UNIVERSIDAD DE GRANADA

ESTUDIO DE TÉCNICAS DE PREDICCIÓN MULTIVARIABLE DE DEMANDA ENERGÉTICA.

MARÍA AGUADO MARTÍNEZ

Trabajo Fin de Grado

Doble Grado en Ingeniería Informática y Matemáticas

Tutores
Miguel Molina Solana
Javier Jiménez

FACULTAD DE CIENCIAS
E.T.S. INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Granada, 26 de junio de 2023

Agradecimientos

Me gustaría dedicar este trabajo a aquellos que me han acompañado estos 5 años. Dicen que lo más bonito del viaje reside en el camino, y yo no podría haberlo hecho sin la ayuda de mi familia, mi pareja y de todos los que han estado a mi lado mientras lo recorría.

Por supuesto, tengo que agradecer a mis dos tutores, D. Miguel Molina y D. Javier Jiménez, por la confianza desde el primer momento en el que se planteó la oportunidad. Han sido muchas las horas dedicadas, pero ha sido un placer trabajar con vosotros.

DECLARACIÓN DE ORIGINALIDAD DEL TFG

Yo, **María Aguado Martínez**, con DNI 71168499J, declaro que el presente Trabajo de Fin de Grado es original, no habiéndose utilizado fuentes sin ser citadas debidamente. De no cumplir con este compromiso, soy consciente de que, de acuerdo con la Normativa de Evaluación y de Calificación de los estudiantes de la Universidad de Granada, de 20 de mayo de 2013, esto c*onllevará automáticamente la calificación numérica de cero [...] independientemente del resto de las calificaciones que el estudiante hubiera obtenido. Esta consecuencia debe entenderse sin perjuicio de las responsabilidades disciplinarias en las que pudieran incurrir los estudiantes que plagien.*

Para que así conste lo firmo el 26 de junio de 2023

Granada, a 26 de junio de 2023

Yo, **María Aguado Martínez**, alumna del Doble Grado en Ingeniería Informática y Matemáticas de la **Facultad de Ciencias de la Universidad de Granada**, con DNI 71168499J, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Granada, a 26 de junio de 2023

D. Miguel Molina Solana y **D. Javier Jiménez**, investigadores del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Estudio de técnicas de predicción multivariable de demanda energética***, ha sido realizado bajo su supervisión por María Aguado Martínez, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda. Y para que conste, expiden y firman el presente informe en Granada a 26 de junio de 2023.

Granada, a 26 de junio de 2023

Estudio de técnicas de predicción multivariable de demanda energética

María Aguado Martínez

Palabras clave

Series temporales, eficiencia energética, LSTM, Machine Learning, imputación univariable

Resumen

La degeneración del planeta debida a las emisiones descontroladas de CO_2 es uno de los problemas principales a los que nos enfrentamos en los últimos tiempos. El sector inmobiliario ha sido identificado como uno de los principales responsables, representando más del 40 por ciento del uso global de energía y aproximadamente el 30 por ciento de las emisiones totales de gases perjudiciales para la atmósfera. Por este motivo, comprender y corregir los patrones de consumo en los inmuebles es fundamental para aumentar la eficiencia energética.

El objetivo principal de este trabajo es investigar las distintas técnicas matemáticas y estadísticas que pueden ser utilizadas para predecir el consumo de energía en edificios. En particular, se han analizado las variables medioambientales recogidas a lo largo de tres años en un edificio de oficinas en California.

En este contexto, los datos suelen presentar problemas de imprecisión o incluso ausencia de los mismos. Para solventarlo se ha implementado un algoritmo de imputación univariable mediante el uso de algoritmos de Machine Learning; en particular, Prophet y Support Vector Regression.

Una vez procesados los datos, se han estudiado las propiedades principales de las series temporales como la estacionalidad, estacionariedad y las posibles descomposiciones, mediante el uso de visualizaciones y pruebas estadísticas, y la relación entre el conjunto de variables consideradas.

Por último, se han llevado a cabo experimentos de predicción univariable y multivariable con distintos algoritmos, incluyendo Prophet, ARIMA, SARIMAX, Unobserved Components y TBATS, y se ha realizado una comparación de los resultados obtenidos con distintas configuraciones de ventana móvil. También se ha experimentado con distintas arquitecturas de redes neuronales LSTM, tanto univariables como multivariables. Finalmente, se ha comparado el valor predictivo de los modelos obtenidos utilizando diferentes medidas como MAE, MSE y MAPE.

