuencias de entrenamiento, el numero de capas, la tasa de aprendizaje (learning

### 3.5. Modelado

rate), función de activación y el uso de capas con dropout o de regularización (L1,L2) para evitar el sobre ajuste.

Recordemos que en el entrenamiento de estos modelos neuronales (LSTM, CNN-1D) usamos una métrica Ad Hoc para entrenamiento y el MAE (y MSE en menor medida) para evaluar los modelos. Pese a que esta no sea una práctica muy común nos hemos visto forzados a realizarla puesto que, como mencionábamos en la sección 2.6, al usar las métricas usuales para entrenar (MAE, MAPE, MSE...) el modelo tendía a predecir la media para minimizar el error en vez de predecir correctamente los picos y los valles. Es por esta razón que empleamos esta práctica poco común. Además, como veremos a continuación, experimentalmente se corroboró que daba mejores resultados.

Tras realizar varios experimentos con distintas configuraciones determinamos que usaremos tan solo los últimos 90 días para entrenar y evaluar el modelo, quedándonos con 69 días de entrenamiento, 7 de validación y 15 de test. Esto lo hacemos así puesto que de coger más días el modelo predice valores que no se ajustan a la magnitud de los valores actuales.

Visualizamos la división de los datos en los conjuntos de train (azul), validación (verde) y test (rojo):

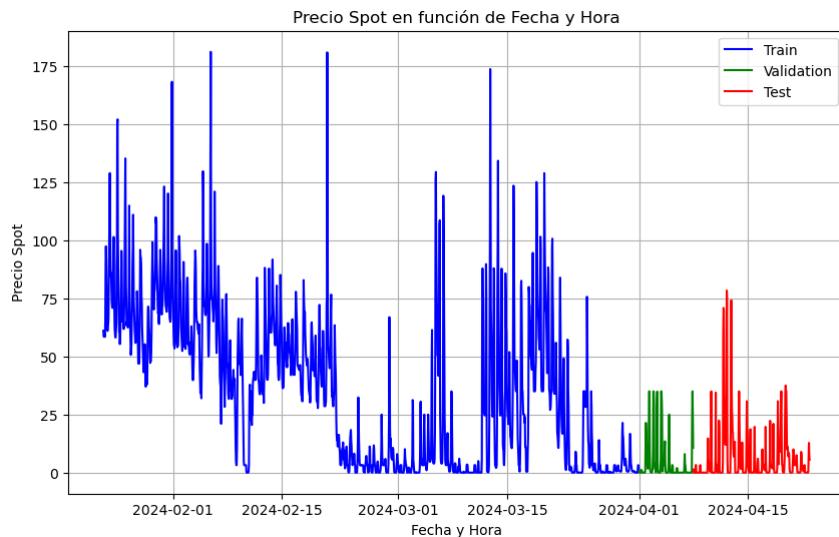


Figura 3.13: División de la serie temporal en train (azul), validación (verde) y test (rojo)

A pesar de que en el apartado 3.4.4 nos decidimos por una estandarización empíricamente nos ha dado mejores resultados una normalización MinMax, salvo para la variable objetivo para la cual mantenemos una estandarización.

Existen múltiples formas de crear secuencias dentro de nuestra serie temporal para el entrenamiento. La más común suele ser coger una secuencia de longitud 'n' para predecir el valor 'n+1', posteriormente desplazamos la serie en 1 unidad y volvemos a crear una secuencia de longitud 'n' en la que su último valor sera

## Capítulo 3. Desarrollo

---

'n+1' y se predecirá 'n+2'.

Esta sería una implementación correcta, pero no la más adecuada para nuestro problema. Nosotros, en primer lugar, debemos predecir los próximos 24 valores, no uno único. Además como hemos visto nuestra serie temporal tiene patrón temporal (valle entre 2 picos), que con la implementación anterior, se perdería. Por todo ello nuestra implementación es la siguiente:

- Tomamos los primeros 24 valores, comenzando a las 00:00 de un día 'D'.  
(x)
- Con estos valores se predecirán los 24 próximos. (y)
- El desplazamiento sera de 24, de esta forma la siguiente secuencia en vez de comenzar a la 01:00 de 'D' comenzara a las 00:00 de 'D+1'.

El hecho de tomar los primeros 24 valores es para predecir, como mínimo, con los datos del día anterior. Se probaron diferentes implementación y empíricamente se concluyo que lo que mejores resultados otorga es usando tan solo el día anterior.

Se compararon arquitecturas con distinta complejidad, desde unas básicas con una sola capa y un moderado numero de unidades a arquitecturas que combinan varias capas LSTM con capas neuronales densas. También se trato de realizar una búsqueda de hiperparámetros pero dio los peores resultados (posiblemente debido a la limitación computacional) por lo que fue descartado.

Tras probar con distintas arquitecturas llegamos a 3 prometedoras. La primera de ellas fue una red LSTM lo más sencilla posible, con tan solo una capa LSTM con 50 unidades y una capa densa de 24 neuronas (1 por cada hora del día) que sera nuestro output. Este modelo para ser tan sencillo da unos resultados buenos como vemos en la fig. 3.14, superiores incluso a los obtenidos por el modelo ARIMA, con un MAE de 5.7.

Posteriormente se llego a 2 arquitecturas más complejas que combinaban varias capas LSTM con varias capas neuronales densas. La primera de ellas no hacia uso de capas con Dropout ni con regularización, mientras que el segundo sí. Ambos dieron unos resultados muy similares, pero el modelo con dropout y regularización era ligeramente mejor.

Puesto que las métricas entre los 3 modelos, y sobre todo entre los complejos, son muy similares, procedemos a graficar las predicciones junto a los valores reales para ayudarnos a seleccionar un 'mejor modelo' (fig. 3.15).

### 3.5. Modelado

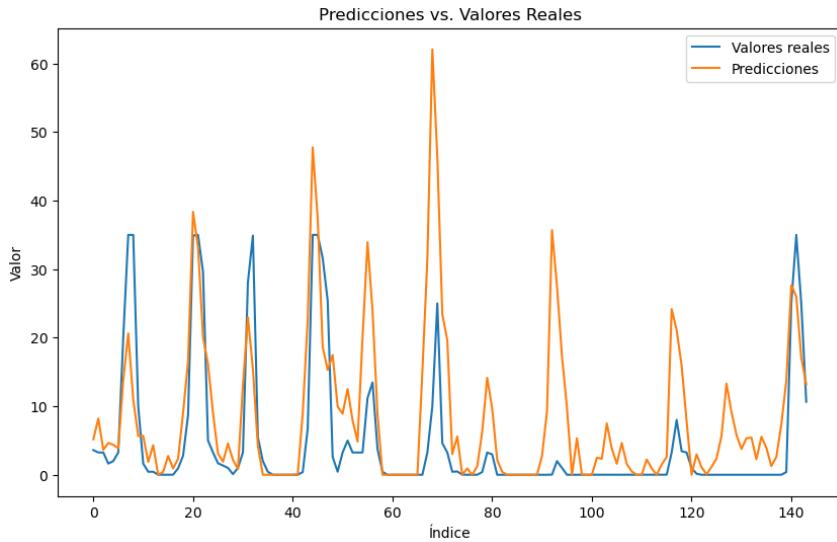


Figura 3.14: Predicciones (naranja) vs valores reales (azul) en validación. Modelo LSTM básico.

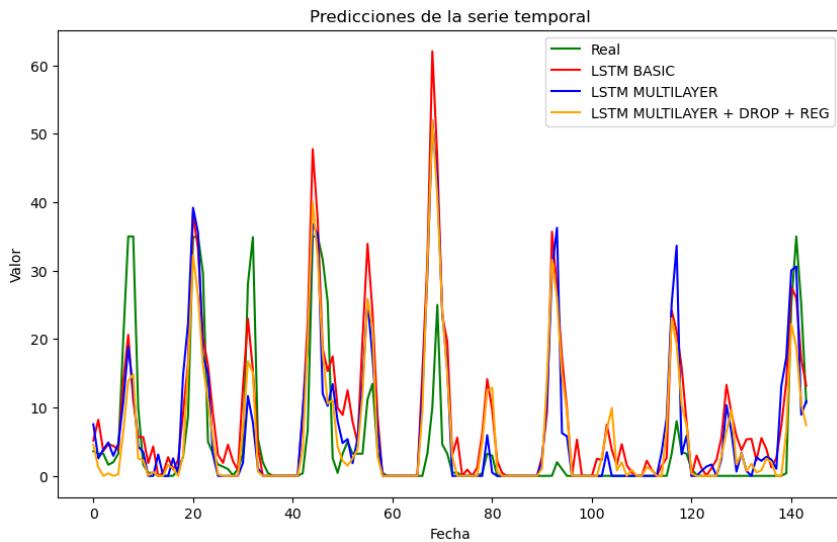


Figura 3.15: Comparativa modelos LSTM. Valores reales (verde), modelo básico (rojo), LSTM multicapa (azul) y LSTM multicapa con dropout + regularización (amarillo).

En la fig. 3.15 apreciamos como el modelo básico (rojo) tiende a sobreestimar los valores y es el que peores métricas obtiene, por tanto sera el primero en descartarse.

Entre los 2 modelos restantes no hay mucha diferencia, debido a que tiene un MSE menor (71.1 en vez de 82.4) y que fue entrenado con capas de regularización (que ayudan a generalizar mejor) optamos por seleccionar como modelo ganador el LSTM multicapa con dropout y regularización (amarillo).

## Capítulo 3. Desarrollo

Como podemos observar en fig. 3.16 obtenemos unas predicciones bastante prometedoras. En los 7 días de predicciones el modelo de media se equivoca en tan solo 4.63€ (MAE), lo cual es bastante positivo y superior a lo obtenido con el modelo ARIMA.

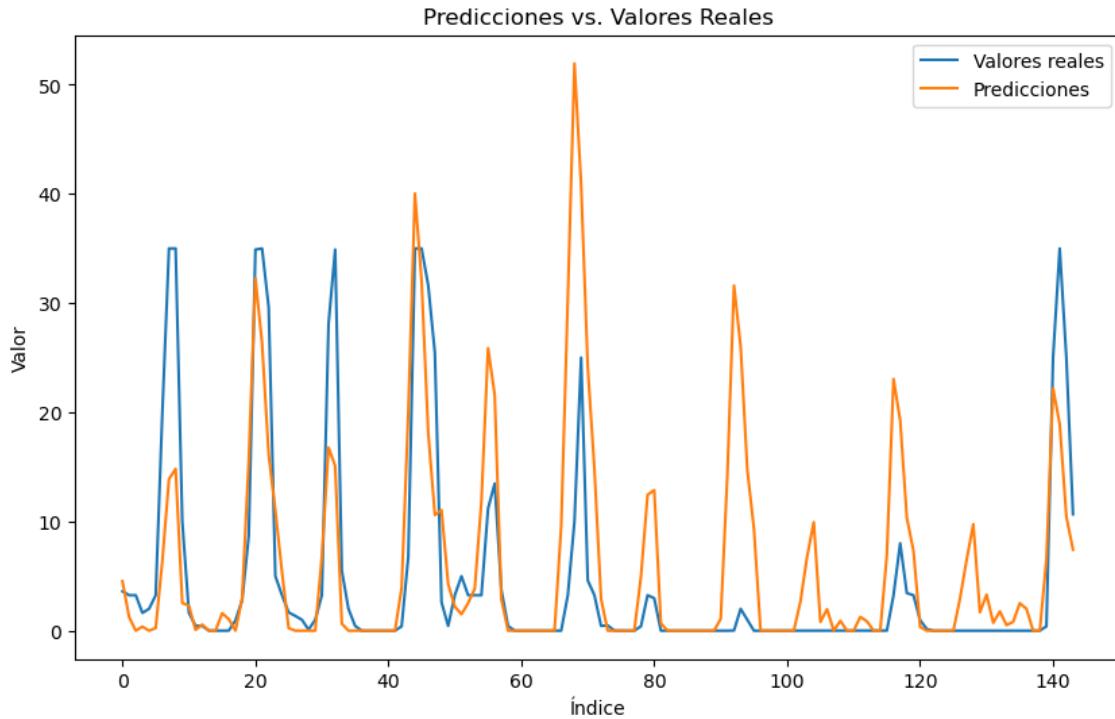


Figura 3.16: Predicciones (naranja) vs valores reales (azul) en validación. Modelo LSTM multicapa con dropout y regularización.

### 3.5.3. CNN-1D

Continuamos con los modelos neuronales, en este caso, las redes convolucionales unidimensionales (CNN-1D).

Seguiremos un planteamiento similar al anterior. En primer lugar implementaremos modelos sencillos para determinar si con estos (con bajos recursos computacionales) podemos realizar buenas predicciones. Paulatinamente aumentamos la complejidad del modelo con el fin de captar patrones más complejos que ayuden a realizar mejores predicciones.

Al igual que en el caso de las LSTM dividimos nuestra serie temporal en secuencias de 1 día (24 valores) y usamos 69 días para train, 7 para validación y 14 para test (fig. 3.13)

Comenzamos con el diseño de una CNN sencilla, con tan solo una capa convolucional con 16 filtros y una capa de MaxPooling de tamaño 2. Este primer modelo básico (fig. 3.17) obtiene unos resultados decentes, con un MAE de 6,05.

### 3.5. Modelado

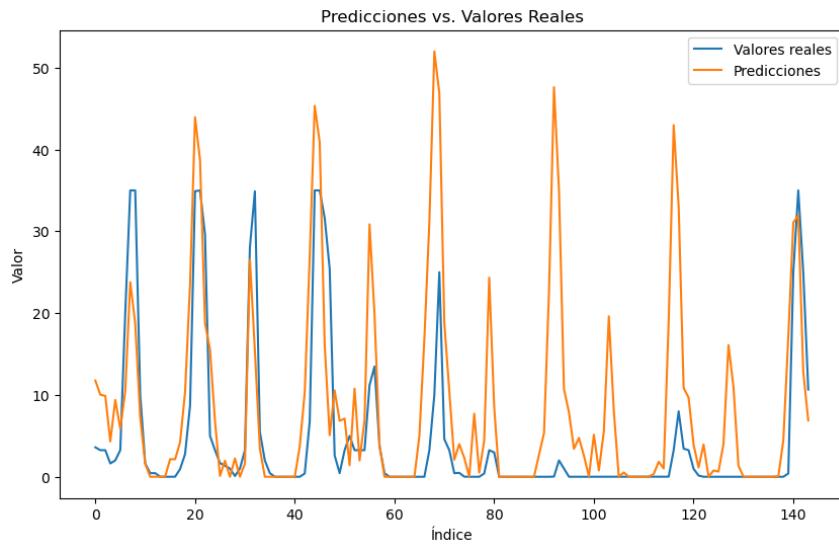


Figura 3.17: Validación vs predicciones CNN básica

Probamos ahora con modelos más profundos con el fin de lograr una mayor capacidad predictiva. Computacionalmente serán más costosos pero pueden captar patrones más complejos.

Modelamos una CNN con 3 capas convolucionales, las 2 primeras con 64 filtros y la ultima con 128, seguido de una capa densa de 64 neuronas conectada a la capa densa de salida con 24 neuronas, una por cada hora del día a predecir.

Como podemos ver en fig.3.18 este modelo da unos resultados mejores que los anteriores, con un MAE de 5,2. Los picos siguen sin ajustarse muy bien pero se nota la mejoría respecto al modelo simple.

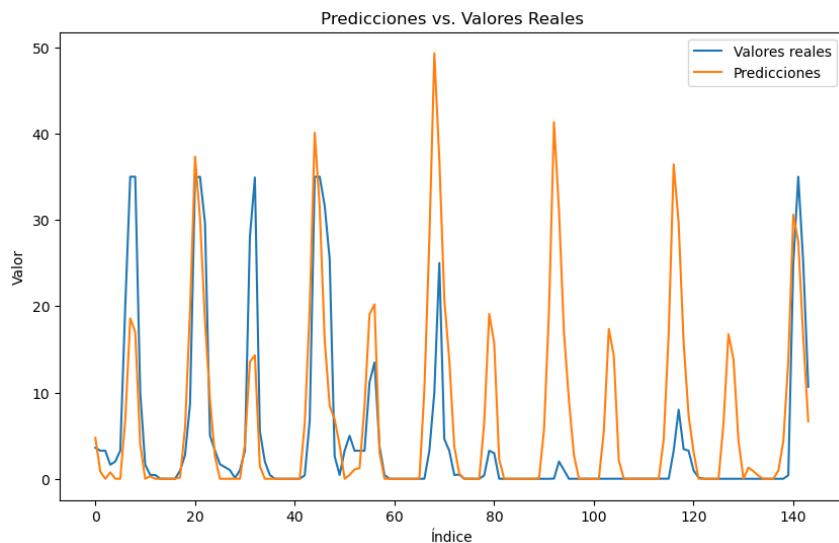


Figura 3.18: Validación vs predicciones CNN multicapa

## Capítulo 3. Desarrollo

---

Tratamos ahora de mejorar los resultados mediante el uso de capas con dropout y regularización. Sin embargo, en esta ocasión, el modelo no se ve beneficiado por el uso de estas. Los resultados son notablemente peores, con un MAE de 7,6 y tanto los valles como los picos no se logran modelar correctamente.

Finalmente se lleva a cabo una implementación de un modelo híbrido que combina capas CNN con capas LSTM.

Este ultimo modelo consta de 3 capas convolucionales (iguales que el modelo multicapa anterior), una capa densa de 64 neuronas seguida por 2 capas LSTM con 128 y 64 neuronas. Finalmente una capa densa de 24 neuronas sera el output.

Como podemos observar en fig.3.19 ahora los valles se modelan correctamente y los picos se ajustan mejor que en otras implementaciones. Obtenemos unas métricas ligeramente mejores que las del modelo multicapa sin dropout, con un MAE de 4.98.

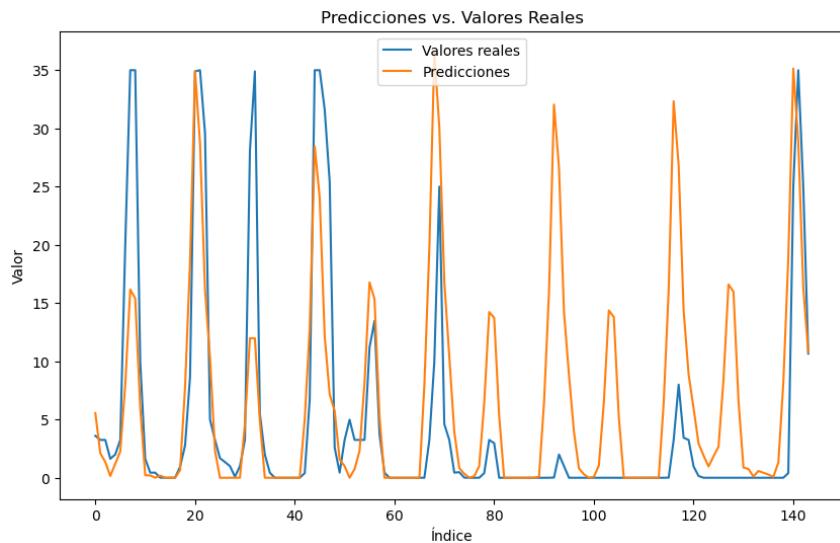


Figura 3.19: Validación vs predicciones CNN + LSTM

Compararemos las gráficas de los 2 mejores modelos con el fin de seleccionar uno de ellos como ganador. Estos modelos serán la CNN multicapa sin dropout ni regularización, y el modelo híbrido que combina capas CNN con LSTM.

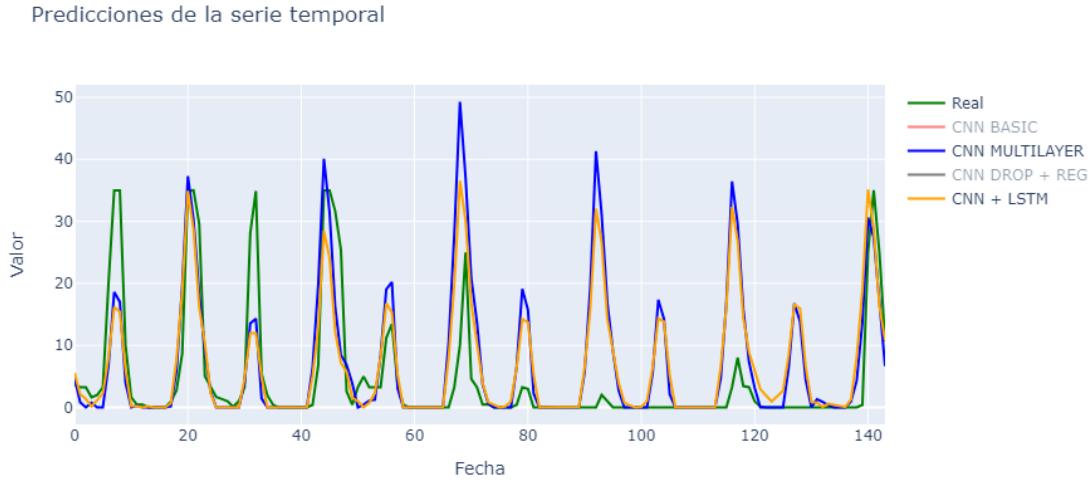


Figura 3.20: Comparativa modelos CNN. Valores reales (verde), CNN multicapa (azul) y CNN+LSTM (amarillo).

Como podemos ver en fig.3.20 ambos modelos predicen de forma muy similar. Puesto que la mayor dificultad se encuentra en hallar la magnitud de los picos y sus MSE es mejor (70 frente a 90) optamos por seleccionar como mejor modelo la red CNN con capas LSTM.

Nuestro modelo CNN ganador (fig. 3.19) en este tramo de 7 días (168 horas), de media, se equivoca tan solo en 4.98€, lo cual es un resultado muy similar al obtenido con el modelo LSTM. En general los picos parecen predecirse correctamente, teniendo mayor dificultad en aquellos que se alejan de la magnitud usual de estos días.

La diferencia entre los resultados obtenidos por los modelos LSTM y CNN+LSTM es pequeña, ambos modelos predicen de forma muy similar. Observando las métricas el modelo LSTM gana al CNN+LSTM tanto en MAE como en MSE, pero bajo unas diferencias casi despreciables.

#### 3.5.4. Temporal Fusion Transformers (TFT)

Por ultimo implementaremos una arquitectura neuronal de estado del arte, los Temporal Fusion Transformers (TFT) (2.5.1).