Study of Multivariate Prediction Techniques of Energy Demand

María Aguado Martínez

Keywords

Time series, energy efficiency, LSTM, Machine Learning, univariate imputation

Abstract

Evironmental degradation due to uncontrolled CO_2 emissions is one of the major problems we face these days. The real estate sector is considered a major contributor, responsible of over 40 percent of global energy consumption and about 30 percent of total emissions of harmful gases into the atmosphere. For this reason, understanding and modifying consumption patterns within buildings is essential to increasing energy efficiency.

The main purpose of this study is to examine various mathematical and statistical methods that can be used to predict building energy consumption. Specifically, environmental variables collected over a period of three years in an office building in California were analyzed.

In this context, data often shows problems such as incorrect or missing values. To address this issue, an univariate imputation algorithm has been implemented using Machine Learning techniques; particularly, Prophet and Support Vector Regression.

Once the data was processed, the main properties of the time series, such as seasonality, stationarity and possible decompositions, were examined using visualization and statistical tests, as well as the relationships between the variables considered.

Additionally, univariate and multivariate prediction experiments were performed using various algorithms such as Prophet, ARIMA, SARIMAX, Unobserved Components and TBATS. We then compared the results obtained with different moving window configurations.

Finally, experiments were conducted using different LSTM neural network architectures, both univariate and multivariate. In conclusion, the predictions obtained from the models were compared with different statistical measures such as MAE, MSE and MAPE.

ÍNDICE GENERAL

Parte I

INTRODUCCIÓN Y ANTECEDENTES

Exposición general de los objetivos e hipótesis del estudio. Justificación de las técnicas de análisis utilizadas.

1

INTRODUCCIÓN

La degeneración del entorno como consecuencia de la emisión descontrolada de CO_2 hacia la atmósfera es un desafío global al que se están enfrentando los países desarrollados en los últimos tiempos. El creciente nivel de industrialización y urbanización en varios países desarrollados ha provocado un aumento en las emisiones de carbono generadas por actividades industriales [18]. Estas actividades también han causado un incremento significativo en la concentración global de gases de efecto invernadero como por ejemplo el CO_2, resultando en una serie de consecuencias comúnmente denominadas calentamiento global o cambio climático.

El cambio climático es un reto complejo que tiene peso en las disciplinas ecológicas, ambientales, socio-políticas y socio-económicas. Se caracteriza por los cambios a largo plazo en las tendencias de temperatura y precipitaciones, junto con otros componentes como la presión o la humedad en el entorno. Otras consecuencias del cambio climático son los patrones irregulares en el clima y el deshielo de los glaciares y su correspondiente elevación del nivel del mar [1]. La Agencia Internacional de Energía (IEA) y la Comisión Europea (EC) tienen como objetivo la reducción del 80 por ciento de las emisiones globales para el año 2050. Para ello, han identificado la eficiencia energética en el sector inmobiliario como una de las cinco medidas para asegurar la descarbonización a largo plazo [15].

Y en ese contexto, el comprender los patrones de consumo de energía en los edificios es fundamental para las empresas de servicios públicos, usuarios y administradores de instalaciones de cara a mejorar la eficiencia energética. Con la llegada y el desarrollo de nuevas tecnologías, es posible obtener conclusiones de los datos obtenidos durante la fase operacional utilizando técnicas como la Inteligencia Artificial o algoritmos de Machine Learning. Predecir correctamente el consumo de energía puede ser vital para mejorar la eficiencia energética de los edificios y por ende, para reducir las emisiones en el sector inmobiliario [9].

El **objetivo principal de este trabajo es investigar las distintas técnicas matemáticas y estadísticas que pueden ser utilizadas para predecir el consumo de energía en edificios**. Para ello, se analizarán, implementarán y compararán diferentes métodos, desde los más simples hasta los más complejos, con el objetivo de seleccionar aquellos que permitan obtener un modelo predictivo fiable del consumo energético en edificios. La intención es contribuir a la optimización de la eficiencia energética

en estos espacios, lo que permitirá no solo reducir el impacto ambiental reduciendo las emisiones en la fase operacional y de mantenimiento, sino también ahorrar costes energéticos a largo plazo.

Los datos utilizados en este estudio son abiertos y provienen de una recopilación realizada en un edificio en Berkeley, California. El artículo publicado sobre este edificio [19] detalla la recolección de datos sobre el consumo energético total del edificio, así como el consumo fragmentado por zonas y por tipo. Además, incluye información relevante sobre la operabilidad de los sistemas de climatización y ventilación (HVAC), los parámetros ambientales tanto del interior como del exterior del edificio, y la ocupación de los espacios. El periodo de tres años de recolección de datos, desde enero de 2018 hasta diciembre de 2020, cubre tanto el período previo como durante la pandemia, lo que nos ha permitido obtener una visión completa y detallada del consumo energético del edificio.