Antes de comenzar con la explicación del desarrollo del modelo queremos hacer constar que, debido a la poca documentación disponible y ante la falta de un experto en la arquitectura al que poder consultar, no hemos dispuesto del tiempo necesario para llevar a cabo una implementación completa y óptima para nuestro caso de uso. Esto es debido a que, por su naturaleza, la arquitectura es favorecida por el uso de una gran cantidad variables, de las cuales aun no disponemos y llevaría tiempo realizar un buen análisis y preprocesado que beneficie al modelo. Aun así, la implementación del modelo ha sido satisfactoria a

## Capítulo 3. Desarrollo

---

pesar de no llevar a cabo esta labor de inclusión de nuevas variables.

Ya que actualmente la documentación no es muy extensa trataremos de ser lo más claros y explicativos posibles sobre el funcionamiento de las clases e implementaciones usadas. Ahora sí comenzamos con el desarrollo de los modelos Temporal Fusion Transformers.

Implementaremos esta arquitectura mediante la librería PyTorch-Forecasting [22]. Esta implementación sera ligeramente distinta de las anteriores debido a la naturaleza de la arquitectura de los TFT.

En primer lugar crearemos nuestro dataset mediante la clase TimeSeriesDataset de PyTorch-Forecasting. Esta clase facilita mucho ciertos aspectos asociados a las series temporales que manualmente seria costoso de implementar en nuestros modelos, ya sean TFTs u otros. La principal ventaja de usar esta clase es la automatización de tareas comunes como:

- Clasificar variables en 'static', 'known time-varying', etc. Descrito en la sección 2.5.1.
- Codificar variables (texto a categórico por ejemplo).
- Normalizar dinámicamente los valores (por día, semana, mes, año...).
- Convertir eficientemente series temporales en datasets de pandas a tensores de PyTorch.
- Determinar longitudes mínimas y máximas tanto de predicción (multihorizonte) como de la secuencia predictora (longitud del encoder).
- Generar conjuntos de datos de inferencia, validación y prueba.

Entre otras cosas.

Los parámetros más importantes de TimeSeriesDataset, y que afectaran al entrenamiento del modelo son:

- **Grupo:** Este es el parámetro más importante junto con los datos. Será esta columna de nuestro dataframe la que determine los valores que pertenecen a una serie temporal en particular. En nuestro caso probaremos con 2 configuraciones: ventanas temporales de longitud semanal y de longitud mensual.
- **Data:** Este es el parámetro más importante por razones obvias. Por como esta implementado esta vez el conjunto de entrenamiento y validación sera distinto a como los habíamos creado antes. Nuestro dataset completo sera dividido en series temporales según la etiqueta 'group', y posteriormente, de cada serie temporal resultante, se cogerán para entrenamiento todos los valores menos los últimos 'max prediction lenght', que serán usados para validación.

El principal problema es que no podremos dividir los datos en un conjunto de test como hacíamos antes. Se barajaron distintas opciones como usar el 80% para entrenar y validar, y la ultima mitad para evaluar con test,

pero la descartamos puesto que de esta forma estaríamos evaluando las predicciones de 2023 y en test las de 2024, siendo el funcionamiento del mercado distinto.

Ante este problema decidimos evaluar con los datos de validación, sin los de test, y posteriormente para la comparativa entre modelos, hacer predicciones durante dos semanas de abril y manualmente estimar el error y hacer el plot correspondiente de test.

- **Min encoder length:** lo definimos a 24 pues como mínimo queremos realizar las predicciones con los valores del día anterior.
- **Max encoder length:** Probaremos con distintas configuraciones. Generalmente querremos que sea lo más largo posible pero sin alejarse lo suficiente como para que modelice ruido.
- **Max prediction length:** Lo establecemos a 24 puesto que queremos realizar las predicciones de las 24 próximas horas, las del día 'D+1'. Podemos establecer el valor que queramos y así hacer modelos específicos a diferentes horizontes temporales.

Por estos motivos, para poder sacar el mayor provecho de las ventajas que ofrece esta clase creamos nuevas variables categóricas en nuestro dataframe que agrupen bajo un mismo id las filas que pertenezcan al mismo día, a la misma semana y al mismo mes. De esta forma podremos normalizar dinámicamente la serie temporal según los valores pertenecientes a un mismo grupo. La elección de si la normalización se realiza según el día, semana o mes se decidirá de forma empírica en función del desempeño del modelo.

Los hiperparámetros que deberemos ajustar, además de la tasa de aprendizaje, son:

- **Gradient clip:** Define el valor máximo permitido para el gradiente durante el entrenamiento. Esto ayuda a prevenir el problema de gradientes explosivos, mejorando la estabilidad del entrenamiento.
- **Hidden size:** Determina la dimensionalidad de los vectores en las capas ocultas del modelo. Cuanto mayor sea el tamaño oculto, más capaz será el modelo de capturar patrones complejos en los datos, pero también aumentará la carga computacional y el riesgo de sobre ajuste. Puede variar entre 8 y 512.
- **Dropout:** Durante el entrenamiento, se omiten aleatoriamente algunas neuronas de las capas ocultas según la probabilidad fijada.
- **Hidden continuous size:** Define la dimensionalidad de las representaciones ocultas específicas para las variables continuas. Similar al tamaño oculto general, pero se aplica exclusivamente a las variables continuas para capturar mejor sus características.
- **Attention head size:** Determina el número de cabezas de atención en el mecanismo de atención del modelo. Las cabezas de atención permiten que

## Capítulo 3. Desarrollo

---

el modelo enfoque diferentes partes de la secuencia en paralelo, capturando múltiples patrones y relaciones temporales de manera eficiente.

Puesto que es un modelo tan complejo y con tantos parámetros que, a priori, no tenemos intuición de cuales pueden beneficiar más a nuestro modelo optamos por hacer uso de una búsqueda de hiperparámetros. Para ello establecemos los siguientes intervalos

- Gradient clip: [0.01, 1.0].
- Hidden size: [8, 64].
- Hidden continuous size: [8, 64].
- Attention head size: [1, 4].
- Learning rate: [0.001, 0.3].
- Dropout: [0.1, 0.3]

A esta búsqueda de hiperparámetros se suman la elección de los siguientes valores:

- **Establecimiento de subseries temporales:** Consiste en la elección de cómo dividir la serie temporal en secuencias ('group' de TimeSeriesDataset), esto es, a establecer el tamaño de una ventana en la cual consideramos que los valores siguen un mismo comportamiento. En nuestro caso, y al no disponer del tiempo ni recursos para probar con más configuraciones, optamos por agrupar por semana y por mes. Seleccionamos estas puesto que en una agrupación semanal se pueden capturar los patrones y tendencias locales de forma más precisa que en agrupaciones mayores; y la agrupación mensual puesto que favorece una normalización dinámica más estable que la semanal (sin llegar a generalizar demasiado como una agrupación anual) además de permitir contar con un mayor numero de días previos (encoder lenght) para realizar las predicciones. Creemos que una agrupación por estaciones anuales podría ser conveniente, pero nos vemos obligados a dejarlo como trabajo futuro.
- **Fecha a partir de la cual entrenar:** deberemos también seleccionar a partir de cuando usar datos de entrenamiento. En nuestro caso compararemos modelos entrenados con secuencias del ultimo año (últimos 365 días) o lo últimos 700 días (establecemos 700 y no 2 años para que no aprenda relaciones ahora inexistentes consecuentes de la guerra de Ucrania).
- **Nº días previos para realizar predicciones** (encoder lenght): Deberemos seleccionar el numero de días usados para realizar la inferencia. Subseries temporales más grandes permiten mayor numero de días previos, pero no es necesario (ni seguramente conveniente) usar todos. Podemos establecer una subserie mensual y beneficiarnos de una transformación inversa siguiendo la tendencia mensual, pero usando tan solo los 15 últimos días para predecir por ejemplo. En nuestro caso trataremos de tener las secuencias más largas posibles sin llegar a considerar días que se alejen del comportamiento actual de la serie.

Además deberemos seleccionar una métrica de evaluación. La implementación de PyTorch ofrece el uso de Quantile Loss (2.6) por lo que haremos uso de esta para evaluar nuestros modelos TFT. Sin embargo, para comparar los resultados de los TFT con los de otros modelos usaremos el MAE (y el MSE si procede).

A continuación llevaremos a cabo una búsqueda de hiperparámetros individual para modelos con subseries temporales agrupadas semanal y mensualmente. La metodología sera la siguiente:

1. **Búsqueda de hiperparámetros:** Al realizar la comparativa de modelos somos aun más conscientes de los grandes recursos computacionales que requiere el entrenamiento de estos modelos, con ejecuciones de 1 o 2 días de duración y aun así no se llegaban a realizar más de 50 combinaciones diferentes. Tratamos de agilizar el entrenamiento haciendo uso de GPUs, pero al no disponer de ninguna propia tratamos de usar Google Colab, que pone a nuestra disposición sus GPUs, pero al ser también larga la ejecución acababa cerrando nuestra sesión, por lo que abandonamos esta opción. Finalmente, la búsqueda de hiperparámetros se realiza con nuestros propios recursos computacionales, lo que nos impone limitaciones.
2. **Selección de lr:** Pese a que en la búsqueda de parámetros también se prueban diferentes learning rates decidimos hacer uso de Learning Rate Finder de la librería Pytorch-Lighting para hallar la tasa de aprendizaje optima una vez hallada la configuración inicial de hiperparámetros.
3. **Reentrenamiento del modelo:** Reentrenaremos el modelo con los hiperparámetros y la tasa de aprendizaje hallada.

#### Modelos con secuencias semanales

En primer lugar implementamos un TFT con secuencias semanales.

Se llevo a cabo una búsqueda de hiperparámetros sobre un conjunto de train cogiendo los últimos 365 días para train y usando los últimos 3 días para realizar la inferencia. Esta búsqueda de hiperparámetros tardó 24 horas en ejecutar, y se probaron tan solo 50 combinaciones distintas. El resultado fue un modelo sencillo con una loss<sup>1</sup> en validación de 7,55. El modelo presenta la siguiente combinación de hiperparámetros:

- Gradient clip val: 0.47
- Hidden size: 8
- Dropout: 0.1
- Hidden continuous size: 8
- Attention head size: 4
- Learning rate: 0.001 (por lr finder)

---

<sup>1</sup>Recordemos que la loss en este caso es Quantile Loss, no MAE.

## Capítulo 3. Desarrollo

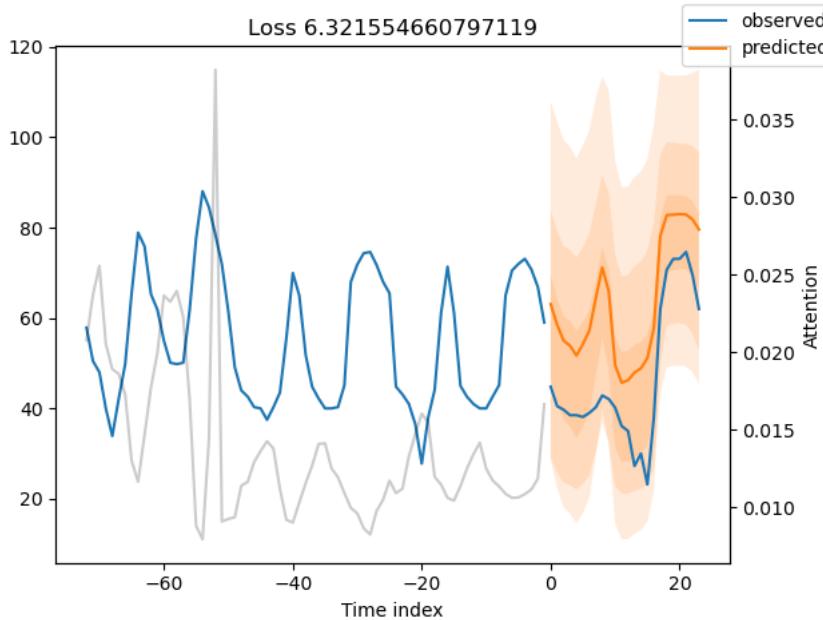


Figura 3.21: Predicción del modelo TFT con subsecuencias semanales.

Este es un modelo sencillo ya que tiene valores de hidden size y attention head size bajos. Esto podría beneficiar a nuestro modelo ya que, en general, los modelos sencillos tienen una mayor capacidad de generalización y además es mucho más ligero.

Se realiza otra búsqueda de hiperparámetros, esta vez cogiendo 700 días y una encoder lenght de 6 días (todos menos el último).

- **Gradient clip val:** 0.01
- **Hidden size:** 36
- **Dropout:** 0.3
- **Hidden continuous size:** 22
- **Attention head size:** 4
- **Learning rate:** 0.001 (por lr finder)

Este es un modelo más complejo que el anterior, lo cual es lógico pues estamos usando más días y secuencias más largas. La loss es de 10,17 y de 10,45 para modelos usando los últimos 3 y 6 días respectivamente.

De entre estas 2 implementaciones nos quedamos con aquella con menor loss, es decir, aquella que usa los últimos 365 días en vez de 700 ya que tiene una loss notablemente más baja (próxima a 7 en vez de a 10).

En la fig. 3.21 mostramos una de sus predicciones para hacernos una idea de los resultados predictivos de este modelo.

### Modelos con secuencias mensuales

Comenzamos ahora con la implementación de modelos con secuencias mensuales. Estos modelos tardan significativamente más en entrenar y realizar las predicciones ya que las secuencias de entrenamiento son más largas. En nuestro caso usaremos los últimos 7 o 15 días para realizar inferencias.

En primera instancia se entrena un modelo con los parámetros obtenidos por la búsqueda de hiperparámetros anterior, y logramos una loss de 3,71, lo cual fue un resultado excelente. Era muy notable la mejoría respecto a los modelos con secuencias semanales, por ello nos enfocamos en una implementación de un TFT con secuencias mensuales.

Se lleva a cabo una búsqueda de hiperparámetros, pero, tras 48 horas de ejecución tan solo se habían comparado 12 modelos distintos, por ello optamos por considerar los siguientes modelos usando:

- **Hiperparámetros previos:** Nuestra primera implementación con los hiperparámetros del anterior modelo dieron unos muy buenos resultados, por ello no descartamos su uso.
- **Mejores hiperparámetros hallados:** Entrenar con los mejores hiperparámetros hallados entre la comparación de 12 modelos distintos. No tienen por qué ser los mejores, tan siquiera tienen que ser buenos debido a las pocas combinaciones probadas.
- **Hiperparámetros manuales:** Seleccionamos intuitivamente hiperparámetros para elaborar un modelo. Este será principalmente un 'hidden size' de longitud 24 (ya que es 24 el periodo de la serie temporal). El resto los fijamos tras haber realizado varios experimentos.

Mediante la búsqueda de hiperparámetros hallamos los siguientes valores, que dan un modelo con loss igual a 4.07:

- **Gradient clip val:** 0.063
- **Hidden size:** 32
- **Dropout:** 0.3
- **Hidden continuous size:** 19
- **Attention head size:** 2
- **Learning rate:** 0.15 (por lr finder)

De estos 3 modelos el que mejores resultados da es el modelo ajustado con hiperparámetros manuales, con el que obtenemos una loss de 3.13.

En la fig. 3.22 mostramos a modo de ejemplo una predicción obtenida por este modelo.

## Capítulo 3. Desarrollo

---

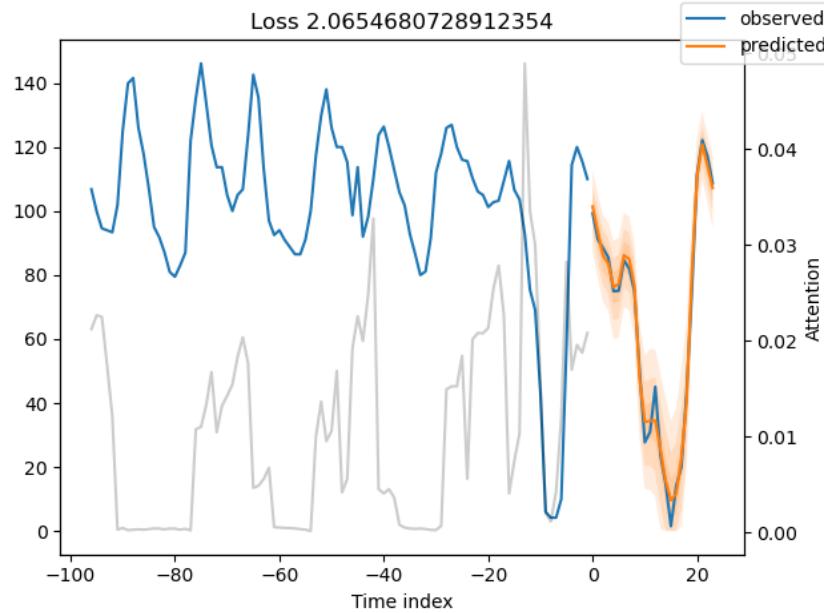


Figura 3.22: Predicción del modelo TFT con subsecuencias semanales

### Comparativa de modelos

Procedemos a comparar los resultados de cada arquitectura TFT. El nombre del modelo sigue la siguiente nomenclatura '*group\_modelDays\_nPrevHours*' (en 'group', 'm' hace referencia a 'month' y 'w' a 'week'). Recordemos tambien que la loss en este caso no es el MAE sino la Quantile Loss (2.6)

Visualizamos los 5 experimentos con mejores resultados. La tabla completa pue-de ser consultada en los anexos (B):

<b>model</b>	<b>loss</b>	<b>epochs</b>	<b>gradient</b>	<b>hidden</b>	<b>dropout</b>	<b>h.cont</b>	<b>att. head</b>	<b>lr</b>
m_113_72	1.618	30	0.01	36	0.30	22	4	0.010
m_133_72	2.063	25	0.01	36	0.30	22	4	0.001
m_365_360	3.133	20	0.03	24	0.25	24	4	0.005
m_365_360	3.347	20	0.03	24	0.25	24	4	0.010
m_365_168	3.640	30	0.31	13	0.29	8	1	0.002
m_365_72	3.703	25	0.01	36	0.30	22	4	0.001
m_365_168	3.740	25	0.01	36	0.30	22	4	0.001
m_200_72	3.748	40	0.03	24	0.25	24	2	0.010
m_365_360	3.846	40	0.03	24	0.25	24	2	0.010
m_365_168	4.072	20	0.06	32	0.30	19	2	0.150

Cuadro 3.1: 10 mejores experimentos TFT.

Como podemos ver entre los 10 mejores modelos no hay ninguno con una ven-tana semanal, todas son mensuales. La primera conclusión que sacamos es que el modelo es favorecido por el uso de ventanas mensuales. Además, también

obtenemos mejores resultados en los modelos que usamos tan solo los últimos 365 días en vez de los últimos 700.

En cuanto al numero de días previos que usamos para realizar las predicciones observamos que no hay grandes diferencias entre coger más o menos días siempre y cuando estos no superen la mitad de la longitud de la ventana. Es decir, el uso de valores muy alejados del presente afectan negativamente al rendimiento del modelo.

Finalmente, y a partir de esta tabla, podemos seleccionar un modelo ganador TFT. Este sera el modelo 'm\_365\_360'. Tomamos como modelo ganador este modelo y no ninguno de los 2 anteriores por el hecho de que los modelos entrenados con tan solo 113 o 133 días tan solo hacen 3 y 4 predicciones respectivamente, lo cual no pueden verse afectados negativamente por predicciones pasadas más complicadas que las nuevas.

Los parámetros e hiperparámetros de nuestro modelo ganador serán:

- **ventana:** mensual
- **nº días previos:** 15
- **epochs:** 20
- **gradient\_clip\_val:** 0.03
- **hidden\_size:** 24
- **dropout:** 0.25
- **hidden\_continuous\_size:** 24
- **attention\_head\_size:** 4
- **learning\_rate:** 0.005

Procedemos a reentrenar el modelo. Las gráficas de las predicciones obtenidas por este modelo pueden ser consultadas en los anexos (C).



## **Capítulo 4**

# **Comparativa y análisis de resultados**

### **4.1. Comparativa de modelos**

En este trabajo hemos estudiado e implementado varios modelos de predicción de series temporales, abarcando desde métodos clásicos como ARIMA (AutoRegressive Integrated Moving Average) hasta avanzados modelos de aprendizaje automático como LSTM (Long Short-Term Memory) y redes neuronales convolucionales (CNN). Además, hemos incluido un modelo de última generación, los Temporal Fusion Transformers (TFT), que representan el estado del arte en esta área.

Tras realizar un análisis exhaustivo y un entrenamiento individual de cada modelo, procedemos a comparar los resultados obtenidos en la tabla 4.1. Esta comparación se basa en predicciones realizadas sobre un conjunto de datos de test que abarca del 9 al 22 de abril de 2024, y tiene como objetivo evaluar tanto cuantitativamente (mediante el MAE) como visualmente (a través de gráficas) las predicciones de cada modelo. De esta manera, buscamos identificar el 'mejor modelo' en términos de precisión y adecuación a los datos.

Con esta comparativa, no solo pretendemos determinar cuál es el modelo más preciso, sino también evaluar los recursos computacionales que cada uno requiere. Esto nos permitirá verificar si los modelos más modernos y complejos implican una carga computacional significativamente mayor y si esta carga adicional se traduce en mejoras sustanciales en las predicciones.

## Capítulo 4. Comparativa y análisis de resultados

Modelo	Error (MAE)	T.train(s)	Pred
ARIMA	10.8	8.67	<p style="text-align: center;"><b>Últimos valores y predicciones</b></p> <p style="text-align: center;">Tiempo</p>
LSTM	4.63	14.3	<p style="text-align: center;"><b>Predicciones vs. Valores Reales</b></p> <p style="text-align: center;">indice</p>
CNN	4.98	23.3	<p style="text-align: center;"><b>Predicciones vs. Valores Reales</b></p> <p style="text-align: center;">indice</p>
TFT	1.23	1697	<p style="text-align: center;"><b>Predicciones vs. Valores Reales TFT</b></p> <p style="text-align: center;">Indice</p>

Cuadro 4.1: Comparación de modelos

#### 4.1. Comparativa de modelos

---

Observando la tabla 4.1 podemos confirmar que los mejores resultados son los obtenidos por el Temporal Fusion Transformer, lo cual no nos sorprende puesto que se trata de un modelo muy complejo de estado del arte.