ORGANIZACIÓN

Esta memoria se organiza de la siguiente manera:

En primer lugar, se realiza una revisión de los fundamentos matemáticos e informáticos esenciales para comprender correctamente el estudio. En ella, se introducen conceptos como los procesos estocásticos, los procesos ARMA o el concepto de prueba estadística. Además, también se exponen los algoritmos de aprendizaje automático principales utilizados a lo largo del trabajo.

Después, se introduce la metodología seguida en el estudio. Tras estudiar los datos concretos que vamos a utilizar, pasamos al preprocesamiento de los datos, incluyendo la descripción del algoritmo de imputación implementado como parte del proyecto. Además, se estudian las propiedades de la variable objetivo considerada como serie temporal así como la interdependencia entre el resto de variables. Más adelante, procedemos a exponer los algoritmos de predicción con los que se ha experimentado.

Por último, presentamos los experimentos principales llevados a cabo como parte del estudio. Comenzamos por los experimentos que ayudaron a definir correctamente la implementación del algoritmo de imputación, y después exponemos el motivo por el cual no se ha podido llevar a cabo una descomposición de la variable objetivo para facilitar las predicciones. Por último, se presentan los experimentos llevados a cabo en cada uno de los algoritmos estudiados, así como en las redes neuronales, y como conclusión comparamos los resultados obtenidos.

TEMPORIZACIÓN Y RECURSOS

La Figura 1 muestra la organización del proyecto mediante un Diagrama de Gantt, con un total de 457 horas dedicadas. Es necesario destacar la simultaneidad de distintas tareas independientes, como por ejemplo la documentación de los conceptos

matemáticos y la implementación y ejecución de los distintos experimentos de predicción. Este trabajo se ha realizado en el contexto del Trabajo de Fin de Grado del Doble Grado en Ingeniería Informática y Matemáticas, con un esfuerzo previsto de 18ECTS.

El coste que habría tenido realizar este proyecto, teniendo en cuenta un precio de 12€/hora, y un horario de media jornada durante 8 meses, sería de 9240€. No se ha incluido el coste del equipamiento utilizado, pues se ha llevado a cabo en un ordenador portátil personal, en particular un MacBook Pro M1 de 16GB de RAM (1800€), con el que ya se contaba antes de comenzar el trabajo.

TAREA	INICIO	FIN	HORAS
Contextualización del proyecto			**10**
Documentación de la eficiencia energética en el sector inmobiliario	3-10-22	8-10-22	2
Documentación de trabajos previos relevantes	10-10-22	16-10-22	4
Documentación sobre aprendizaje automático	12-10-22	18-10-22	4
Carga y procesamiento de datos			**123**
Análisis exploratorio de datos	17-10-22	27-10-22	12
Corrección código de limpieza	1-11-22	21-11-22	20
Estudio de algoritmos de imputación de datos	23-11-22	29-11-22	8
Implementación y ejecución del algoritmo de imputación	28-11-22	5-1-23	25
Experimentos algoritmo de imputación	7-12-22	16-1-23	50
Análisis y selección de las variables	30-1-23	3-2-23	8
Estudio de series temporales			**42**
Documentación sobre las propiedades de series temporales	4-1-23	6-1-23	5
Análisis de las propiedades sobre la variable objetivo	9-1-23	3-2-23	8
Experimentos de descomposición de la variable objetivo	30-1-23	13-2-23	25
Estudio multivariable	20-2-23	22-2-23	4
Predicción con ML			**109**
Estudio y selección de modelos ARIMA y SARIMAX univariable	5-2-23	15-2-23	10
Estudio y selección de modelo Prophet univariable	10-2-23	14-2-23	8
Estudio y selección de modelo UC univariable	5-2-23	10-2-23	5
Estudio y selección de modelo TBATS univariable	11-2-23	13-2-23	5
Estudio de versiones multivariable de los algoritmos	20-2-23	17-3-23	15
Desarrollo y ejecución de experimentos de ventana móvil	20-3-23	22-4-23	60
Análisis de resultados de experimentos de ventana móvil	23-4-23	27-4-23	6
Predicción de series con redes LSTM			**38**
Estudio sobre arquitecturas LSTM	27-2-23	8-3-23	6
Implementación y experimentación con arquitecturas LSTM univariable	7-3-23	8-4-23	18
Experimentación con arquitecturas LSTM multivariable	3-4-23	23-4-23	12
Análisis de resultados	1-5-23	1-5-23	2
Fundamentos matemáticos e informáticos			**64**
Documentación de procesos estocásticos y sus propiedades	20-1-23	4-2-23	17
Documentación de los procesos ARIMA y SARIMA y Prophet	31-1-23	12-2-23	15
Documentación de pruebas estadísticas relevantes	18-2-23	23-2-23	7
Documentación de los fundamentos de las redes neuronales	21-2-23	25-2-23	10
Documentación de los algoritmos de aprendizaje supervisado	27-2-23	14-3-23	13
Documentación de la evaluación de modelos de predicción	8-3-23	9-3-23	2
Descripción de la Metodología			**53**
Documentación de los datos utilizados	23-3-23	4-4-23	10
Documentación del algoritmo de imputación	6-4-23	16-4-23	17
Documentación de las propiedades de series temporales	18-4-23	23-4-23	12
Documentación de los modelos de predicción	26-4-23	3-5-23	14
Descripción de Experimentos			**18**
Documentación de los experimentos realizados	6-5-23	30-5-23	18
Total			**457**