Lo que sí nos sorprende es la alta precisión con la que los TFT logran realizar las predicciones. Analizando la gráfica, observamos que en numerosas ocasiones los valores reales y predichos se superponen casi perfectamente, lo que demuestra la capacidad del modelo para estimar con gran exactitud los próximos valores del precio SPOT.

Además, el error (MAE) obtenido por el TFT es cuatro veces menor que el error obtenido por los modelos LSTM y CNN, y ocho veces menor que el error del modelo ARIMA. Este error es además bastante bajo: el modelo, de media, se equivoca en tan solo 1,23€ en las predicciones a lo largo de dos semanas. Teniendo en cuenta la volatilidad de la serie y que la desviación típica del precio SPOT para estas dos semanas es de 12,76€, errar en tan solo 1,23€ es un resultado verdaderamente positivo.

El modelo ARIMA, el más antiguo, es el que peores resultados obtiene. Aún así, para ser un modelo sencillo da unos buenos resultados que para ciertas aplicaciones puede ser adecuado.

También podemos destacar que los Temporal Fusion Transformers (TFT) son capaces de manejar series temporales y secuencias más largas de manera efectiva. Los TFT han sido entrenados usando datos del último año completo y con secuencias de 15 días de longitud. En contraste, los modelos ARIMA, LSTM y CNN-1D mostraron un mejor rendimiento cuando se limitaban a considerar períodos más cortos. El modelo ARIMA se benefició de un número reducido de términos autorregresivos, mientras que los modelos LSTM y CNN-1D obtuvieron mejores resultados al considerar solo los últimos tres meses y secuencias diarias. Esto destaca la superioridad de los TFT para modelar largas secuencias de datos.

En cuanto a los tiempos de ejecución también es muy notable la diferencia entre el modelo TFT y el resto, tardando casi media hora en entrenar el modelo TFT frente a menos de medio minuto el resto de modelos. Esto también deberá ser tomado en cuenta a la hora de seleccionar un modelo adecuado para un problema particular. En nuestro caso, este tiempo de entrenamiento no es una limitación, por lo que el TFT sí es adecuado para nuestro trabajo.

Una vez hemos evaluado todos los modelos podemos concluir, sin equivocación alguna, que los Temporal Fusion Transformers dan unos resultados inalcanzables para los otros modelos estudiados. Además, también podemos sacar como conclusión que, para nuestro trabajo, los modelos más novedosos y complejos dan mejores resultados que los más antiguos. Aun así esta conclusión no es generalizable en todos los casos, puesto que puede haber problemas sencillos que se vean afectados negativamente por el uso de modelos complejos.

Aun así debemos tener en cuenta que no hemos podido llevar a cabo una implementación completa añadiendo nuevas variables o ejecutando búsquedas de hiperparámetros más largas, por lo que creemos que se podrían obtener incluso

## **Capítulo 4. Comparativa y análisis de resultados**

---

unos mejores resultados.

### **4.2. Interpretación del modelo TFT**

Una vez hemos concluido y justificado el uso del TFT en nuestro trabajo procedemos a sacar conclusiones sobre las predicciones obtenidas.

En primer lugar, nuestro modelo Temporal Fusion Transformer, en las 2 semanas predicciones que ha hecho en test, tiene un MAE de tan solo 1.23. Esto significa que de todos los valores predichos, nuestro modelo de media se equivoca tan solo en 1.23€ de media. Esto indica que nuestro modelo es bastante fiable, obteniendo predicciones muy precisas.

Dado este bajo margen de error, las predicciones del TFT permiten operar en el mercado con una ventaja estratégica significativa. Al poder anticipar con gran exactitud los valores futuros del precio SPOT, la empresa puede ajustar sus ofertas y estrategias de compra/venta de manera mucho más precisa. Esto se traduce en la capacidad de realizar ofertas que maximicen el margen de ganancia, al ofrecer precios que están muy cerca de los valores reales esperados.

Por otra parte, recordemos que una de la ventaja que ofrece los TFT es que es un modelo explicable, por lo que podemos sacar conclusiones de como se están realizando las predicciones y por que obtenemos esos resultados. Pytorch-Forecasting nos permite:

- Visualizar la atención de cada uno de los valores pasados
- Ranking de importancia de variables: Tanto para variables estáticas, como para encoder y decoder

para poder interpretar el modelo.

Antes de comenzar con las visualizaciones explicaremos que diferencias tiene la importancia de las variables para generar el encoder y el decoder:

- **Encoder:** Las importancias en el encoder indican qué variables históricas y en qué puntos en el tiempo fueron relevantes para modelar el comportamiento de la serie temporal hasta el momento actual.
- **Decoder:** Las importancias en el decoder reflejan qué variables y qué puntos temporales (del pasado) son más relevantes cuando se hacen predicciones.

En primer lugar procedemos a visualizar qué días de los pasados tienen más peso a la hora de realizar las predicciones (fig. 4.1).

Como era de esperar los días más próximos tienen valores de atención más altos. Lo que sí nos sorprende es que no se observa que los valores con más atención sean múltiplos de 24.

Ahora visualizamos cuales son las variables que mayor peso tiene a la hora de generar el encoder (fig. 4.2).

## 4.2. Interpretación del modelo TFT

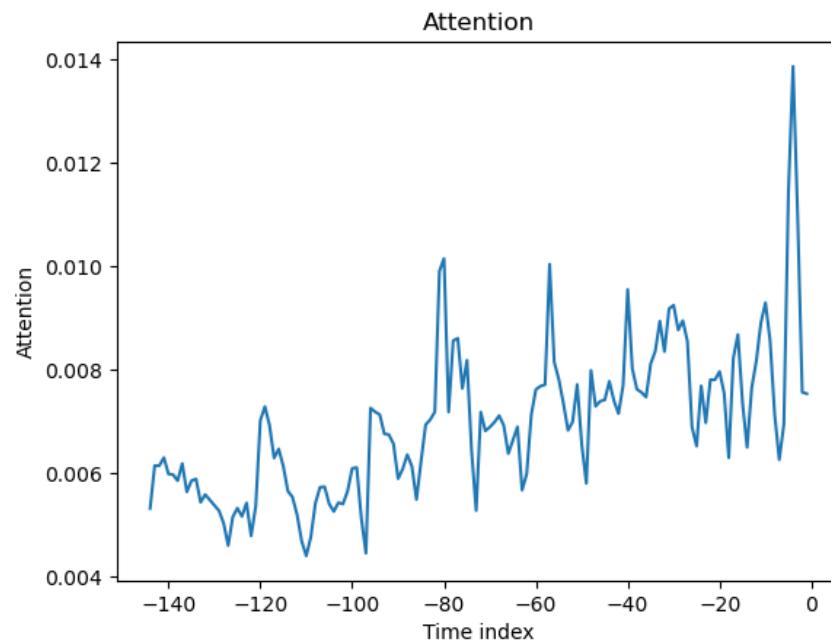


Figura 4.1: Suma de la atención para cada día pasado para realizar las predicciones.

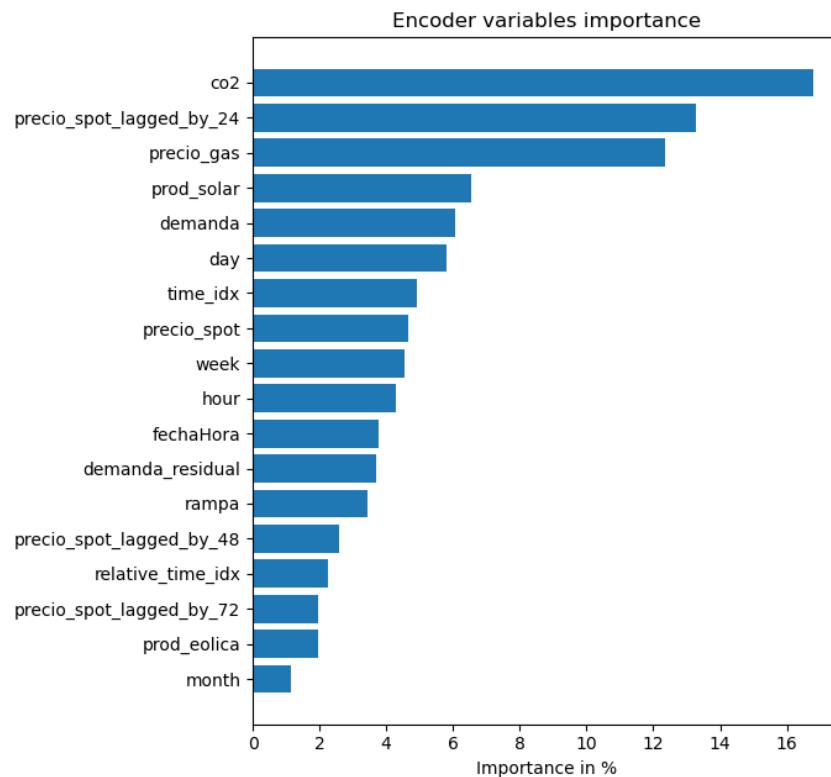


Figura 4.2: Ranking de importancia de variables en el encoder.

## Capítulo 4. Comparativa y análisis de resultados

---

La variable con más peso fue el CO2, lo cual puede tener sentido ya que vimos en la matriz de correlación (fig.3.7) que el precio SPOT y el CO2 estaban correlacionados. La siguiente variable con más importancia es el precio SPOT del día anterior, lo cual es lógico, pues como bien sabemos nuestra serie temporal tiene estacionalidad diaria. La tercera variable con más peso es el precio del gas, lo cual tiene sentido al estar también correlacionada. Aun así nos sorprende que la variable más correlacionada, la demanda residual, este en la parte baja del ranking.

Finalmente visualizamos en la fig. 4.3 cuáles son las variables que mayor peso tienen en el decoder.

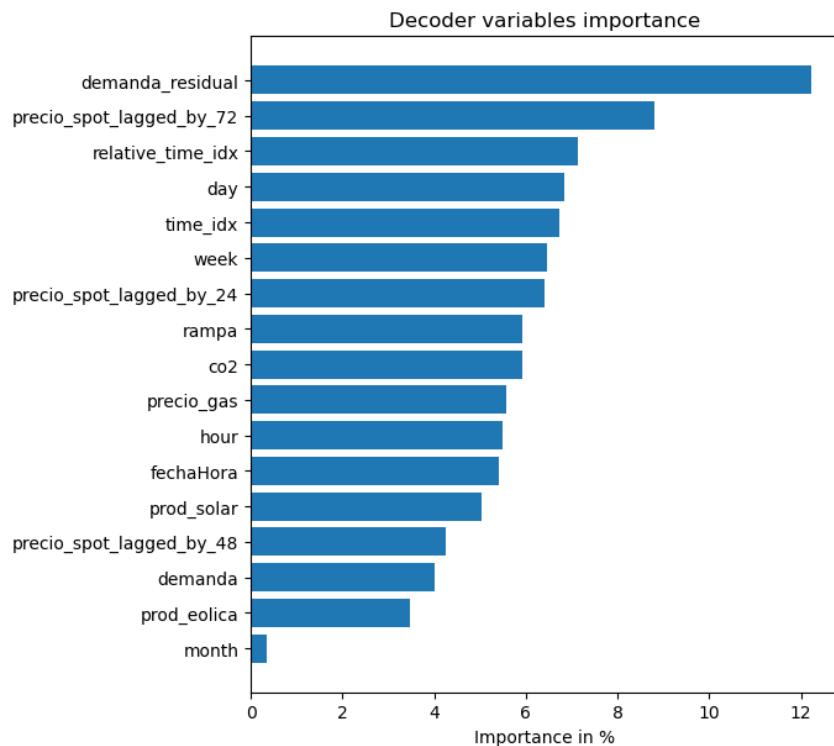


Figura 4.3: Ranking de importancia de variables en el decoder.

Ahora sí aparece la demanda residual encabezando la lista, la variable con mayor correlación con nuestra variable objetivo. Curiosamente ahora no es el precio SPOT del día anterior, sino el de 3 días antes (72h) la segunda variable con más peso para generar las predicciones. En tercer lugar encontramos el índice que indica el orden de la secuencia temporal, seria equivalente al día u hora, realmente a cualquier valor que sirva para ordenar la serie temporal, simplemente indica que el orden de la secuencia es importante.

Finalmente visualizamos en la fig. 4.4 las variables estáticas por orden de importancia.

Estas son las variables que no dependen del tiempo. Como vemos la que mayor influencia tiene es la media del precio SPOT durante ese mes (no la media de la

## 4.2. Interpretación del modelo TFT

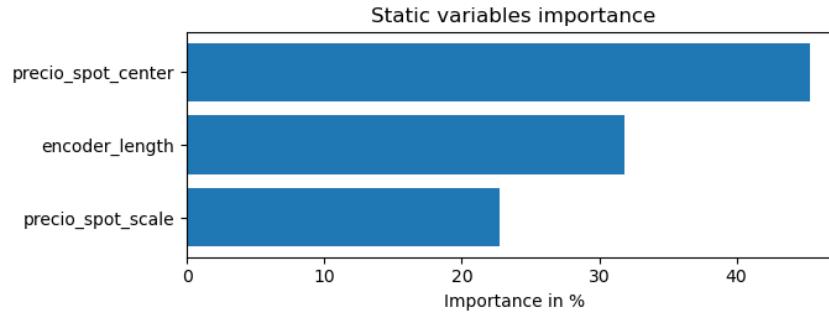


Figura 4.4: Ranking de importancia de variables estáticas

secuencia). Nos sorprende también que sea la longitud del encoder la que preceda a la varianza del precio SPOT en ese mes. En un principio no le encontramos el sentido pero tras analizarlo puede tenerlo ya que puede haber meses incompletos en nuestro dataset y es posible que en algún caso deba usar secuencia de menor longitud (la secuencia mínima la establecimos a 24h), pero aun así nos hubiese parecido más lógico que la varianza tuviese más peso.

Teniendo todo esto en cuenta podemos sacar las siguientes conclusiones:

- La alta importancia del CO2 en el encoder sugiere que el modelo reconoce la correlación significativa entre el CO2 y el precio SPOT.
- La relevancia del precio SPOT del día anterior en el encoder confirma la estacionalidad diaria de la serie temporal, confirmando que el comportamiento reciente del precio es fundamental para modelar los presentes.
- La sorprendentemente baja importancia de la demanda residual en el encoder, a pesar de su alta correlación con el precio SPOT, puede indicar que su influencia aplica tan solo en el momento de realizar la predicción, lo que se refleja en su alta importancia en el decoder.
- La relevancia del precio SPOT del día anterior en el encoder resalta la estacionalidad diaria de la serie temporal, confirmando que el comportamiento reciente del precio es fundamental para modelar los presentes.
- La influencia de la media mensual del precio SPOT como la variable estática más importante implica que las tendencias mensuales proporcionan información valiosa para el modelo, ayudándolo a capturar patrones a largo plazo.
- La presencia del precio SPOT de tres días antes en el decoder destaca el peso que tienen los valores pasados para realizar predicciones precisas.



## **Capítulo 5**

# **Análisis de impacto**

Nuestro trabajo puede tener impacto sobre todo en un contexto empresarial, concretamente a empresas participantes en el mercado eléctrico diario como pueden ser generadoras.

Estas se pueden beneficiar de la implementación de nuestro modelo Temporal Fusion Transformer de la siguiente forma.

Como comentamos en la introducción, por como se realiza la formación de precios en el mercado energético español, las generadoras generalmente ofertan más barato de lo deseado para evitar ofertar por encima del precio de casación y quedar fuera del mercado ese día. Por ello, con nuestro sistema, las generadoras tendrían una intuición muy precisa del precio de casación del día siguiente, por lo que podrían ofertar a este precio (o uno muy cercano a este) maximizando los beneficios al vender al mayor precio posible. Además, la capacidad de explicabilidad del modelo TFT ofrece un valor añadido significativo, permitiendo a los expertos del dominio ajustar y optimizar aún más las predicciones, incrementando la fiabilidad y eficacia del sistema.

En los que a economía respecta, existe la posibilidad de que el comportamiento del propio mercado en el que se intenta predecir sea influenciado por el uso de este sistema de predicción.

Esto sucedería si una gran parte de los participantes del mercado usasen este modelo para operar en él, ya que acabarían desplazando a la izquierda (a un valor más bajo) el punto de corte entre las curvas de oferta y demanda. Esto sucedería debido a que habría muchas ofertas más bajas de las que habría en un escenario sin el modelo, por tanto el precio tendería a abaratarse.

Aun así, esta es una posibilidad muy improbable ya que requeriría que todos (o casi todos) los participantes del mercado operasen según nuestro modelo.

Finalmente, este trabajo podría ser de utilidad personal para aquellos individuos que generen energía en su propiedad, como pueden ser los paneles solares. Existen tarifas en las que como particular puedes vender la energía sobrante no consumida a una comercializadora. En este caso, y si el precio de venta es

## **Capítulo 5. Análisis de impacto**

---

igual al precio de casación del mercado diario, se podría optimizar el momento de venta de la energía para hacerlo en su precio máximo.

# **Capítulo 6**

## **Conclusiones y trabajo futuro**

### **6.1. Conclusiones**

El principal objetivo de este trabajo fue desarrollar un modelo capaz de predecir el precio casado del MWh (precio SPOT del mercado energético) para cada hora del día siguiente, con el fin de optimizar la oferta en el mercado. Tras la implementación y evaluación de diversos modelos, podemos concluir que hemos cumplido este objetivo de manera efectiva.

Nuestro modelo, el Temporal Fusion Transformer (TFT), ha demostrado una precisión notable, con un error medio (MAE) de solo 1.23€. Este nivel de precisión permite maximizar los beneficios y minimizar los riesgos, ya que proporciona estimaciones altamente acertadas del precio futuro. Además, la capacidad de explicabilidad del modelo TFT ofrece un valor añadido significativo, permitiendo a los expertos del dominio ajustar y optimizar aún más las predicciones, incrementando la confiabilidad y eficacia del sistema.

Un segundo objetivo era evaluar si la mayor complejidad del modelo TFT justificaba su uso comparado con modelos más simples como ARIMA. A pesar de que el TFT requiere un mayor esfuerzo computacional y una curva de aprendizaje más empinada, los resultados obtenidos confirman que su uso está justificado en contextos donde la precisión es crítica. Las predicciones del TFT fueron considerablemente mejores que las del resto de modelos (fig. 4.1).

Aun así, cabe mencionar que en otros escenarios un modelo CNN+LSTM o incluso un modelo ARIMA podría ser adecuado si no se requiere tanta precisión en las predicciones (por ejemplo, solo para saber si el próximo valor subirá o bajará) o si no se disponen de grandes recursos computacionales. Estos modelos son mucho más sencillos de aprender, implementar y entrenar, y además son más ligeros.

La interpretabilidad del TFT ha proporcionado una comprensión profunda de las variables que influyen en la formación del precio SPOT, revelando varias conclusiones interesantes. La alta importancia del CO<sub>2</sub> en el encoder confirma no solo su correlación significativa con el precio SPOT, sino que también sugie-

## **Capítulo 6. Conclusiones y trabajo futuro**

---

re un vínculo subyacente entre las emisiones y las dinámicas de precio en el mercado energético. La relevancia del precio SPOT del día anterior en el encoder subraya la estacionalidad diaria, indicando que el modelo capta eficientemente patrones repetitivos de corto plazo. La baja importancia de la demanda residual en el encoder, contrastada con su alta importancia en el decoder, sugiere que esta variable tiene un impacto más directo y puntual en la predicción precisa, reflejando cambios repentinos en la oferta y demanda del mercado. Además, la media mensual del precio SPOT como variable estática más influyente implica que las tendencias a largo plazo son cruciales para entender las fluctuaciones del mercado, ayudando a estabilizar las predicciones a lo largo del tiempo. La consideración del precio SPOT de tres días antes en el decoder resalta la capacidad del modelo para integrar información histórica específica, esencial para capturar las dinámicas complejas del mercado y mejorar la precisión de las predicciones.

Finalmente, concluimos que los objetivos del trabajo se han cumplido con éxito, obteniendo resultados muy satisfactorios. La implementación del TFT ha demostrado ser una herramienta poderosa para predecir el precio SPOT, proporcionando a la empresa una ventaja estratégica significativa en el mercado energético.

En resumen, la utilización del TFT no solo optimiza las operaciones actuales, sino que también abre nuevas oportunidades estratégicas basadas en análisis avanzados de datos. Esto consolida la posición de la empresa en el mercado energético y asegura un futuro más rentable y eficiente.

### **6.2. Trabajo Futuro**

Aun habiendo logrado unos buenos resultados, como hemos comentado al evaluar los modelos TFT, existen líneas de mejora con las que creemos que podríamos superar los resultados obtenidos. Estas líneas de continuación son:

- Introducción de nuevas variables: Tanto reales (telemedidas, intercambios de energía entre países, índices como IBEX365, días festivos, etc.) como transformaciones de las mismas (ratios de producción de energías renovables frente a producción total, logaritmos, lags, diferencias, etc.). Esto creemos que beneficiaría al modelo puesto que los TFT son modelos grandes que incorporan un proceso de selección de variables para seleccionar para cada predicción las que mejor ajustan el modelo.
- Uso de GPUs/TPUs para búsqueda de hiperparámetros: Como comentamos en la sección 3.5.4 nos vimos limitados por nuestros recursos computacionales para realizar una búsqueda de hiperparámetros más extensa. Creemos que mediante el uso de GPUs o TPUs se agilizaría el proceso drásticamente logrando obtener modelo con mejores resultados.
- En general se podría estudiar mejor la implementación existente de los TFT de Pytorch, ya que esta no está aún muy bien documentada por su reciente elaboración y ha requerido un gran esfuerzo y tiempo lograr implementarlo en este trabajo. De disponer más tiempo posiblemente se lograse identificar

## **6.2. Trabajo Futuro**

---

alguna mejora para el modelo.

- Ya que los TFT lo permiten, estudiar la posibilidad de realizar predicciones a mayores horizontes temporales ('D+n')
- Aplicación en otros mercados: Evaluar la aplicabilidad del TFT en diferentes mercados energéticos y otros contextos donde la predicción precisa de series temporales sea crucial.
- Desarrollar interfaces y sistemas integrados que utilicen las predicciones del TFT para la toma de decisiones automatizada en tiempo real.

Estas líneas de trabajo futuro permitirán no solo mantener la relevancia del modelo desarrollado, sino también mejorar continuamente su desempeño y adaptabilidad a nuevos desafíos y oportunidades en el mercado energético.



# Bibliografía

- [1] *6.1. Formación de precios en el mercado mayorista diario de electricidad - Energía y Sociedad* — energiaysociedad.es, <https://www.energiaysociedad.es/manual-de-la-energia/6-1-formacion-de-precios-en-el-mercado-mayorista-diario-de-electricidad/>, [Accessed 21-04-2024].
- [2] *3.1. Tecnologías y costes de la generación eléctrica* - Energía y Sociedad — energiaysociedad.es, <https://www.energiaysociedad.es/manual-de-la-energia/3-1-tecnologias-y-costes-de-la-generacion-electrica/>, [Accessed 21-04-2024].
- [3] S. Hochreiter y J. Schmidhuber, «Long Short-Term Memory», *Neural Computation*, vol. 9, n.º 8, págs. 1735-1780, nov. de 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: <https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf>. dirección: <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [4] V. L. Tran, T. C. Vo y T. Q. Nguyen, «One-dimensional convolutional neural network for damage detection of structures using time series data», *Asian Journal of Civil Engineering*, vol. 25, págs. 827-860, 2024. DOI: 10.1007/s42107-023-00816-w. dirección: <https://doi.org/10.1007/s42107-023-00816-w>.
- [5] B. Lim, S. Ö. Arik, N. Loeff y T. Pfister, «Temporal Fusion Transformers for interpretable multi-horizon time series forecasting», *International Journal of Forecasting*, vol. 37, n.º 4, págs. 1748-1764, 2021, ISSN: 0169-2070. DOI: <https://doi.org/10.1016/j.ijforecast.2021.03.012>. dirección: <https://www.sciencedirect.com/science/article/pii/S0169207021000637>.
- [6] A. Pal y P. Prakash, *Practical Time Series Analysis*. Packt Publishing, 2017, ISBN: 9781788290227. dirección: <https://books.google.es/books?id=NkFWtAEACAAJ>.
- [7] A. Haber, *10. Decomposition of Time-Series in Python: Trend and Seasonal (Periodic) Components 013; Fusion of Engineering, Control, Coding, Machine Learning, and Science*. aleksandarhaber.com, <https://aleksandarhaber.com/decomposition-of-time-series-in-python-trend-and-seasonal-periodic-components/>, [Accessed 08-05-2024].

## BIBLIOGRAFÍA

---

- [8] F. Rosenblatt, «The perceptron: a probabilistic model for information storage and organization in the brain.», *Psychological review*, vol. 65 6, págs. 386-408, 1958. dirección: <https://api.semanticscholar.org/CorpusID:12781225>.
- [9] Wikipedia, *Perceptrón — Wikipedia, La enciclopedia libre*, [Internet; descargado 14-noviembre-2023], 2023. dirección: <https://es.wikipedia.org/w/index.php?title=Perceptr%C3%B3n&oldid=155349909>.
- [10] R. M. Schmidt, «Recurrent neural networks (rnns): A gentle introduction and overview», *arXiv preprint arXiv:1912.05911*, 2019.
- [11] K. Mani, *GRU's and LSTM's — towardsdatascience.com*, <https://towardsdatascience.com/grus-and-lstm-s-741709a9b9b1>, [Accessed 27-05-2024].
- [12] K. O'shea y R. Nash, «An introduction to convolutional neural networks», *arXiv preprint arXiv:1511.08458*, 2015.
- [13] P. Raghav, *Understanding of Convolutional Neural Network (CNN) — Deep Learning — RaghavPrabhu*, <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>, [Accessed 27-05-2024].
- [14] A. Vaswani, N. Shazeer, N. Parmar et al., «Attention Is All You Need», *arXiv e-prints*, arXiv:1706.03762, arXiv:1706.03762, jun. de 2017. doi: 10.48550/arXiv.1706.03762. arXiv: 1706.03762 [cs.CL].
- [15] J. Nyandwi, «The Transformer Blueprint: A Holistic Guide to the Transformer Neural Network Architecture», *Deep Learning Revision*, 29 de jul. de 2023. dirección: <https://deepprevious.github.io/posts/001-transformer/>.
- [16] U. C. Galdeano y M. L. Basterra, «Determinantes del precio de la electricidad en España», *Estadística Española*, vol. 59, n.º 194, págs. 119-149, 2017.
- [17] M. Salvador Andaluz, «Análisis del mercado eléctrico en España: costes de generación y repercusiones en el precio de la electricidad», Tesis doct., UPC, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Departament de Física i Enginyeria Nuclear, nov. de 2010. dirección: <http://hdl.handle.net/2099.1/10666>.
- [18] MIBGAS, *Mercado Ibérico del Gas*, Accedido el 16 de Mayo de 2024. dirección: <https://www.mibgas.es/es>.
- [19] ICE, *Intercontinental Exchange*, Accedido el 16 de Mayo de 2024. dirección: <https://www.ice.com/index>.
- [20] ESIOS Electricidad, <https://www.esios.ree.es/es>, Accedido el 16 de Mayo de 2024.
- [21] G. Box y G. Jenkins, *Time Series Analysis: Forecasting and Control* (Holden-Day series in time series analysis and digital processing). Holden-Day, 1970, ISBN: 9780816210947. dirección: <https://books.google.es/books?id=5BVfnXaq03oC>.
- [22] J. Beitner, *PyTorch Forecasting: Time series forecasting with PyTorch*, 2020. dirección: <https://github.com/jdb78/pytorch-forecasting>.

## **Anexos**



## **Apéndice A**

### **Primer anexo**

# PREDICCION DEL PRECIO SPOT CON MODELOS ARIMA

Nicolas Vega Munoz

2024-04-10

El análisis de series temporales es una técnica fundamental en el campo de la estadística y la ciencia de datos, especialmente en situaciones donde los datos están correlacionados en el tiempo. En este documento, exploraremos el modelado de series temporales utilizando el método ARIMA (Autoregressive Integrated Moving Average).

El objetivo principal de este análisis es desarrollar un modelo predictivo robusto que pueda capturar y predecir patrones en los datos de una serie temporal específica. Utilizaremos datos históricos de precios SPOT, los cuales están sujetos a fluctuaciones estacionales y estocásticas, lo que los hace adecuados para el modelado con ARIMA.

A lo largo de este informe, abordaremos los siguientes pasos:

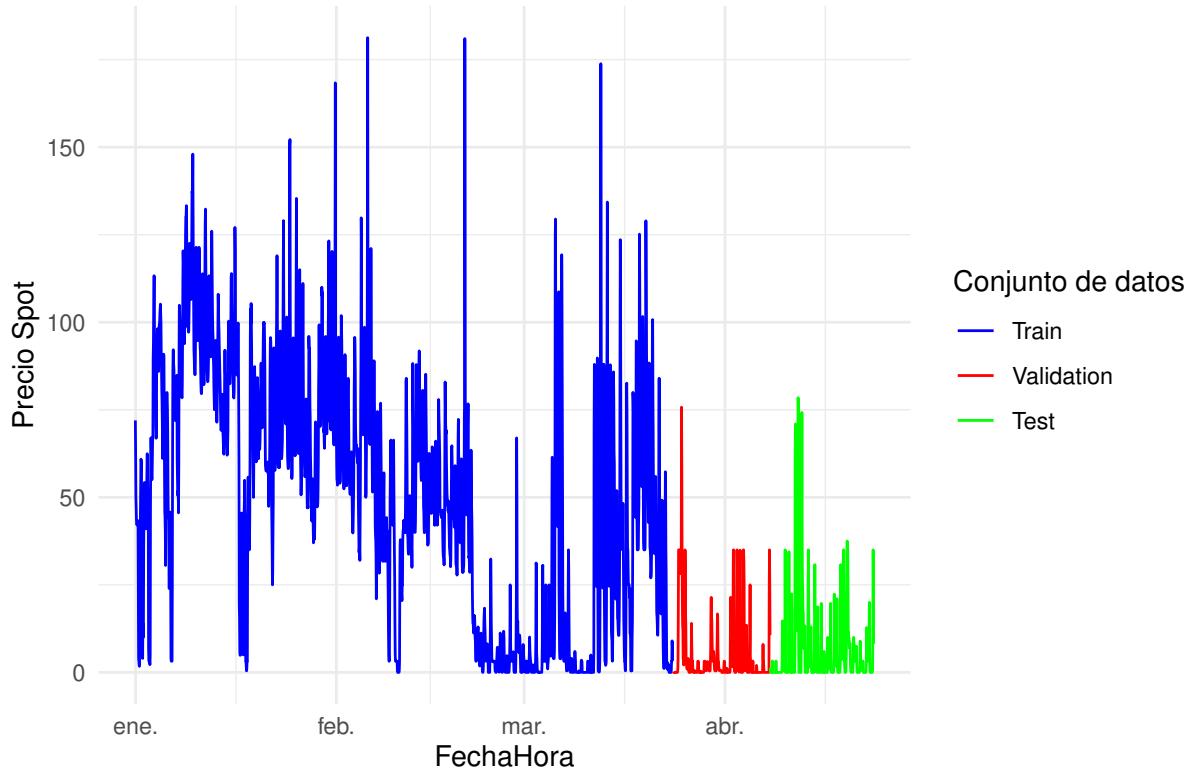
- 1. Preparación de datos:** Cargaremos y preprocesaremos los datos de la serie temporal, asegurándonos de que estén en un formato adecuado para el modelado.
- 2. Diagnóstico de estacionalidad y tendencia:** Analizaremos la serie temporal para identificar componentes de tendencia y estacionalidad, lo que nos permitirá seleccionar los parámetros apropiados para el modelo ARIMA.
- 3. Ajuste de modelos ARIMA:** Ajustaremos varios modelos ARIMA a los datos, utilizando diferentes configuraciones de parámetros para encontrar el modelo que mejor se ajuste a la serie temporal.
- 4. Evaluación del modelo:** Evaluaremos el desempeño de los modelos ARIMA utilizando métricas de evaluación adecuadas, como el error cuadrático medio (MSE) y el error absoluto medio (MAE).
- 5. Selección del mejor modelo:** Basándonos en las métricas de evaluación y en el diagnóstico de los residuos, seleccionaremos el modelo ARIMA más adecuado para realizar predicciones futuras.

A lo largo de este proceso, nos centraremos en aplicar los conceptos teóricos detrás del modelo ARIMA, así como en utilizar herramientas prácticas de programación en R para implementar y evaluar los modelos. Este análisis nos proporcionará información valiosa sobre la dinámica subyacente de la serie temporal estudiada y nos permitirá realizar predicciones precisas.

En primer lugar leemos nuestro dataset y configuramos el numero de días que usaremos para evaluar el modelo. Puesto que disponemos de un histórico grande con datos muy alejados de los valores actuales decidimos entrenar el modelo con datos a partir de este año 2024.

Procedemos a dividir el dataset en train, validation y test. Usamos 15 dias para validacion y 15 para test

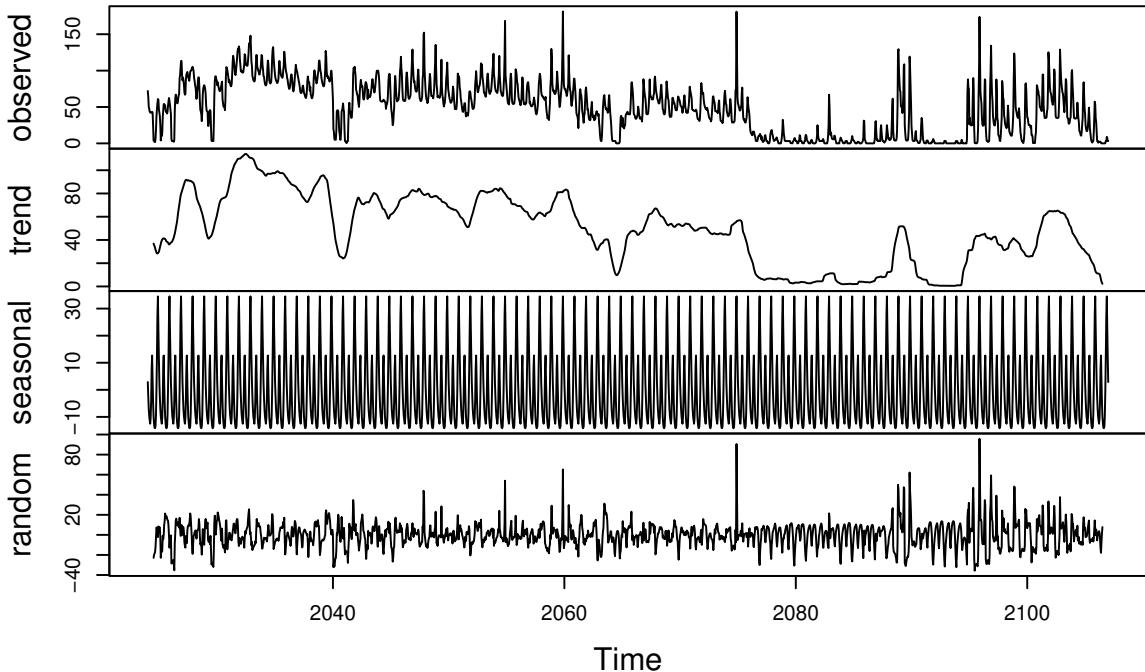
## Precio\_spot de Train, Validation y Test



Como describimos en el apartado de Fundamentos Teóricos el modelo ARIMA tiene como supuesto que nuestra serie temporal es estacionaria en media en varianza, i.e, que la media y la varianza son estables en el tiempo.

Por ello visualizaremos la serie temporal, descomponiéndola y realizaremos los contrastes de hipótesis oportunos para determinar si nos encontramos ante una serie temporal estacionaria y si se presenta una componente estacional.

## Decomposition of additive time series



Al descomponer la serie temporal observamos que esta no es ni estacionaria en media ni en varianza. Por ello deberemos realizar las transformaciones oportunas para disponer de una serie temporal estacionaria adecuada para el modelo ARIMA.

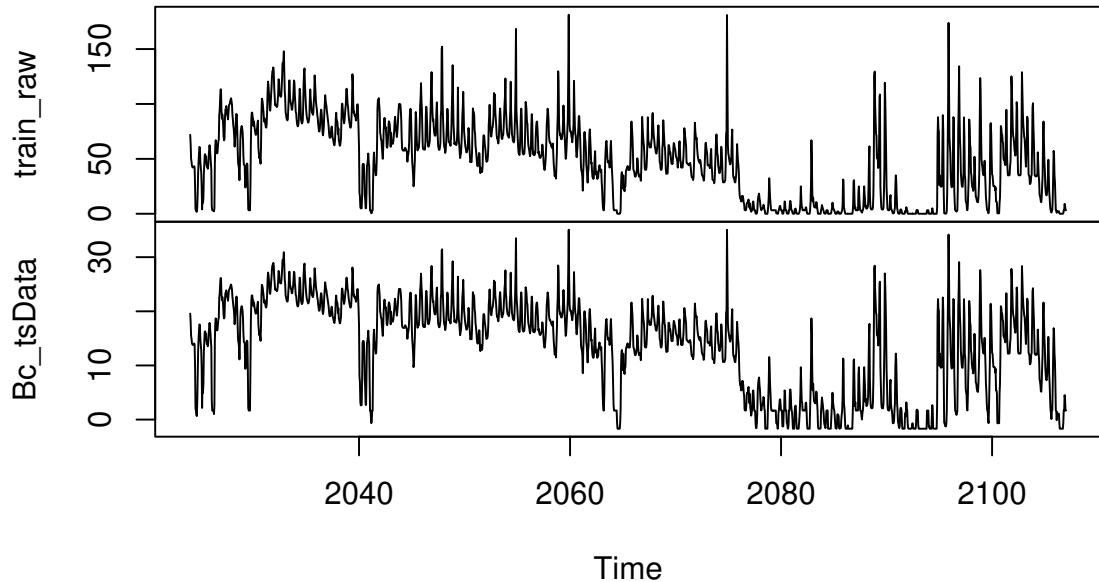
Parece que si existe una componente estacional, pero es difícil de determinar a partir de este gráfico por lo agrupados que están los datos. Evaluaremos la existencia de componente estacional posteriormente mediante las ACF y PACF.

EL modelo ARIMA tiene como supuesto tratar con una serie estacionaria en media y en varianza. Puesto que nuestra serie temporal no lo es, debemos realizar las transformaciones necesarias.

Para hacer la serie temporal estacionaria en varianza realizaremos una transformación de Box-Cox. En primer lugar calculamos el valor de lambda.

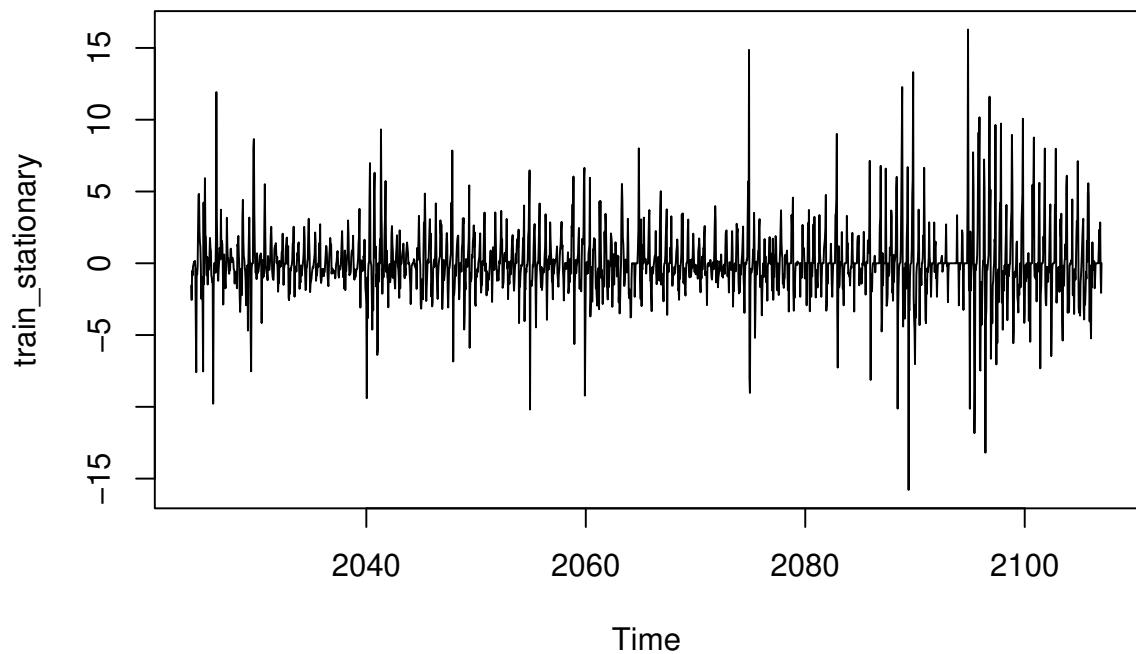
```
## [1] 0.5925395
```

**cbind(train\_raw, Bc\_tsData)**



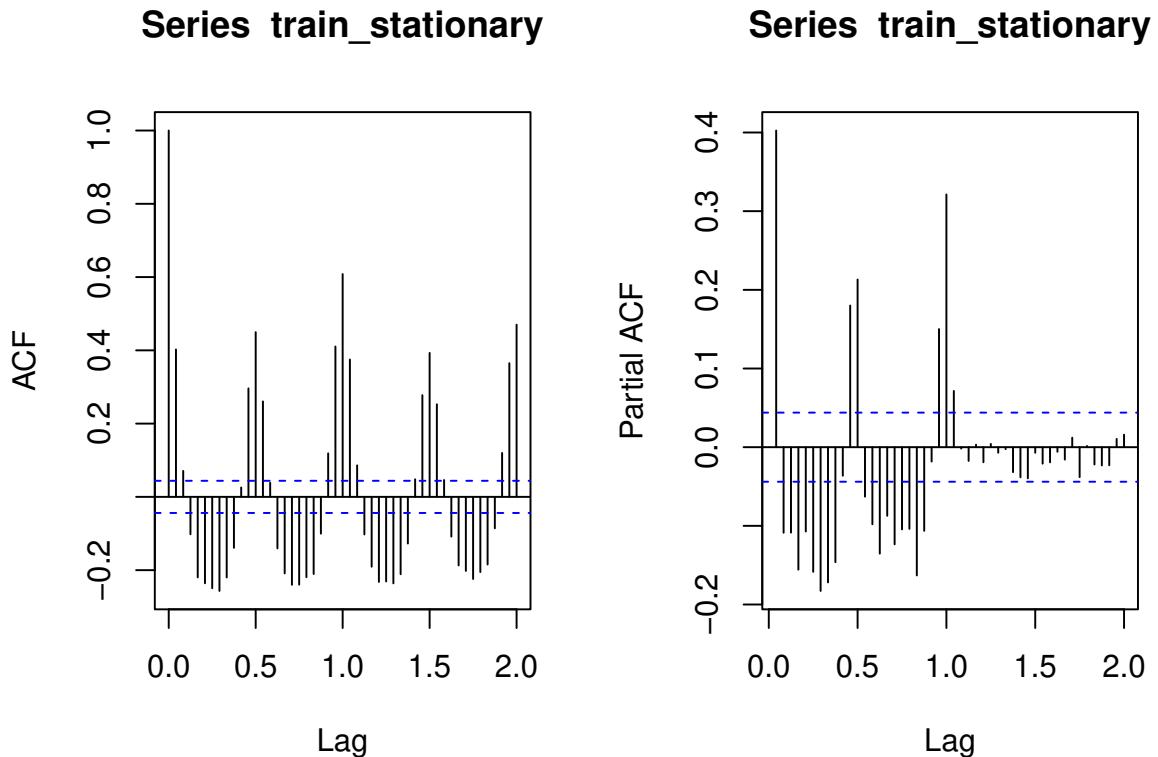
Observamos como ahora la serie es estacionaria en varianza. Ahora deberemos hacerla estacionaria en media, por lo que diferenciamos las veces necesarias la serie. En nuestro caso tan solo sera necesario diferenciar una vez. El parámetro 'd' de nuestro modelo SARIMA sera igual a 1.

```
## [1] 1
```



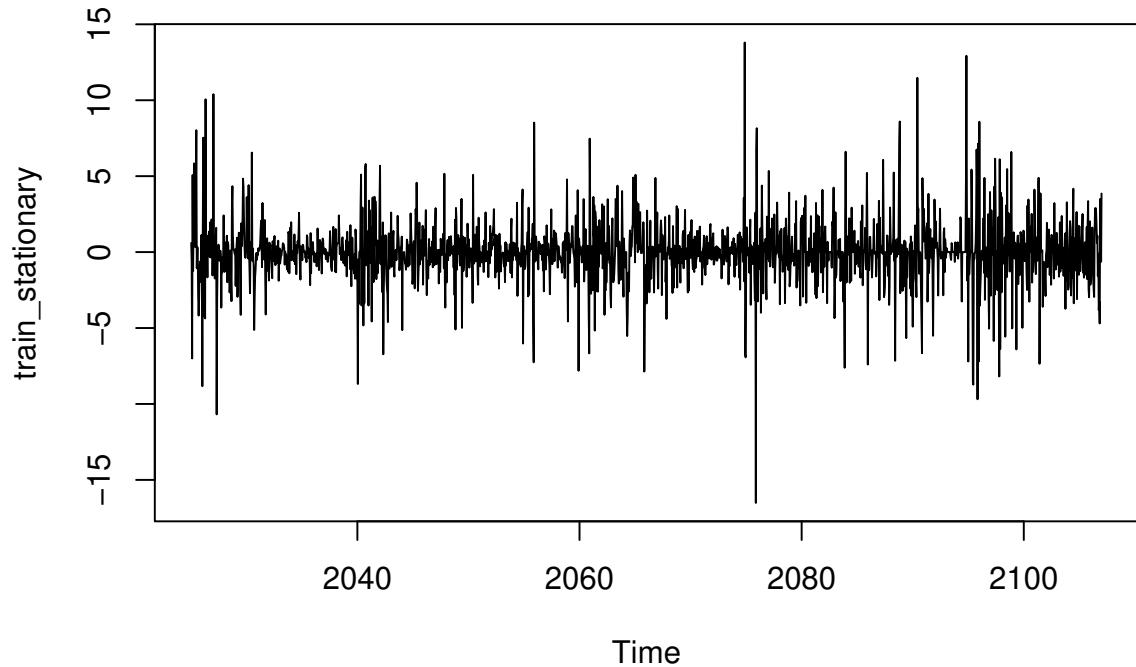
Como podemos observar la serie ahora es estacionaria en media y varianza.