Figura 1: Diagrama de Gantt del proyecto

2

ANTECEDENTES

Es bien conocido que el sector inmobiliario es uno de los principales contribuyentes al cambio climático, y la eficiencia energética de los edificios es una de las áreas con mayor potencial para reducir esta huella. Muchos trabajos previos han demostrado la importancia de mejorar la eficiencia energética en los edificios y han desarrollado diferentes técnicas y herramientas para lograrlo.

Además, las series temporales y el aprendizaje automático han sido ampliamente utilizados en la optimización de la eficiencia energética en los edificios. El aprendizaje supervisado, en particular, ha demostrado ser una técnica efectiva para predecir el consumo energético modelando patrones previamente identificados en los datos históricos.

Este capítulo presentará una revisión detallada de trabajos relevantes en estas áreas, así como una introducción a los conceptos clave de aprendizaje automático y aprendizaje supervisado. Con esta información como base, se establecerá un marco conceptual para el estudio en sí, que busca utilizar técnicas matemáticas y estadísticas para predecir el consumo energético en un edificio.

2.1 TRABAJOS PREVIOS

El primer trabajo que queremos exponer es **Buildings and Climate Change: Summary for Decision Makers** [29], fue publicado en el 2009 y ofrece una revisión exhaustiva sobre cómo se pueden lograr reducciones potenciales de las emisiones de gases de efecto invernadero en edificios. Fue elaborado por la Iniciativa de Clima y Construcción Sostenible (SBCI), una alianza patrocinada por el Programa de las Naciones Unidas para el Medio Ambiente (PNUMA) que reúne a diferentes actores públicos y privados del sector de la construcción, y que persigue promover prácticas sostenibles a nivel global. El trabajo busca ayudar a comprender las mejores prácticas y las tendencias actuales en la reducción de emisiones de gases de efecto invernadero en el sector inmobiliario.

En este estudio se exponen unas estadísticas muy interesantes: el sector inmobiliario es responsable de más del 40 por ciento del uso global de energía y conforma aproxi-

madamente el 30 por ciento de las emisiones totales de gases de efecto invernadero (*Greenhouse Gas*, GHG).

Además, se presentan los mensajes clave actuales sobre la reducción de emisiones de gases de efecto invernadero en el sector inmobiliario. Se destaca que aunque en el pasado los mayores niveles de emisiones provenían de países desarrollados, es probable que en el futuro los países en proceso de rápida industrialización superen esos niveles. También, se resalta la importancia de la tecnología actual para reducir el consumo de energía en edificios nuevos y existentes, lo cual se puede lograr entre un 30 y un 80 por ciento. Por último, se deduce que la reducción de emisiones en el sector inmobiliario tiene también un impacto significativo en la economía y la sociedad, ya que la construcción, renovación y mantenimiento de edificios contribuyen entre un 10 y un 40 por ciento al PIB de los países. Por lo tanto, las estrategias de mitigación de gases pueden ser una fuente de creación de negocios y mejora del alojamiento, lo cual contribuye a mejorar el nivel de vida de la población.

También es necesario destacar **Life cycle assessment of buildings: A review** [27], artículo publicado en el 2011 que expuso la importancia de utilizar el análisis del ciclo de vida de un edificio (Life Cycle Asessment, LCA). El LCA en edificios es una herramienta utilizada para la evaluación cuantitativa de los materiales utilizados, los flujos de energía y los impactos ambientales de un edificio. Es una técnica para evaluar diferentes aspectos relacionados con la construcción y demolición de un edificio, y su posible impacto a lo largo de la vida útil del mismo, lo que en última instancia ayuda a los productores y consumidores a tomar decisiones más informadas y sostenibles.