Procedemos ahora a estudiar la estacionalidad.



Como podemos ver en la PACF la serie temporal tiene el valor mas significativo en el retardo (lag) 24 (valor de la frecuencia), por lo tanto la serie tiene componente estacional diaria.

Realizamos una diferenciación estacional. El parámetro ‘D’ de ARIMA sera igual a 1.

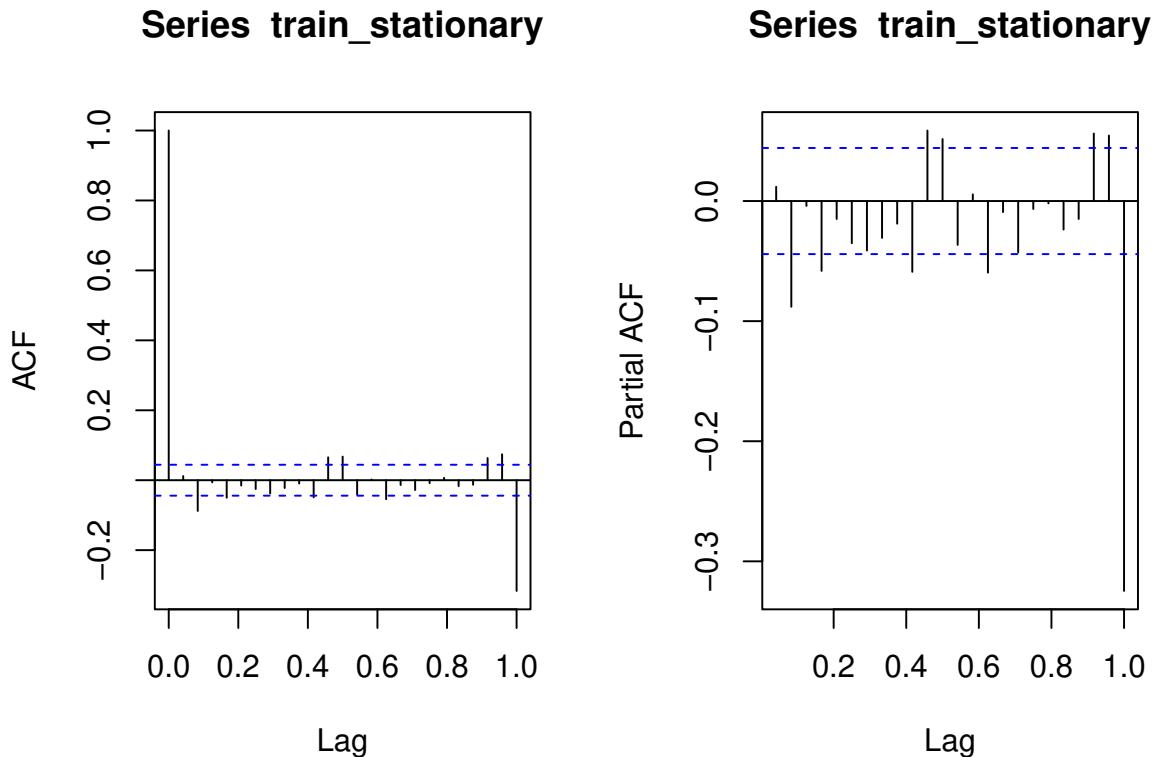


En principio ya hemos transformado nuestra serie a una estacionaria en media y varianza y sin componente estacional. Aun así, realizamos el test de Dickie-Fuller aumentado para estar seguros de que no debemos realizar mas transformaciones. La hipótesis nula es que nuestra serie no es estacionaria.

```
##  
##  Augmented Dickey-Fuller Test  
##  
## data:  train_stationary  
## Dickey-Fuller = -13.489, Lag order = 12, p-value = 0.01  
## alternative hypothesis: stationary
```

Como el p-valor es menor que 0.05 podemos rechazar la hipótesis nula de que la serie no es estacionaria, por tanto podemos concluir que nuestra serie es estacionaria.

Una vez tenemos nuestra serie temporal estacionaria podemos comenzar con la modelización. En primer lugar graficaremos las ACF y PACF para hallar los parámetros del modelo ARIMA(p,d,q)x(P,D,Q)s.



Observamos en estas gráficas como no hay ningún término significativo (ni autorregresivo ni de medias móviles) consecutivo, por lo que los valores  $p$  y  $q$  serán igual a 0. Los parámetros ' $d$ ' y ' $D$ ' tendrán valor 1 por las diferenciaciones realizadas a la serie temporal y ' $s$ ' sera 24 por su componente estacional.

Tras haber realizado este estudio estadístico podemos concluir que nuestro modelo SARIMA sera de la forma  $(0,1,0)x(0,1,0)24$

Una vez disponemos de los parámetros de nuestro modelo SARIMA entrenamos el modelo con nuestros datos de train.

```
## Series: train_stationary
## Regression with ARIMA(0,1,0)(0,1,0)[24] errors
##
## Coefficients:
##             demanda      EUA  precio_gas  prod_eolica  prod_solar demanda_residual
##             3e-04   -0.1127     0.3175      -3e-04     -3e-04           0e+00
## s.e.       1e-04    0.1355     0.1871       1e-04      1e-04           1e-04
##             rampa
##             1e-04
## s.e.       1e-04
##
## sigma^2 = 23: log likelihood = -5799.51
## AIC=11615.02  AICc=11615.09  BIC=11659.6
##
## Training set error measures:
##                  ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set 0.006341458 4.756403 3.097877 NaN Inf 1.418735 -0.4817001
```

A priori no parece que tengamos un mal modelo, en general las métricas tienen valores bajos y un AIC=11631.238578.26

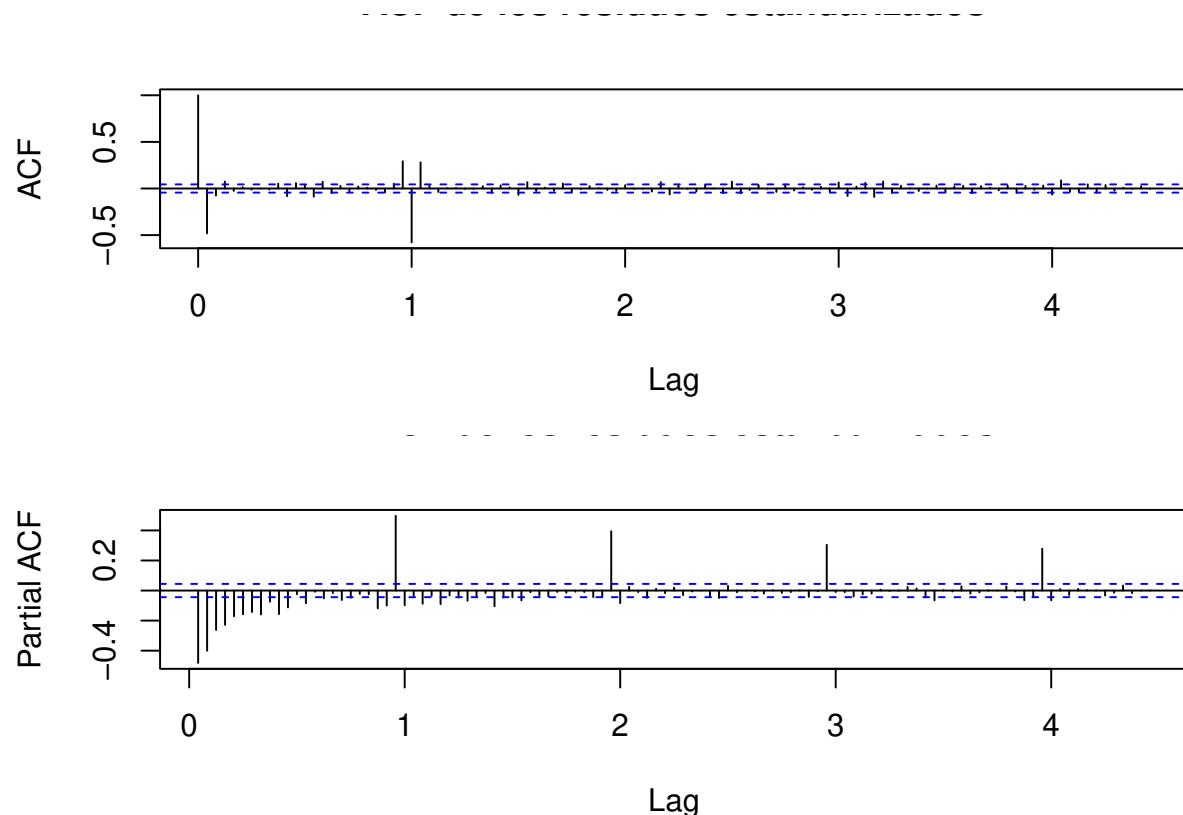
Una vez entrenado el modelo SARIMA debemos diagnosticar los residuos verificando que cumple los supuestos ARIMA:

-Autocorrelación: Los residuos no deben mostrar autocorrelación significativa, es decir, no deben exhibir patrones discernibles en sus autocorrelogramas

-Normalidad: Los residuos del modelo deben seguir una distribución normal.

-Estacionariedad: Los residuos deben ser estacionarios, lo que significa que su media y varianza deben ser constantes a lo largo del tiempo

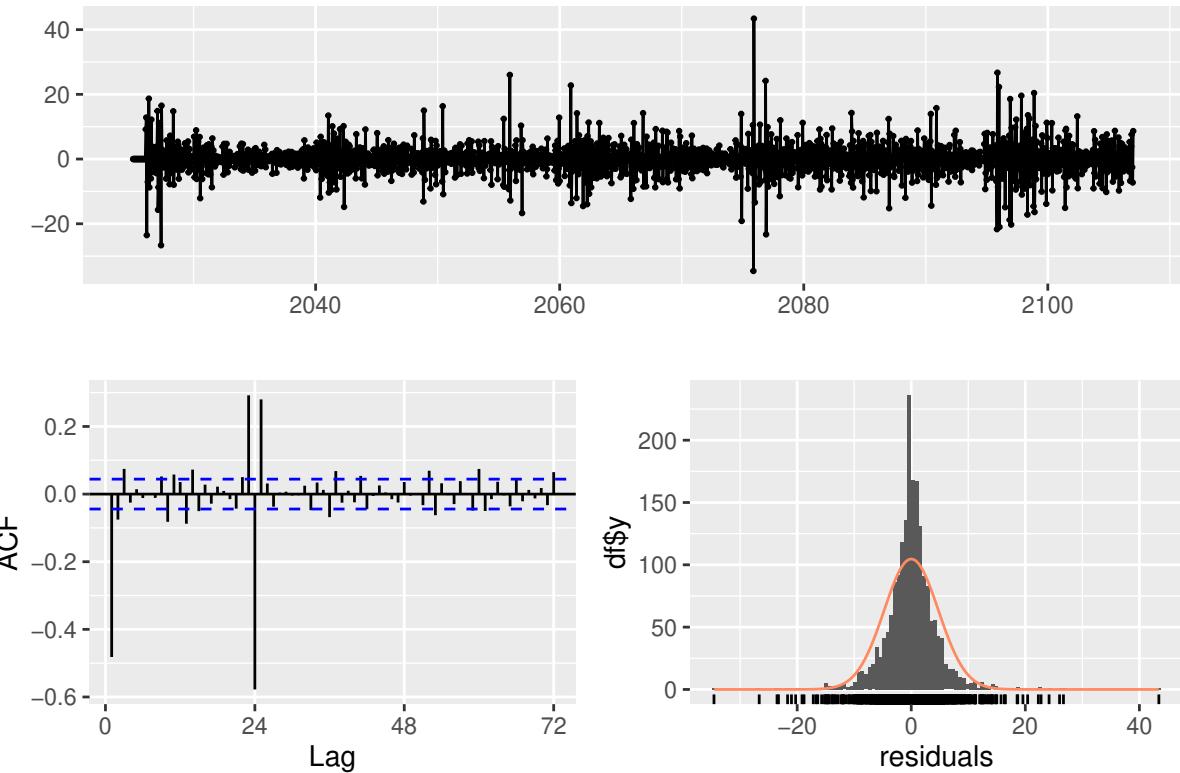
Verificamos la autocorrelación.



La ACF de los residuos no parece mostrar estructura y tiene casi todos los valores dentro de las bandas de confianza. Aun así en la PACF los retardos múltiplos de 24 se salen de las bandas de confianza. Todo apunta a que los residuos no son aleatorios. Aun así continuamos con la diagnosis.

Realizamos el test de Ljung-Box.

### Residuals from Regression with ARIMA(0,1,0)(0,1,0)[24] errors

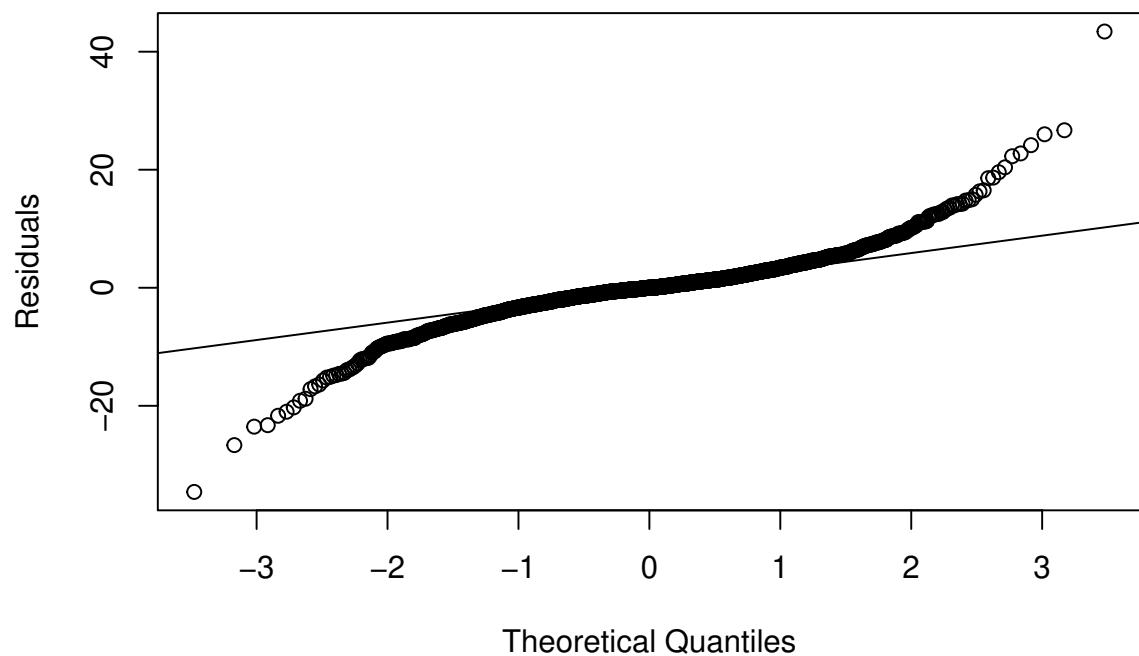


```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(0,1,0)(0,1,0)[24] errors  
## Q* = 1595.6, df = 48, p-value < 2.2e-16  
##  
## Model df: 0. Total lags used: 48
```

El p-valor del test de Ljung-Box es menor que 0.05 luego se puede rechazar que las primeras autocorrelaciones sean nulas, y no se puede asumir que los residuos sean ruido blanco.

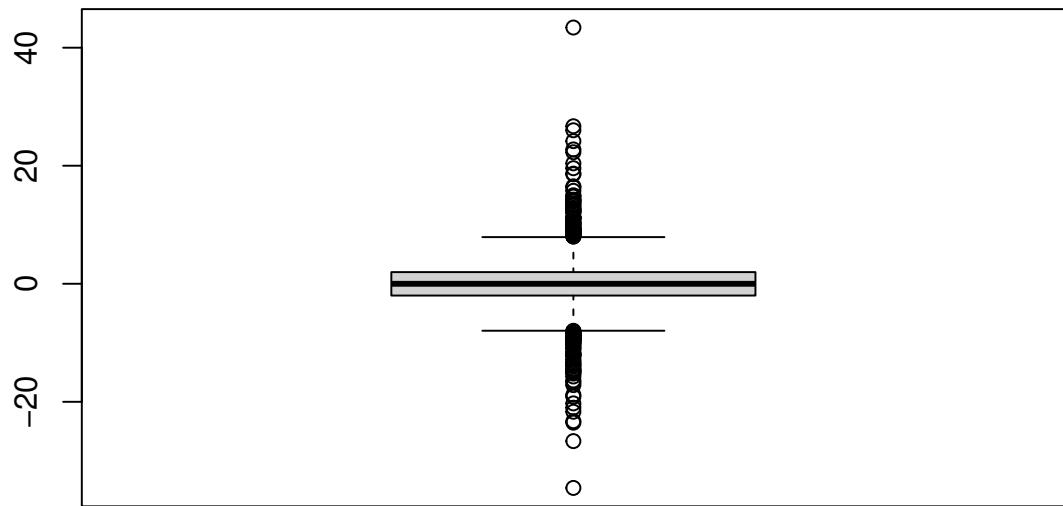
Ahora si, en la ACF, podemos ver una notable correlación con lag igual a 24. Los residuos sí parecen ajustarse a una distribución normal. En la gráfica de los residuos podemos observar que si parecen ser estacionarios. Visualizamos el QQ Plot de los residuos.

### Normal Q-Q Plot



Podemos ver que los residuos aproximadamente siguen una distribución normal, a pesar de la gran cantidad de valores atípicos.

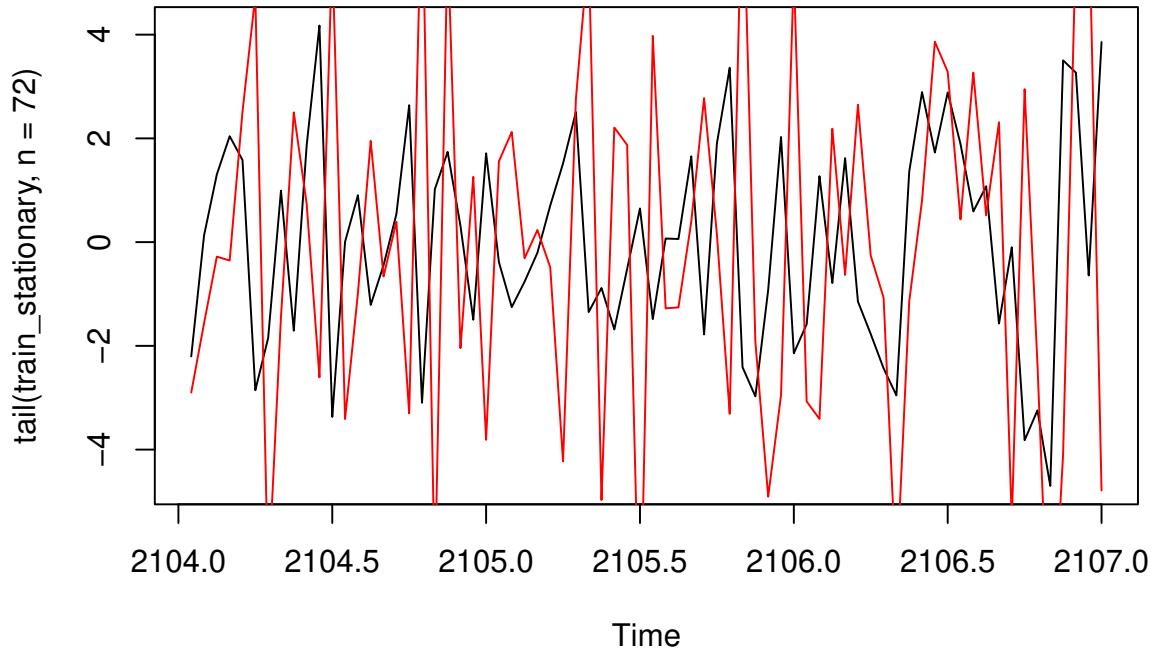
## Boxplot de los residuos



En este box plot observamos mas fácilmente la cantidad de valores atípicos presentes. Aun así la mayoría de residuos están entorno al 0.

Tras analizar los residuos podemos concluir que el modelo no supera la diagnosis. Aun así, si empíricamente el modelo tiene buena capacidad predictiva no lo descartaremos por razones prácticas.

Veamos gráficamente la diferencia entre la serie original y el modelo ajustado (en rojo)



Como se puede apreciar el modelo no se ajusta correctamente a nuestra serie.

Realizamos predicciones con los datos de validación para evaluar el modelo:

```
##               ME      RMSE      MAE      MPE MAPE      MASE      ACF1
## Training set 0.006341458 4.756403 3.097877  NaN Inf 1.668222 -0.4817001
## Test set     -35.381709379 40.217503 35.406744 -Inf Inf 19.066706      NA
```

Como podemos ver nuestro modelo tiene unos errores altos y muy superiores a los obtenidos en train. Por ello haremos uso de la función AutoArima que hará una búsqueda de parámetros con el objetivo de obtener el mejor modelo ARIMA desde un enfoque empírico, en vez de teórico.

El mejor modelo hallado es con los parámetros ARIMA(2,0,0)(2,1,0)[24].

Análogo al entrenamiento del modelo teórico.

```
## Series: train_stationary
## Regression with ARIMA(2,0,0)(2,1,0) [24] errors
##
## Coefficients:
##             ar1      ar2      sar1      sar2 demanda      EUA  precio_gas
##             -0.0553  -0.1495  -0.8499  -0.4626    5e-04  0.0589      0.6635
##             s.e.    0.0226   0.0225   0.0206   0.0209   1e-04  0.1207      0.1728
##             prod_eolica  prod_solar demanda_residual rampa
##                     -5e-04       -6e-04          2e-04  1e-04
##             s.e.      1e-04       1e-04          1e-04  1e-04
```

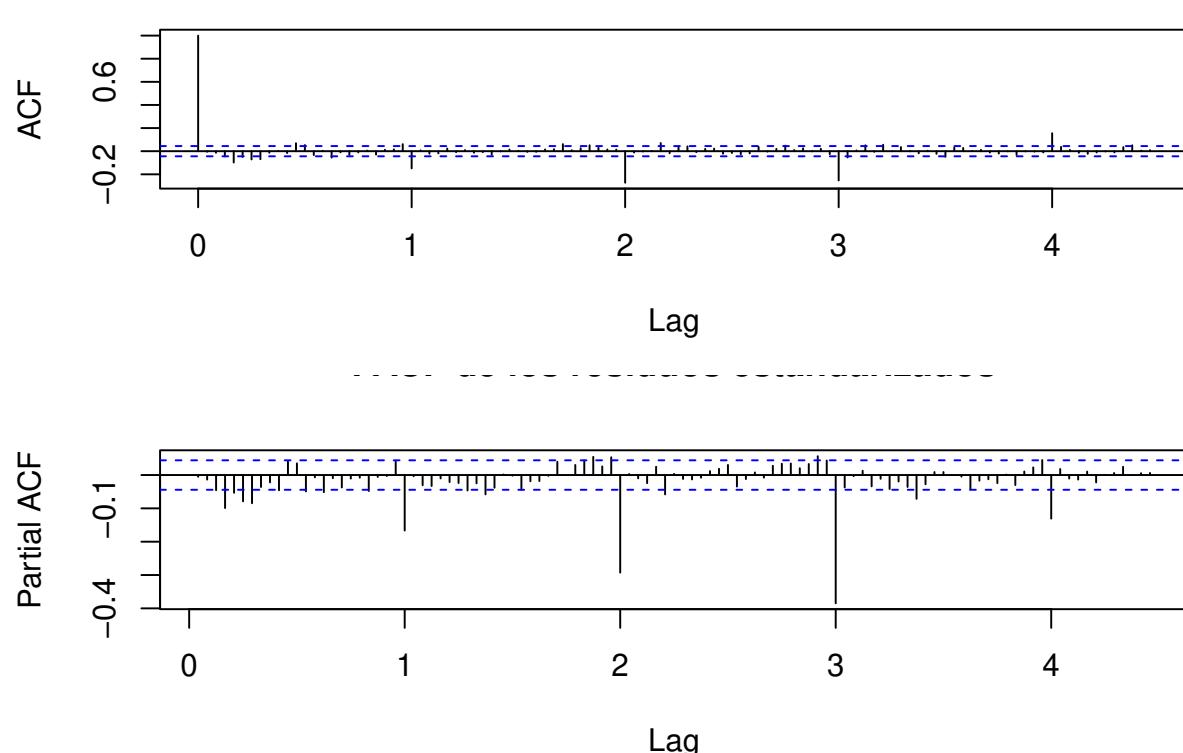
```

## 
## sigma^2 = 5.394: log likelihood = -4401.68
## AIC=8827.36   AICc=8827.52   BIC=8894.23
## 
## Training set error measures:
##               ME      RMSE      MAE MPE MAPE      MASE      ACF1
## Training set -0.006693363 2.301674 1.506854 NaN  Inf 0.6900941 -0.005332612

```

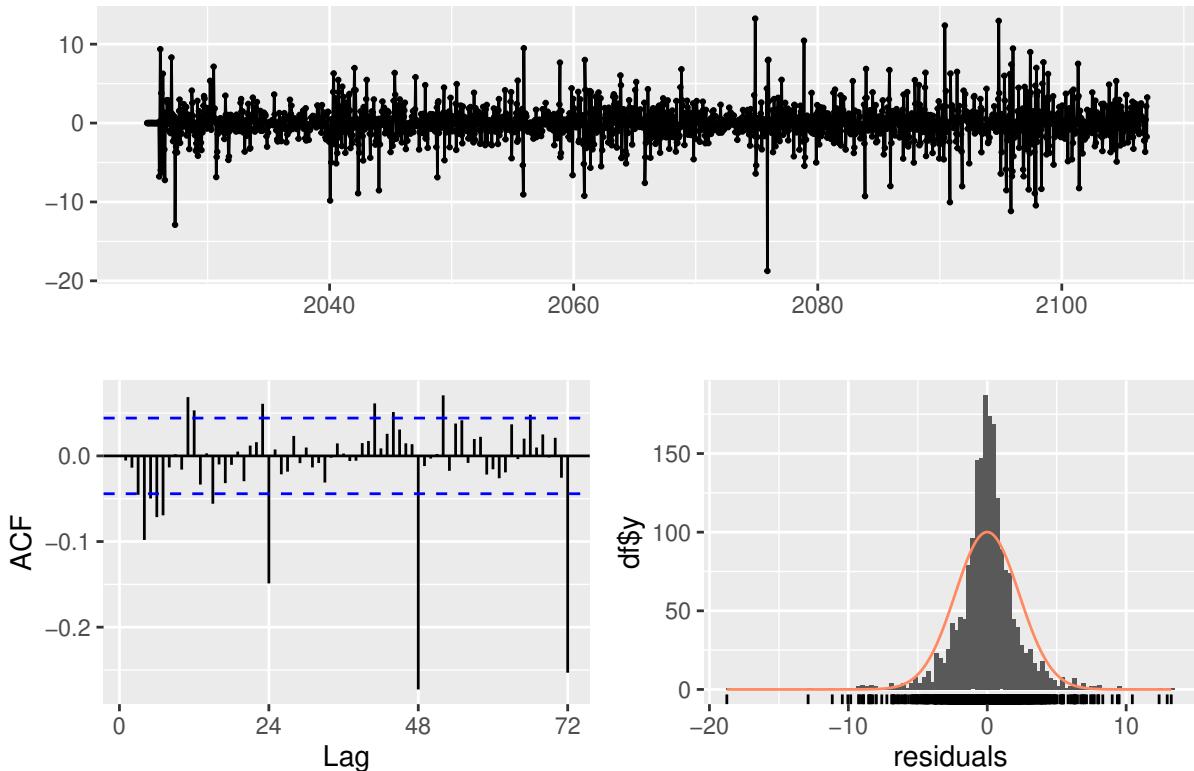
En primera instancia se observa una ligera mejora en entrenamiento respecto al modelo anterior. EL AIC ha disminuido notablemente.

Realizamos la diagnosis del modelo de forma análoga al anterior.



En esta ocasión si que hay una notable correlación en los retardos múltiplos de 24. Lo vemos mas claramente a continuacion:

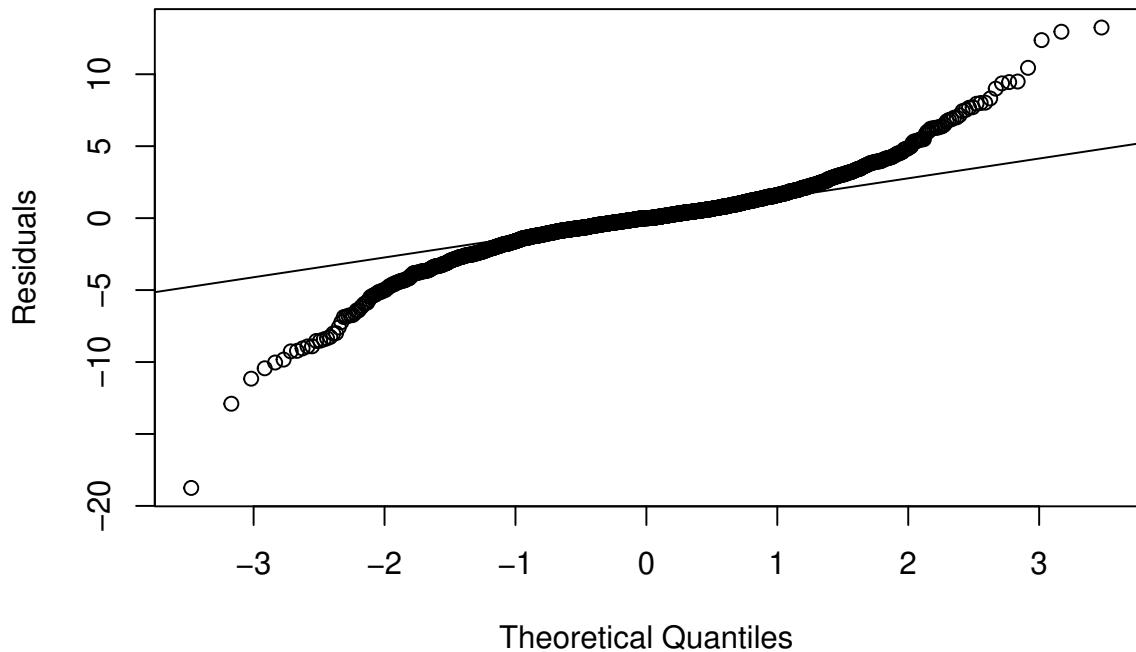
### Residuals from Regression with ARIMA(2,0,0)(2,1,0)[24] errors



```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(2,0,0)(2,1,0)[24] errors  
## Q* = 303.34, df = 44, p-value < 2.2e-16  
##  
## Model df: 4. Total lags used: 48
```

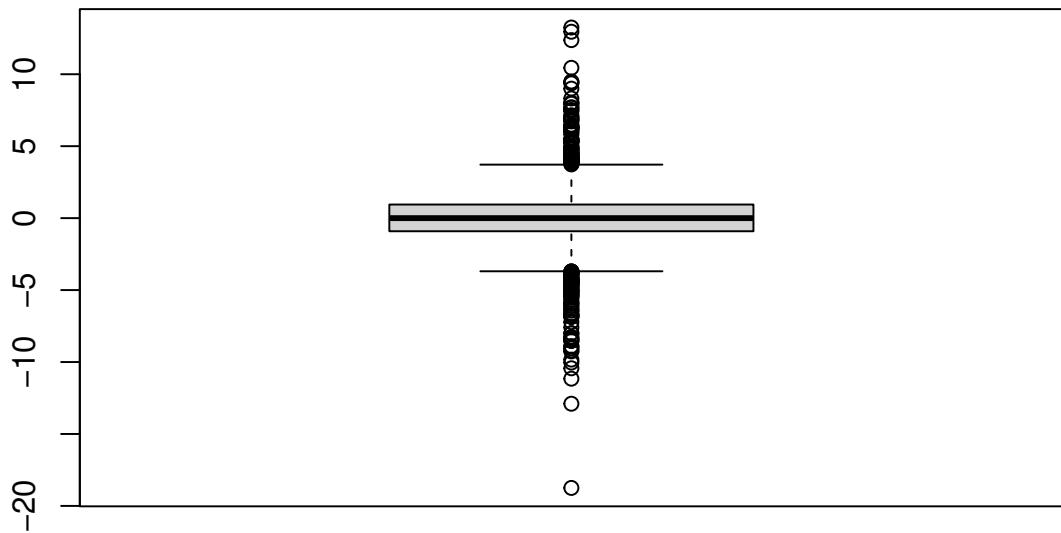
El p-valor de test de Ljung-Box es menor que 0.05 luego se puede rechazar que las primeras autocorrelaciones sean nulas, y no se puede asumir que los residuos sean ruido blanco. En la ACF podemos ver una notable correlación con lag igual a 24 y múltiplos de él. Los residuos sí parecen ajustarse a una distribución normal. En la gráfica de los residuos podemos observar que si parecen ser estacionarios.

### Normal Q-Q Plot



Podemos ver que los residuos aproximadamente siguen una distribución normal, a pesar de la gran cantidad de valores atípicos.

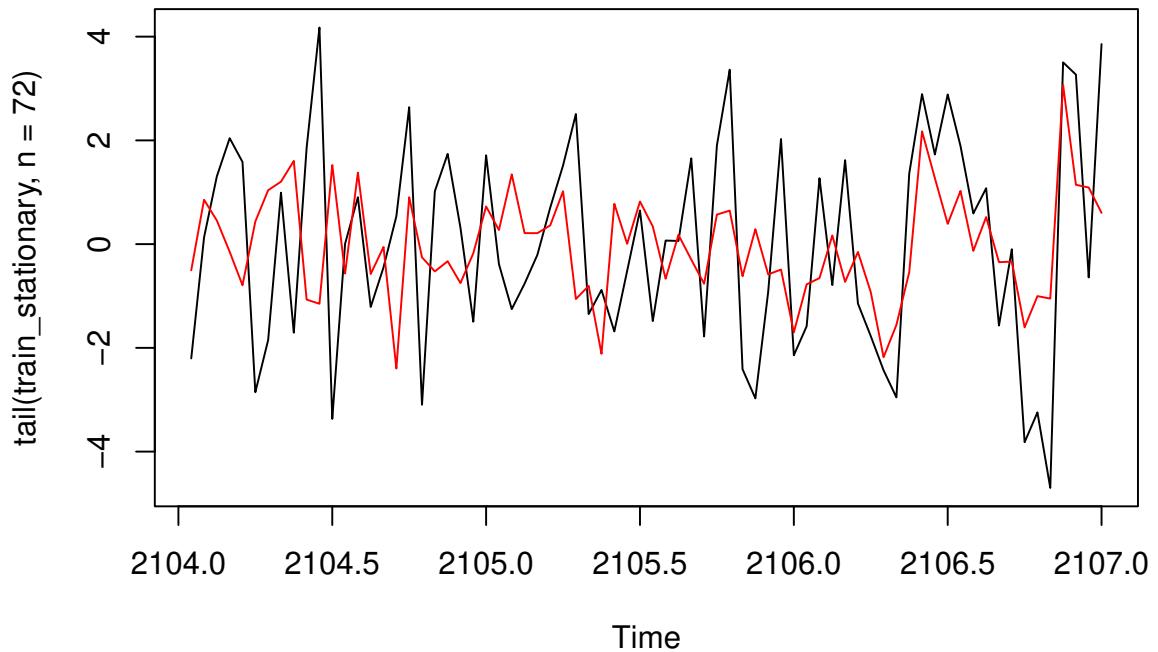
## Boxplot de los residuos



En este box plot observamos mas fácilmente la cantidad de valores atípicos presentes. Aun así la mayoría de residuos están entorno al 0.

Tras analizar los residuos podemos concluir que este modelo ARIMA tampoco supera la diagnosis. Aun así, al igual que antes, si el modelo tiene buena capacidad predictiva lo consideraremos valido.

Veamos gráficamente la diferencia entre la serie original y el modelo ajustado (en rojo)



El modelo da unos resultados bastante mejores que los del anterior modelo. Aun así no se acerca a lo esperado en este trabajo.

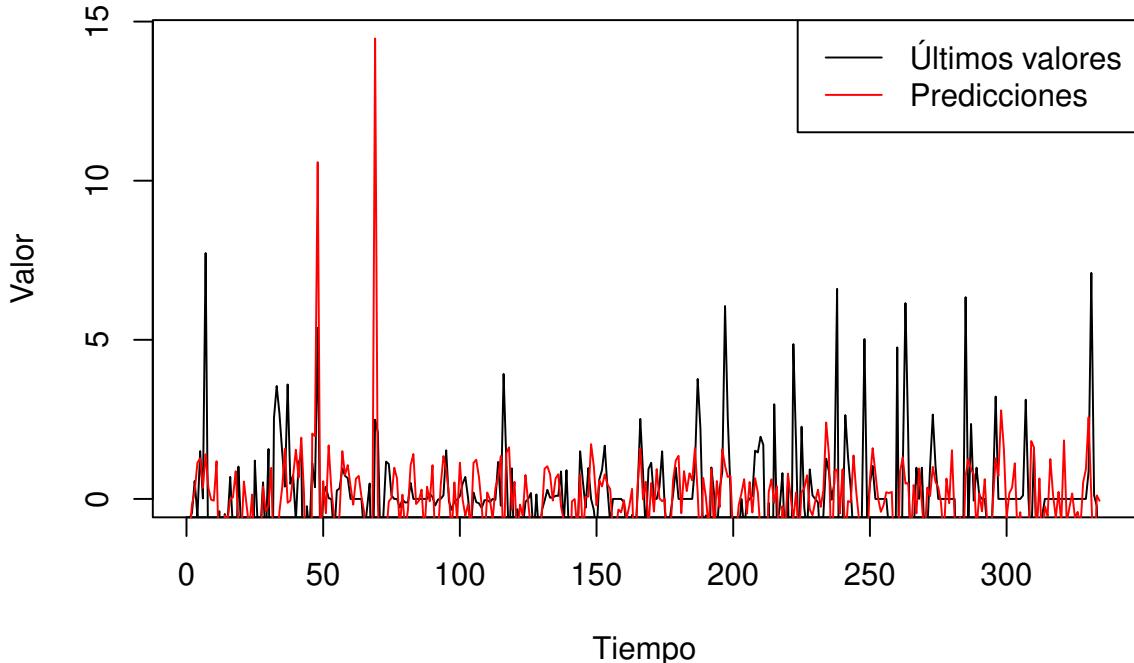
Realizamos predicciones con los datos de validación para evaluar el modelo:

```
##               ME      RMSE     MAE MPE MAPE      MASE      ACF1
## Training set -0.006693363 2.301674 1.506854 NaN Inf 0.8114482 -0.005332612
## Test set      0.014231453 2.317620 1.474850 NaN Inf 0.7942138      NA
```

Observamos como efectivamente obtenemos unos resultados mejores que con el anterior modelo. Las métricas de error tanto en train como en validación son muy buenas y prácticamente no hay diferencia entre ellas, lo cual es algo muy positivo.

Procedemos a visualizar las predicciones vs valores reales en validación.

## Últimos valores y predicciones



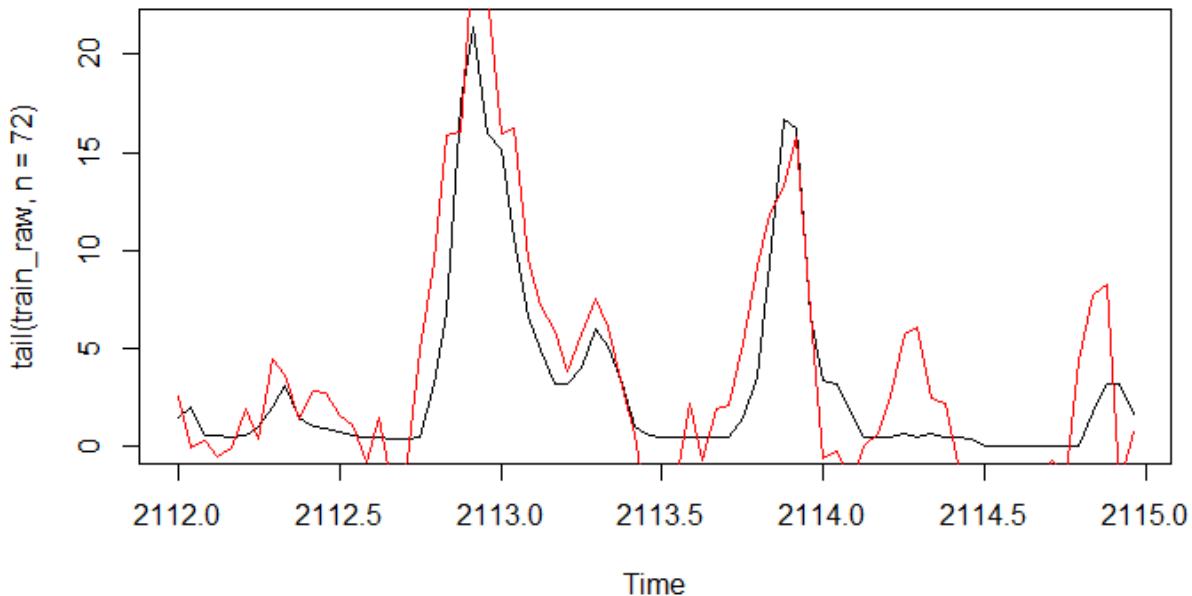
Como se puede observar, pese a que las métricas en si son buenas, las predicciones no lo son. Parece que un modelo SARIMAX no es la mejor aproximación a nuestro problema. Aun así, a modo de ultimo intento, trataremos de entrenar un modelo con la serie original, sin sufrir transformación alguna. Hacemos uso de autoarima. Esta aproximacion obvia toda la base teorica de los modelos ARIMA.

El mejor modelo hallado es ARIMA(1,1,4)(1,0,0)[24]

```
## Series: train_raw
## Regression with ARIMA(1,1,4)(1,0,0) [24] errors
##
## Coefficients:
##             ar1      ma1      ma2      ma3      ma4      sar1    demanda      EUA
##             0.8754   -1.074   -0.1192   0.1463   0.0562   0.2860   0.0031   -0.0794
## s.e.       0.0335    0.042    0.0337   0.0347   0.0305   0.0222   0.0003    0.4991
##             precio_gas  prod_eolica  prod_solar demanda_residual     rampa
##             2.5647     -0.0032     -0.0022                  6e-04   -5e-04
## s.e.       0.7388      0.0003      0.0003                  3e-04   2e-04
##
## sigma^2 = 61.36: log likelihood = -6921.85
## AIC=13871.71  AICc=13871.92  BIC=13950.06
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set -0.08447855 7.805974 4.812318 NaN Inf 0.3239711 0.001803556
```

Las métricas han empeorado notablemente, el AIC ha aumentado a casi el doble respecto al anterior modelo.