El LCA se realiza siguiendo las fases del ciclo de vida del edificio, que abarcan desde la extracción de los recursos hasta su eliminación final tras demolerlo. Evalúa los impactos ambientales a lo largo de su vida útil, incluyendo la producción de energía, la emisión de gases de efecto invernadero y otros desechos peligrosos.

En este artículo se menciona que en torno al 80 por ciento de las emisiones de GHG de los edificios se dan durante la fase operacional, es decir, la fase en la que la energía se utiliza para la calefacción, refrigeración, ventilación e iluminación, entre otras aplicaciones. Por esta razón, es fundamental abordar medidas de reducción de emisiones en esta fase, ya que serán más efectivas y tendrán un mayor impacto en la mitigación de las emisiones de gases de efecto invernadero.

El siguiente trabajo que se ha revisado ha sido **A review on time series forecasting techniques for building energy consumption** [6], fue publicado en el 2017 y realiza una revisión de las distintas técnicas de predicción de la demanda de energía en un edificio. Establece que el análisis de los patrones de consumo energético en edificios se puede realizar mediante dos enfoques: la simulación y las técnicas basadas en datos previos. El artículo destaca que aunque en los últimos años se ha avanzado en el desarrollo de software de simulación para el diseño de edificios sostenibles desde un punto de vista energético, es difícil modelar con precisión el consumo energético una vez que el edificio se encuentra en funcionamiento. Esto es debido a que en estos casos el consumo está sujeto a factores externos como las condiciones meteorológicas, la ocupación, propiedades térmicas de los materiales del edificio y las interacciones

entre los distintos sistemas de energía presentes en el edificio, en particular el aire acondicionado y la iluminación [6]. Por esta razón, en los edificios en funcionamiento resulta más conveniente utilizar técnicas basadas en datos previos.

Estas técnicas normalmente incluyen algoritmos de Machine Learning (ML) e Inteligencia Artificial para el análisis y la predicción de series temporales. Una serie temporal es una secuencia ordenada de valores registrados en intervalos constantes de tiempo.

Asimismo, en **Machine learning for estimation of building energy consumption and performance: a review** [26] se explora la manera en que los modelos de predicción pueden tener un impacto positivo en la eficiencia energética de un edificio. Se destaca la utilidad de los modelos de predicción de series temporales como herramientas valiosas para optimizar tanto los Sistemas de Gestión de Energía (EMS) como los Sistemas de Calefacción, Ventilación y Aire Acondicionado (HVAC, por sus siglas en inglés). De esta forma, se puede mejorar la gestión de la energía en el edificio y lograr un uso más eficiente de la misma.

El sistema EMS se utiliza para la generación de los datos y para el control del consumo. Para optimizarlo, se recopila una gran cantidad de datos originaria de los sensores y junto con la información del clima, se busca implementar herramientas analíticas que permitan tanto la evaluación del rendimiento del sistema como la predicción de la demanda en un periodo determinado. Con estas herramientas analíticas se puede implementar la automatización del control de energía y la detección de fallos, y calcular el ahorro en energía obtenido. De la misma manera, el sistema HVAC se puede optimizar mediante la predicción de las cargas en calefacción y refrigeración, pues el cálculo de las mismas supone un gasto en tiempo y en dinero para los administradores.

Es necesario notar que los registros de datos energéticos en el mundo real siempre presentan diferentes tipos de problemas. Uno de los más comunes es la falta de datos [20] que sucede en casi todos los conjuntos de datos. La cantidad de datos ausentes puede afectar significativamente los resultados de investigación. Uno de los factores que más contribuye a esta falta de datos en los registros de energía es la poca fiabilidad de los sensores que recopilan la información. Estos sensores pueden fallar o no estar debidamente calibrados, lo que resulta en datos erróneos o en la falta de los mismos. Además, puede haber problemas en la transmisión de la información desde los sensores hasta los sistemas de registro, lo que también puede resultar en la pérdida de datos. Con la creciente dependencia de los análisis basados en datos en el campo de la energía en edificios, el problema de la fiabilidad de los sensores y falta de datos se vuelve cada vez más crítico en este dominio.

Todos estos trabajos son relevantes para nuestro estudio porque (1) establecen el impacto que tiene el sector inmobiliario en el cambio climático y (2) justifican la importancia de utilizar análisis de series temporales para mejorar la eficiencia energética en este sector.