Puesto que en esta implementación estamos obviando toda la base teórica tampoco consideramos oportuno realizar la diagnosis del modelo, tan solo observaremos su capacidad predictiva.



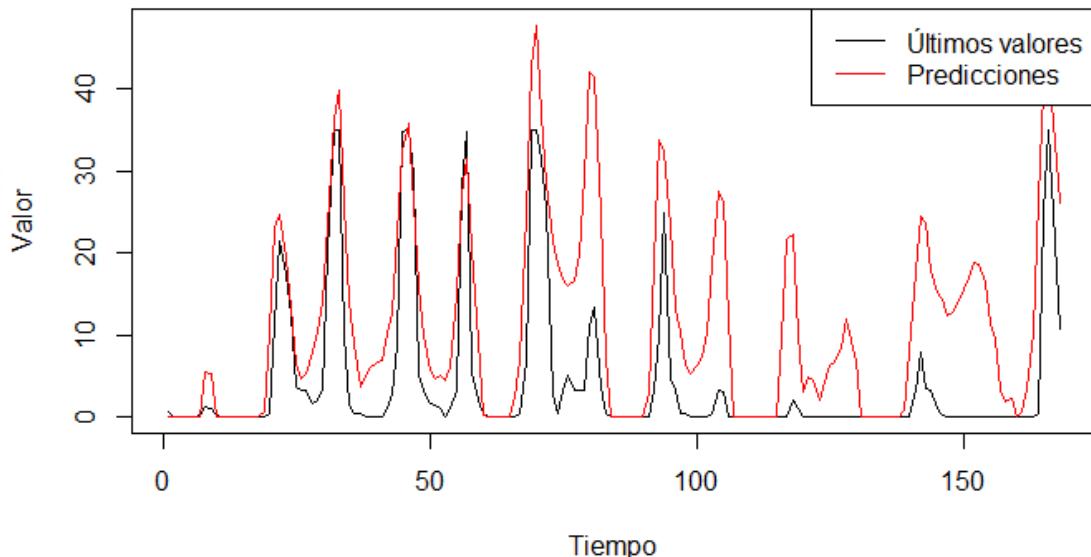
Sorprendentemente en esta ocasión si que parece ajustarse bien a nuestra serie temporal. Realizamos predicciones con el conjunto de validación y evaluamos los resultados.

```
##               ME      RMSE      MAE     MPE MAPE      MASE      ACF1
## Training set -0.08447855 7.805974 4.812318  NaN  Inf 0.7107139 0.001803556
## Test set      -12.16307290 15.539450 12.948720 -Inf  Inf 1.9123497          NA
```

Las métricas obtenidas han empeorado, sin embargo acabamos de ver que estas no reflejan bien la capacidad predictiva del modelo. Procedemos a visualizar las predicciones con el conjunto de validación.

## Últimos valores y predicciones

### Últimos valores y predicciones



Como podemos ver el modelo ajusta muy bien la forma pero no la magnitud. Puesto que este modelo es el que mejor se ajusta a nuestra serie lo seleccionaremos como modelo ganador.

Curiosamente sera este ultimo modelo ARIMA que obvia toda la base teórica nuestro modelo ganador. Sera este modelo y no cualquiera de los otros 2, pese a tener peores métricas, ya que visualmente observamos que es el que mejor modela nuestra serie temporal.

Los resultados obtenidos aun así no son satisfactorios, continuaremos estudiando modelos mas avanzados con el objetivo de realizar mejores predicciones que las obtenidas aquí.



## **Apéndice B**

### **Segundo anexo**

<b>model</b>	<b>loss</b>	<b>epochs</b>	<b>gradient</b>	<b>hidden</b>	<b>dropout</b>	<b>h.cont</b>	<b>att. head</b>	<b>lr</b>
m_113_72	1.618	30	0.01	36	0.30	22	4	0.010
m_133_72	2.063	25	0.01	36	0.30	22	4	0.001
m_365_360	3.133	20	0.03	24	0.25	24	4	0.005
m_365_360	3.347	20	0.03	24	0.25	24	4	0.010
m_365_168	3.640	30	0.31	13	0.29	8	1	0.002
m_365_72	3.703	25	0.01	36	0.30	22	4	0.001
m_365_168	3.740	25	0.01	36	0.30	22	4	0.001
m_200_72	3.748	40	0.03	24	0.25	24	2	0.010
m_365_360	3.846	40	0.03	24	0.25	24	2	0.010
m_365_168	4.072	20	0.06	32	0.30	19	2	0.150
m_365_72	4.100	35	0.02	30	0.32	20	3	0.005
m_133_168	4.200	28	0.01	36	0.30	22	4	0.001
m_200_168	4.250	22	0.04	26	0.27	24	2	0.010
m_133_72	4.300	30	0.02	34	0.31	22	4	0.001
m_113_360	4.450	35	0.01	36	0.30	22	4	0.005
m_365_360	4.500	18	0.03	24	0.25	24	4	0.007
m_365_72	4.600	22	0.01	36	0.30	22	4	0.002
m_365_168	4.650	25	0.01	36	0.30	22	4	0.003
m_365_360	4.700	28	0.03	24	0.25	24	4	0.009
m_133_72	4.800	20	0.02	36	0.30	22	4	0.004
m_365_72	4.850	26	0.01	36	0.30	22	4	0.003
m_365_168	4.900	25	0.01	36	0.30	22	4	0.005
m_113_72	5.000	30	0.01	36	0.30	22	4	0.007
m_365_360	5.100	20	0.03	24	0.25	24	4	0.008
m_200_72	5.200	22	0.04	26	0.27	24	2	0.011
m_365_168	5.300	28	0.06	32	0.30	19	2	0.150
m_365_360	5.400	35	0.03	24	0.25	24	4	0.009
m_113_360	5.500	18	0.01	36	0.30	22	4	0.004
m_365_168	5.600	22	0.01	36	0.30	22	4	0.002
m_133_72	5.700	30	0.02	36	0.30	22	4	0.001

Cuadro B.1: Experimentos TFT.