2.2 INTRODUCCIÓN A MACHINE LEARNING

La Inteligencia Artificial es una ciencia que ha evolucionado a través de la integración de diversas disciplinas, incluyendo filosofía, matemáticas, economía, neurociencia, psicología y ingeniería computacional. Se centra en la creación de sistemas y tecnologías capaces de realizar tareas que normalmente requieren de la inteligencia humana, como la resolución de problemas complejos, la toma de decisiones y la interpretación de información [24]. Aunque todavía no existe un consenso en cuanto a una definición unánime de lo que es la Inteligencia Artificial, es innegable su presencia y relevancia en muchos aspectos de la vida moderna. Desde la automatización de tareas en el transporte y el comercio en línea hasta el procesamiento del lenguaje natural y la toma de decisiones, la inteligencia artificial está revolucionando la forma en que hacemos las cosas en el mundo actual.

Tomemos el ejemplo del reconocimiento facial: esta tarea es algo que los humanos hacemos de manera cotidiana y sin esfuerzo alguno, reconociendo a familiares y amigos en imágenes, incluso si hay diferencias en postura, iluminación, ubicación, etc. Sin embargo, hacemos esto de manera inconsciente y no podemos explicar cómo lo hacemos. Como no podemos explicar nuestra habilidad, es imposible escribir un programa para que lo haga un ordenador. Al mismo tiempo, sabemos que una imagen de una cara no es simplemente una colección aleatoria de píxeles; un rostro tiene una estructura, es simétrico y tiene elementos como los ojos, la nariz y la boca ubicados en lugares específicos. A través del análisis de imágenes de ejemplo de una persona, un modelo de aprendizaje automático puede capturar el patrón específico de esa persona y luego reconocerlo comparando con ese patrón en una imagen dada. Este es un ejemplo de reconocimiento de patrones.

Denominaremos **Machine Learning** a un conjunto de algoritmos y modelos que permiten a las máquinas aprender a partir de datos recopilados y utilizar el conocimiento para realizar tareas específicas sin ser programadas explícitamente para ello.

Estas técnicas son objeto de una gran investigación a nivel mundial debido a su capacidad para procesar grandes cantidades de información y ofrecer resultados precisos y útiles en distintas áreas. El Machine Learning es una parte fundamental de la Inteligencia Artificial, puesto que permite a las máquinas simular procesos cognitivos humanos y adaptarse a entornos cambiantes mediante su capacidad de aprendizaje. Esto significa que el diseñador del sistema no necesita prever y proporcionar soluciones para todas las situaciones posibles, sino que el sistema puede aprender y ajustarse por sí mismo. El aprendizaje automático ayuda a resolver problemas en campos como la visión, el reconocimiento de voz y la robótica [2].

El aprendizaje automático utiliza la teoría estadística en la creación de modelos matemáticos, ya que la tarea principal es inferir a partir de una muestra. El papel de la informática es doble: en primer lugar, durante el entrenamiento, necesitamos algoritmos eficientes para resolver el problema de optimización de lo parámetros del modelo, así como para almacenar y procesar la cantidad masiva de datos que generalmente tenemos. En segundo lugar, una vez que se ha aprendido un modelo, su

representación y la solución algorítmica para la inferencia también deben ser eficientes. Dependiendo del problema, la eficiencia del algoritmo de aprendizaje, es decir, su complejidad en términos de espacio y tiempo, puede ser tan importante como su precisión predictiva.

Vamos a diferenciar tres tipos de técnicas de Machine Learning: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo.

En primer lugar, el **aprendizaje supervisado** se refiere a la técnica de aprendizaje automático en la que se busca obtener un mapeo o una relación entre un conjunto de variables de entrada X y una variable de salida Y [5]. Esta relación se utiliza luego para predecir los valores de la variable de salida para datos no vistos previamente. Una propiedad clave del aprendizaje supervisado es que se proporciona un conjunto de datos de entrenamiento con valores conocidos de X y Y. A partir de estos datos, el algoritmo de aprendizaje supervisado construye un modelo cuyo objetivo es predecir el valor de Y de manera precisa para datos que no se han visto previamente. Más adelante entraremos en detalle en los algoritmos más comunes.

El **aprendizaje no supervisado** es una técnica de inteligencia artificial en la que el modelo se entrena en datos que no tienen etiquetas o una variable objetivo específica de salida. En lugar de aprender la relación entre las variables de entrada y la salida, el objetivo de este tipo de aprendizaje es descubrir patrones o estructuras en los datos. Por ejemplo, el agrupamiento o la partición de los datos en clusters sin la ayuda de etiquetas de clase es un ejemplo de aprendizaje no supervisado. Este tipo de aprendizaje es muy relevante en el procesamiento de contenido multimedia, ya que la agrupación de datos sin etiquetas es una necesidad común.

Hay casos de uso en los que la salida del sistema es una secuencia de acciones. En estas aplicaciones es más importante la secuencia de acciones correctas para alcanzar un objetivo que una acción concreta. A las secuencias de acciones se les denomina políticas y diremos que una acción es óptima si es parte de una buena secuencia. Esto quiere decir que no se pueden valorar como buenas o malas las acciones de algún estado intermedio. En estos caso, el programa de aprendizaje automático debe ser capaz de evaluar las políticas como óptimas o no y generar conocimiento a partir de las secuencias de acciones buenas del pasado. Al final de cada secuencia se obtiene una recompensa o penalización en función de si ésta ha sido buena o no, que hará que el agente cambie su comportamiento y produzca mejores políticas. Estos métodos de aprendizaje se llaman métodos de **aprendizaje por refuerzo** debido a que el conocimiento se genera mediante un proceso de prueba y error.

En este trabajo fin de grado, nos centraremos principalmente en técnicas de aprendizaje supervisado, por lo que a continuación solo abordaremos los algoritmos principales de este paradigma.

2.2.1 *Aprendizaje Supervisado*

Tal y como ya hemos comentado, el aprendizaje supervisado es una tarea de aprendizaje automático que consiste en aprender una función que mapea una entrada a una salida basada en pares de entrada-salida de ejemplo. Los algoritmos de aprendizaje automático supervisados son aquellos que necesitan asistencia externa, o dicho de otra manera, necesitan aprender de unos datos que incluyan tanto unas variables de entrada (o características) como una variable de salida concreta. Para ello, el conjunto de datos de entrada se suele dividir en un conjunto de datos de entrenamiento y un conjunto de datos de prueba. Todos los algoritmos aprenden algún tipo de patrones a partir del conjunto de datos de entrenamiento y los aplican al conjunto de datos de prueba para realizar una predicción del valor de la variable de salida. De esta manera, podemos comparar la predicción obtenida por el modelo con el valor real de la variable de salida en el conjunto de prueba, y obtener una métrica del rendimiento de modelo.

Para continuar es necesario exponer un concepto clave en las técnicas de aprendizaje supervisado: el compromiso (o equilibrio) entre sesgo y varianza. Esta propiedad se refiere a la relación entre la capacidad de un modelo para ajustarse bien a los datos de entrenamiento y su capacidad para generalizar esos aprendizajes a nuevos datos.

El **sesgo** o bias en un modelo es la diferencia entre el valor predicho y el valor real. Un modelo con un sesgo alto tendrá una predicción prácticamente constante y será demasiado simple, lo que lo convierte en un modelo poco preciso. Por otro lado, la **varianza** se refiere a la variabilidad o fluctuación en las predicciones realizadas por un modelo según los datos de entrenamiento que se utilicen. Aunque es común que los modelos de aprendizaje supervisado tengan cierta varianza, un algoritmo robusto debería ser capaz de modelar correctamente las relaciones entre las características y la variable objetivo. Así, las predicciones de un modelo con una varianza alta tendrá una serán muy sensibles a los cambios en los datos de entrenamiento y, por lo tanto, posiblemente no generalizarán bien a nuevos datos.

La relación entre ambas es inversa, es decir, un modelo con un sesgo alto tendrá una varianza baja (*underfitting*), pues sus predicciones son más consistentes y no dependen tanto del conjunto de entrenamiento. De la misma manera, un modelo con una varianza alta tendrá un sesgo bajo (*overfitting*). El objetivo es encontrar un equilibrio entre ambos para obtener un modelo preciso tanto en los datos de entrenamiento como en los nuevos datos. En el siguiente capítulo explicaremos en detalle los conceptos matemáticos que hay detrás de esta propiedad.

Los algoritmos de aprendizaje supervisado se pueden clasificar en algoritmos de clasificación y algoritmos de regresión, dependiendo de si la variable de salida es discreta con pocos valores o continua, respectivamente. Los algoritmos más comunes son: Regresión Lineal, Árbol de Decisión, Algoritmos Bayesianos y Support Vector Machine [21].

El algoritmo más básico, el de **Regresión Lineal**, intenta modelar la relación entre varias variables ajustando una ecuación lineal a los datos observados. Consideramos varias variables como variables explicativas y una única como una variable dependiente. Cuando hay más de una variable explicativa nos encontramos ante un modelo de regresión lineal múltiple, mientras que cuando hay solo una hablaremos de la regresión lineal simple. El método más común para ajustar una línea de regresión es el método de los mínimos cuadrados. Este método calcula la línea de ajuste más adecuada para los datos observados minimizando la suma de los cuadrados de los residuos, esto es, minimizando la suma de los cuadrados de las diferencias entre los valores reales y los estimados por la recta. Debido a que las desviaciones se elevan primero al cuadrado y luego se suman, no hay cancelaciones entre valores positivos y negativos.