## Apéndice C

### Tercer anexo

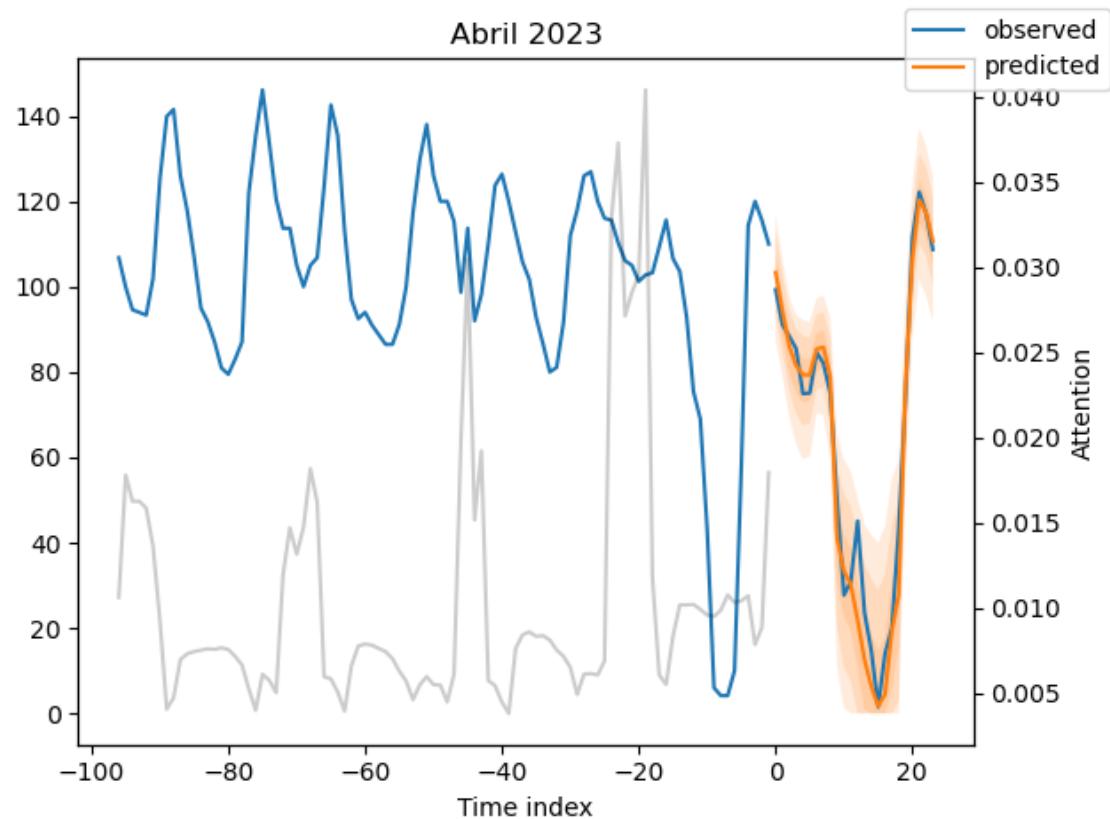


Figura C.1: Predicción para Abril 2023

## Capítulo C. Tercer anexo

---

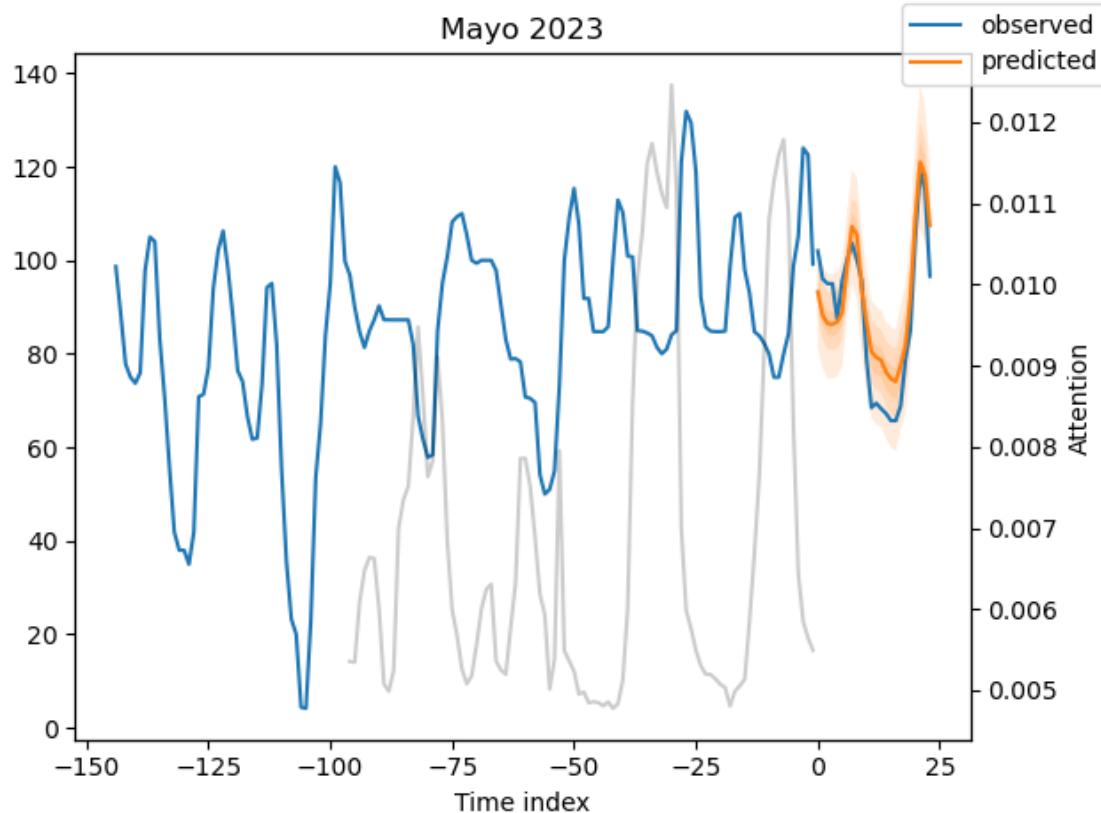


Figura C.2: Predicción para Mayo 2023

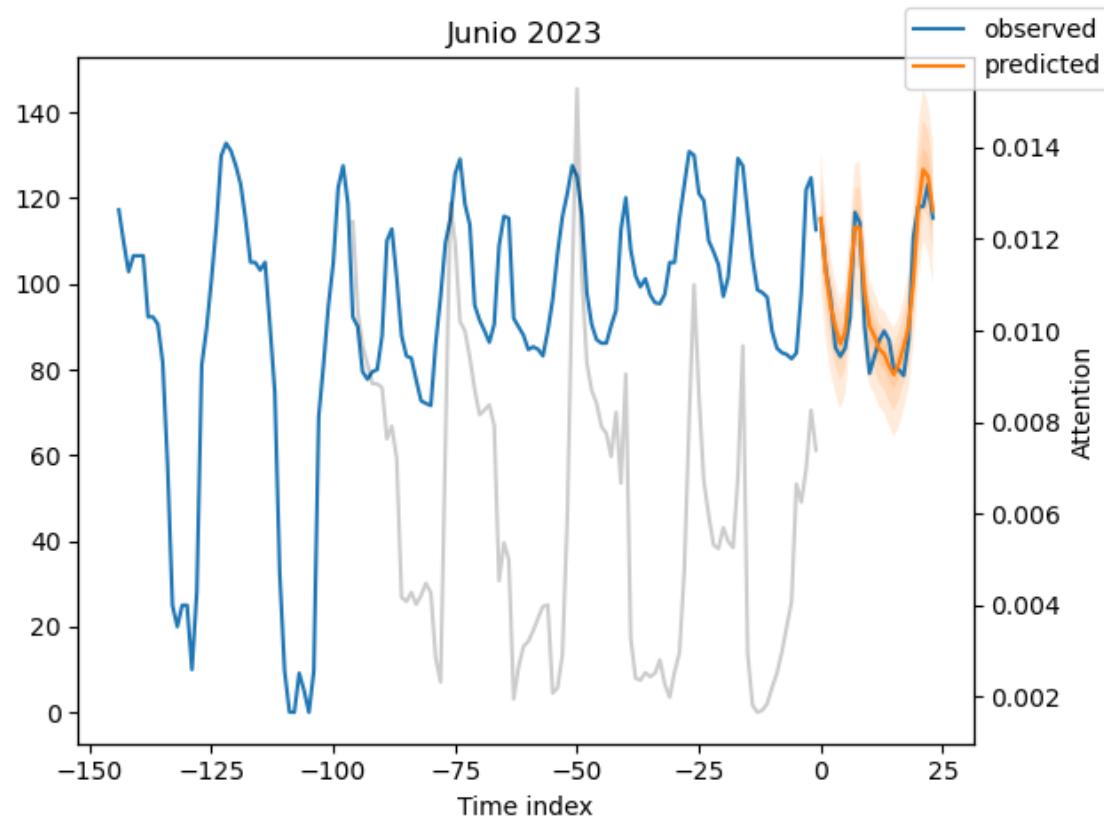


Figura C.3: Predicción para Junio 2023

## Capítulo C. Tercer anexo

---

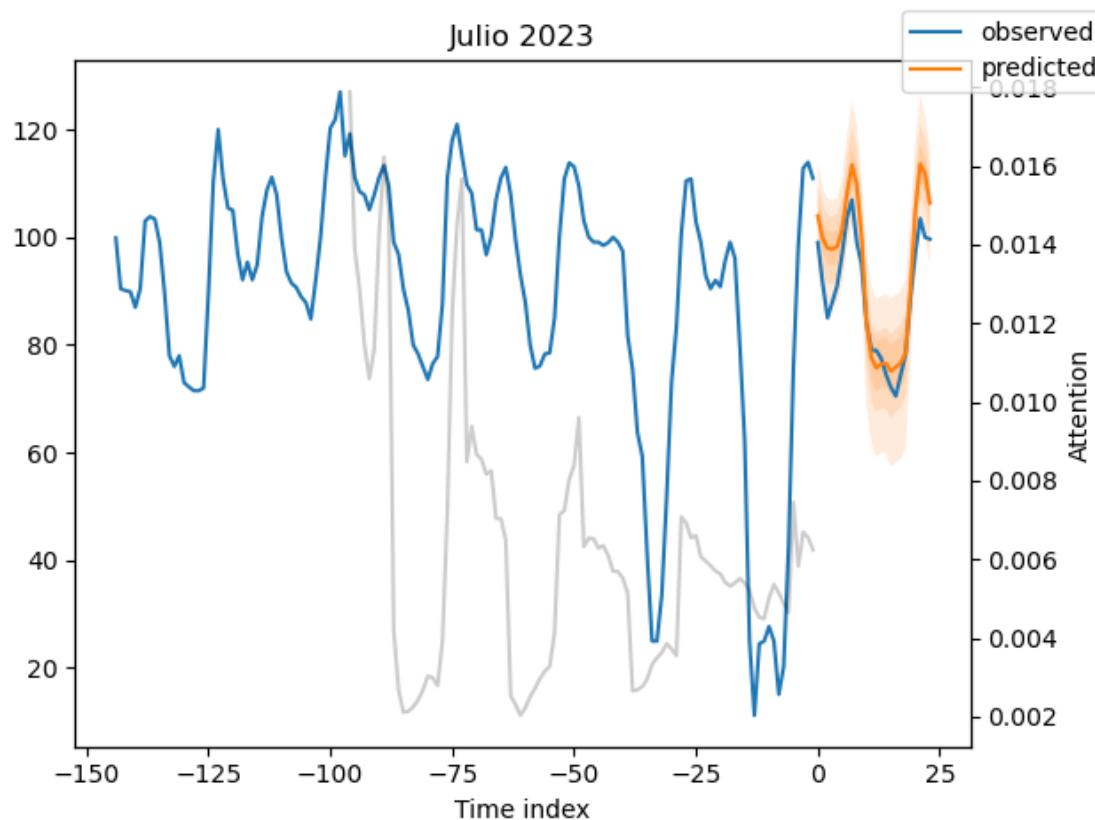


Figura C.4: Predicción para Julio 2023

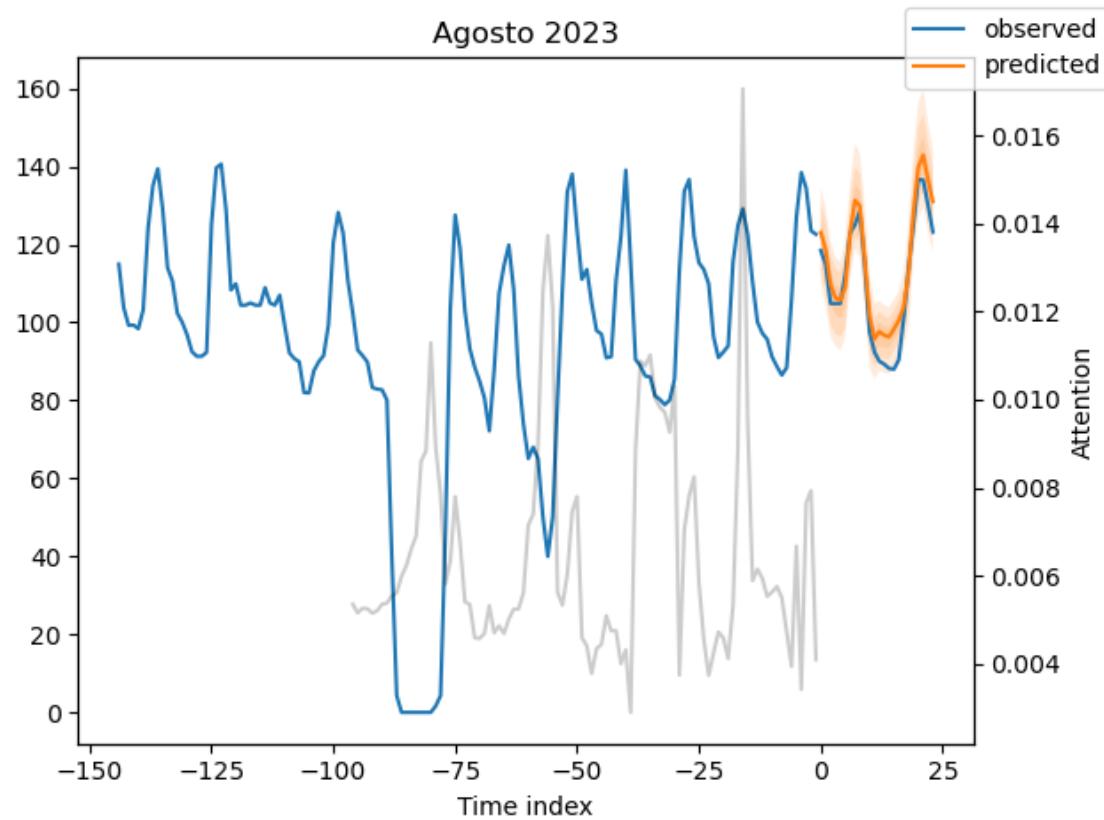


Figura C.5: Predicción para Agosto 2023

## Capítulo C. Tercer anexo

---

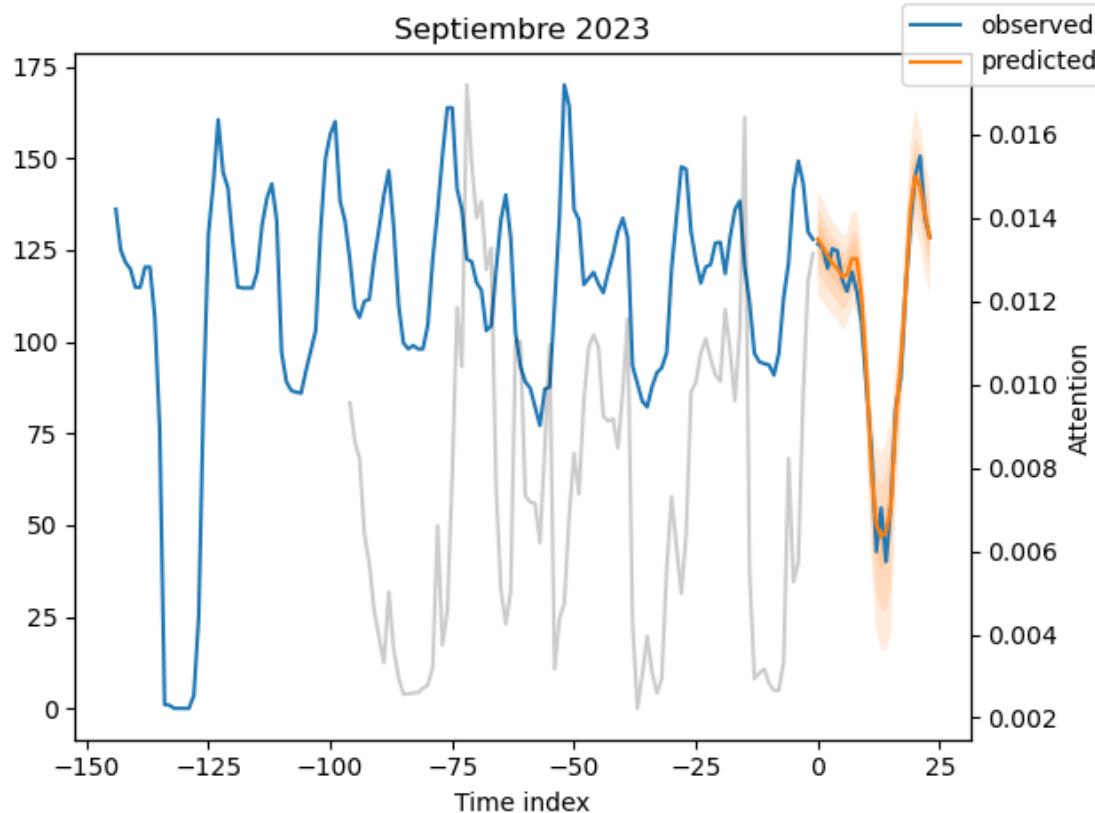


Figura C.6: Predicción para Septiembre 2023

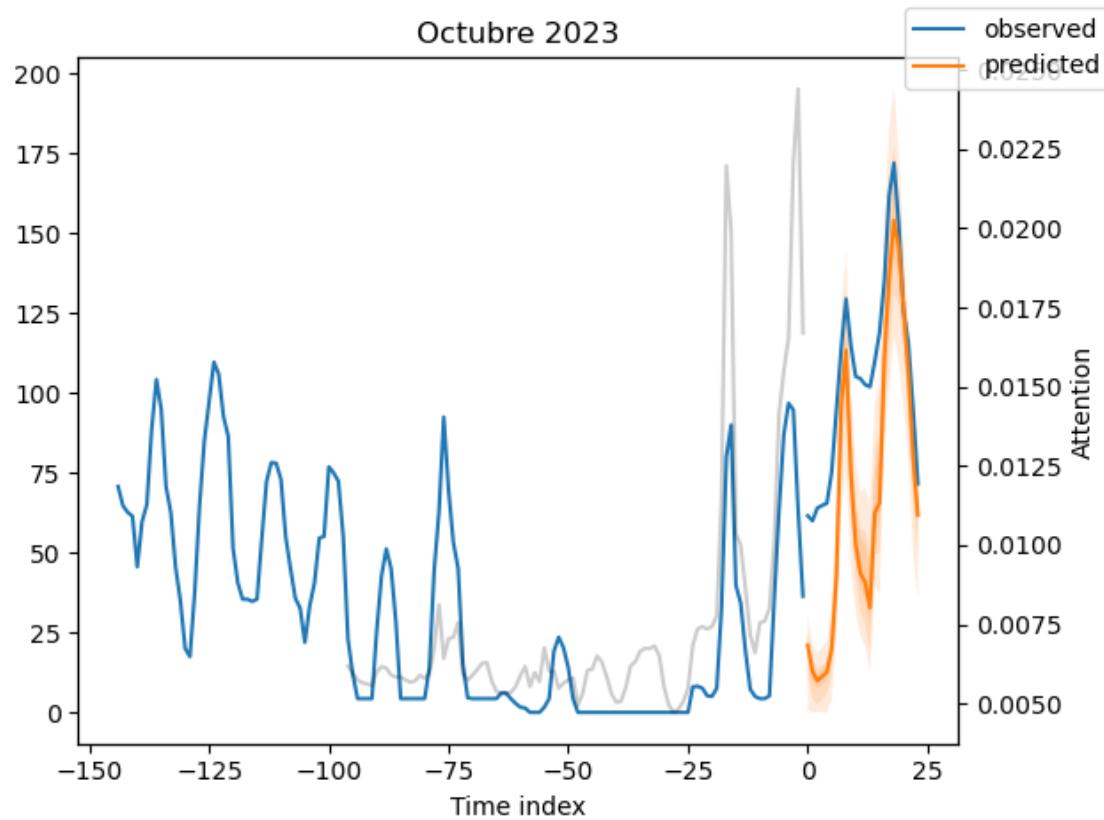


Figura C.7: Predicción para Octubre 2023

## Capítulo C. Tercer anexo

---

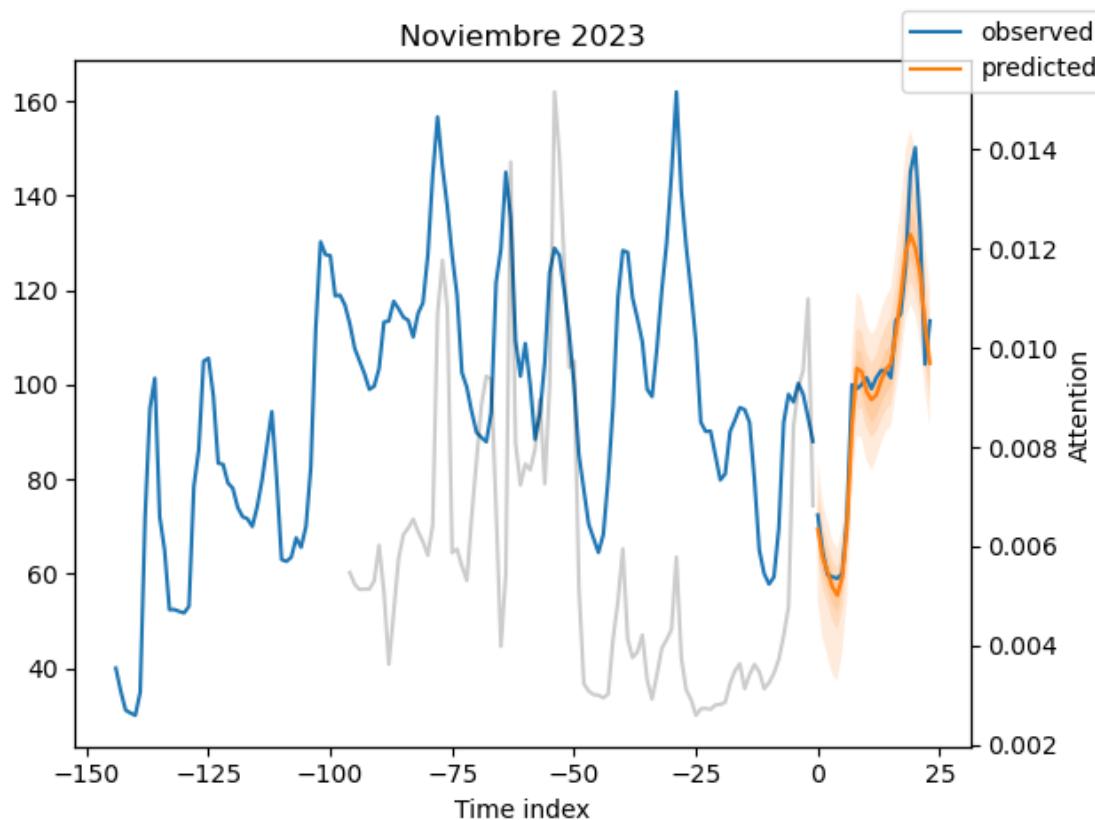


Figura C.8: Predicción para Noviembre 2023

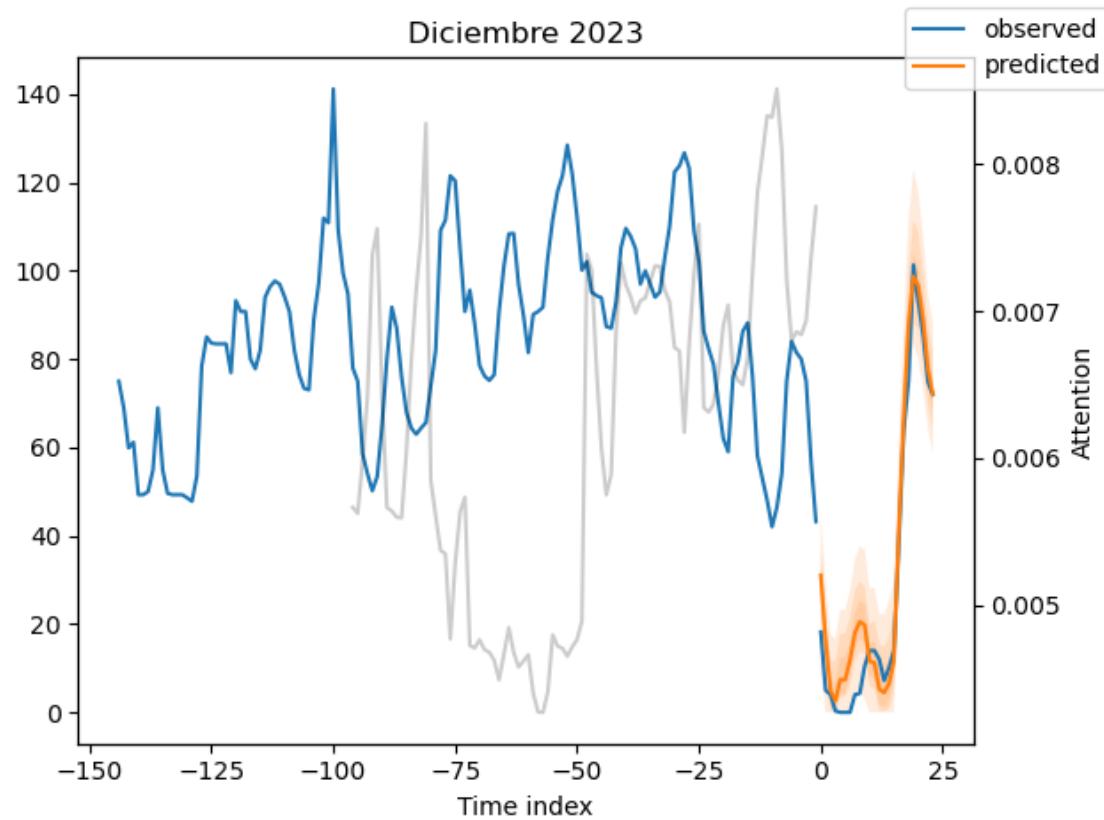


Figura C.9: Predicción para Diciembre 2023

## Capítulo C. Tercer anexo

---

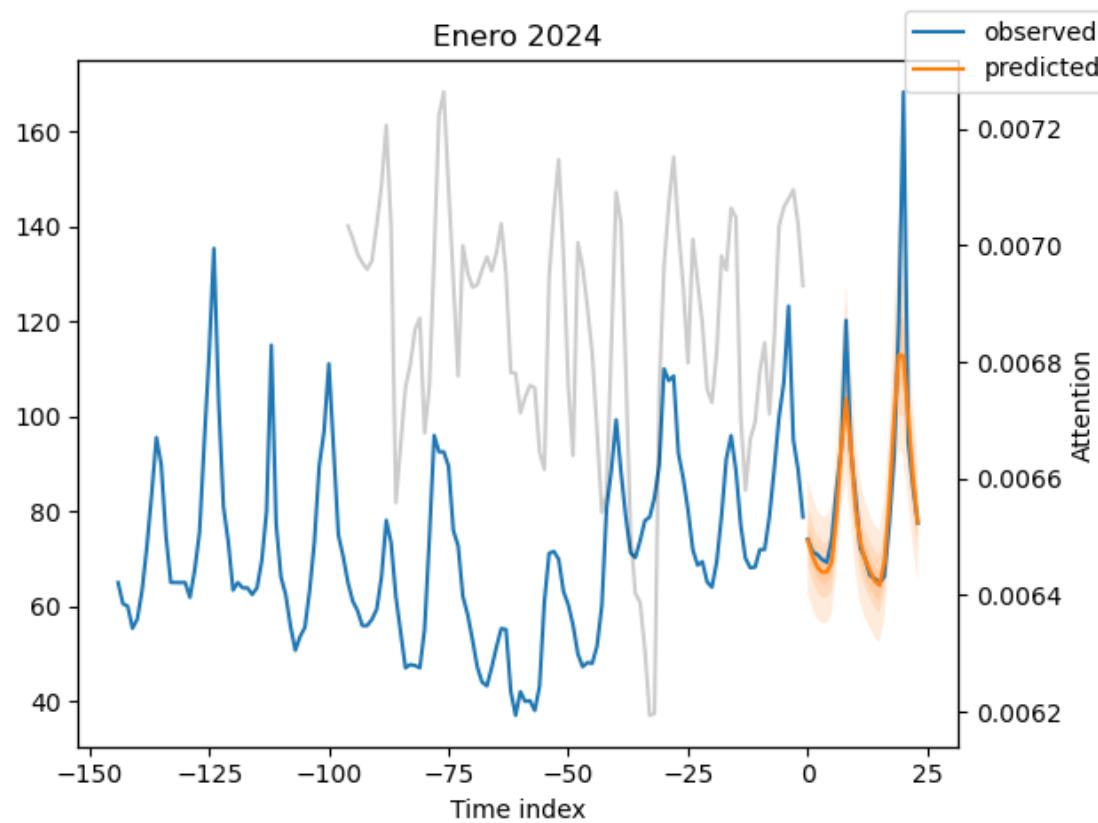


Figura C.10: Predicción para Enero 2024

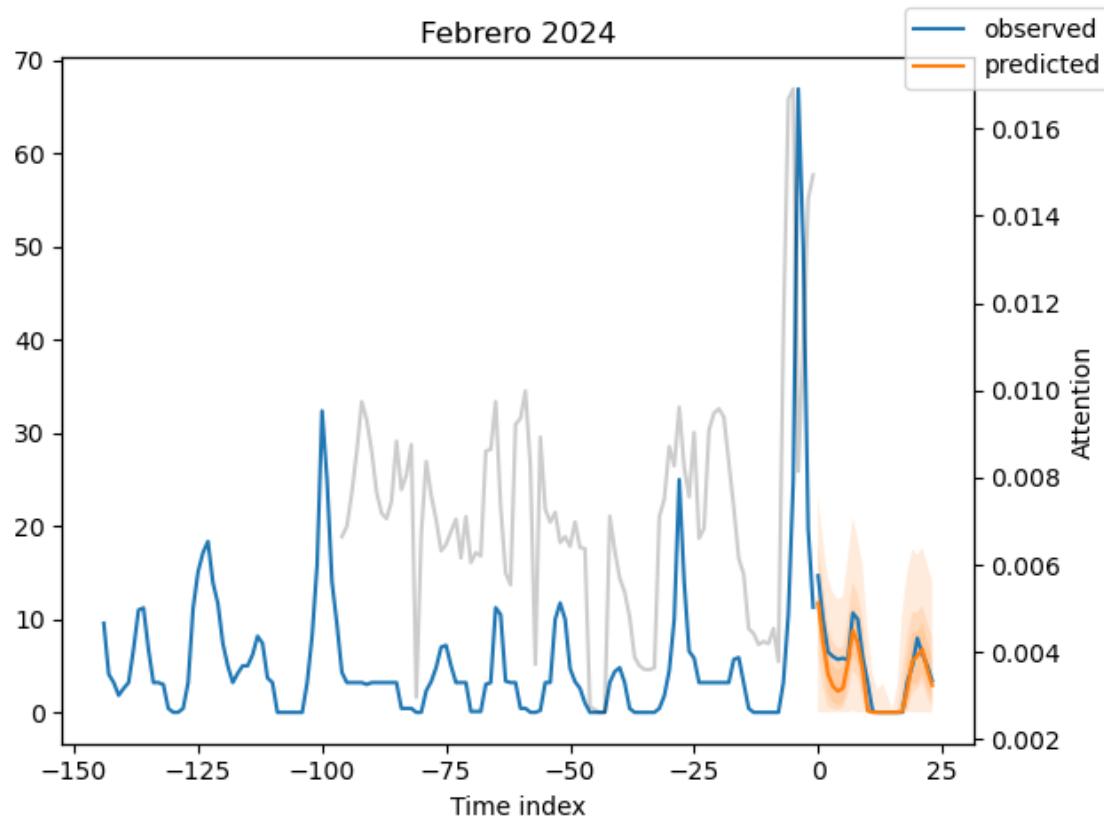


Figura C.11: Predicción para Febrero 2024

## Capítulo C. Tercer anexo

---

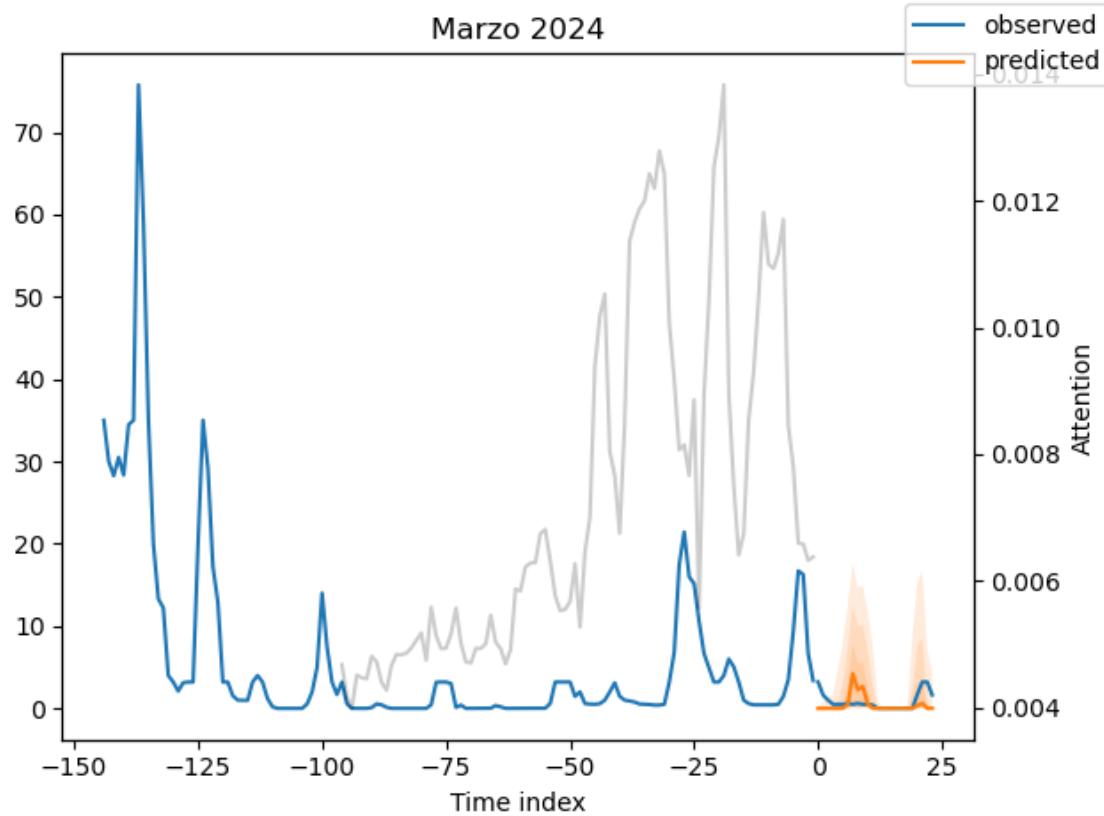


Figura C.12: Predicción para Marzo 2024

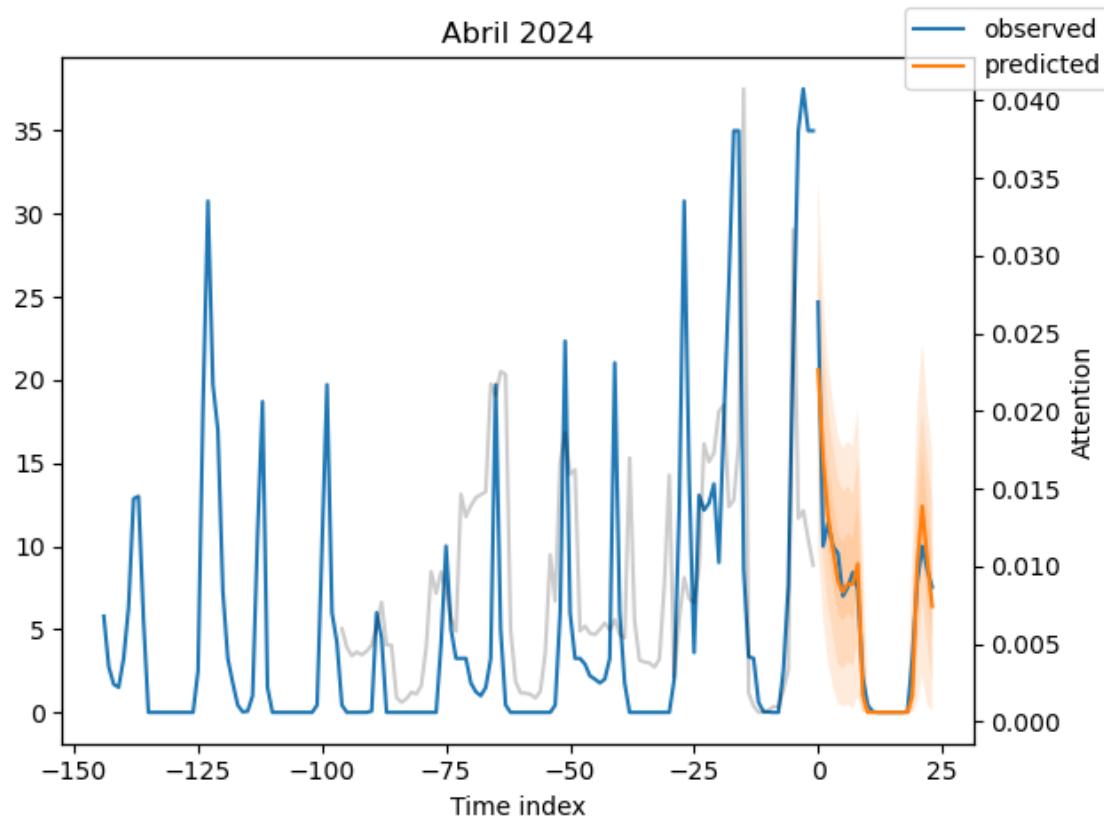


Figura C.13: Predicción para Abril 2024