El siguiente algoritmo, el **Árbol de Decisión**, permite tomar decisiones basadas en una serie de reglas. Utiliza una herramienta visual que se presenta en forma de un árbol, formado por nodos y ramas. Los nodos representan eventos o características de un problema y las ramas representan las posibles decisiones o acciones. Cada nodo es una pregunta sobre los datos y las ramas son las respuestas a esa pregunta, las cuales llevan a otro nodo. El objetivo del algoritmo es construir un árbol que permita predecir una variable objetivo a partir de un conjunto de variables de entrada, por tanto este es un algoritmo de clasificación. Esto se logra mediante la selección de las variables más importantes y la separación de los datos en subgrupos más pequeños en cada nodo hasta que los subgrupos sean lo suficientemente homogéneos como para tomar una decisión precisa.

Los **Algoritmos Bayesianos** son un conjunto de algoritmos de clasificación que se basan en el Teorema de Bayes, asumiendo independencia en los predictores, es decir, se asume que la presencia de una característica concreta en una clase no está relacionada con la presencia de cualquier otra característica. Para clasificar un objeto en las diferentes categorías se calcula tanto su probabilidad a priori como la probabilidad condicional de que pertenezca a una categoría dada sus características. La probabilidad a priori se puede estimar a partir de los datos de entrenamiento y la probabilidad condicional se estima a partir de la relación entre las características y la categoría. Luego, se combinan estas probabilidades para calcular la probabilidad de que un objeto dado pertenezca a cada categoría, y se asigna al objeto a la categoría con la probabilidad más alta. Estos algoritmos incluyen Naive Bayes, Gaussian Bayes, Multinomial Bayes, dependiendo de qué distribución de probabilidad se utilice para calcular las probabilidades.

Por último, **Support Vector Machines (SVMs)** es un algoritmo de aprendizaje supervisado que se utiliza para clasificar datos. La idea detrás de SVMs es encontrar la línea de separación que mejor divide los datos en sus respectivas categorías. Esta línea de separación es llamada hiperplano y es elegida de tal manera que maximiza la distancia entre los puntos más cercanos de cada categoría, estos puntos son llamados vectores de soporte. De este modo, los SVMs son capaces de clasificar nuevos datos basándose en la similitud a los vectores de soporte utilizados en el entrena-

miento. Este algoritmo se utiliza para manejar datos que tienen un gran número de características y para problemas de clasificación binaria.

Los algoritmos que acabamos de introducir son los más utilizados a día de hoy, sin embargo, a medida que los datos se han vuelto más complejos y de mayor escala, ha surgido la necesidad de utilizar técnicas más avanzadas. Es aquí donde entra en juego el aprendizaje profundo, que utiliza **redes neuronales** artificiales para imitar la forma en que el cerebro humano procesa la información. Las redes neuronales han demostrado ser altamente efectivas en aplicaciones como la visión por computadora, el procesamiento del lenguaje natural y la generación de música o imágenes.

Una red neuronal está compuesta por capas de neuronas o unidades que procesan y transmiten información a través de conexiones. Cada unidad recibe como entrada la salida de las unidades de la capa anterior, la procesa mediante una función de activación no lineal y transmite su salida a las neuronas en la capa siguiente. El proceso se repite para cada capa de la red, permitiendo que la red aprenda patrones y relaciones en los datos de entrada y realice predicciones o clasificaciones en función de los patrones aprendidos. Además, existen infinidad de arquitecturas que se pueden utilizar en función de los datos, lo que convierte las redes neuronales en una de las soluciones más comunes para solucionar problemas de predicción en series temporales complejas. Más adelante veremos cómo funciona el aprendizaje en este tipo de arquitecturas.

En la siguiente sección, se explorarán los fundamentos matemáticos esenciales para el análisis de series temporales, incluyendo una introducción breve a los procesos estocásticos, y las técnicas de predicción de series temporales. También se analizarán las pruebas estadísticas que se pueden aplicar a series temporales para evaluar distintas propiedades y se introducirán las redes neuronales en detalle. Además, se discutirán en detalle alguno de los algoritmos de aprendizaje supervisado que se utilizan comúnmente en el análisis de series temporales, y se estudiarán las técnicas de ajuste de algoritmos. Finalmente, se abordará la evaluación de modelos, que permitirán comparar los experimentos y obtener el más adecuado para los datos.

Parte II

FUNDAMENTOS DE MATEMÁTICAS E INFORMÁTICA

Definición de series temporales y procesos estocásticos. Exposición de los algoritmos principales utilizados.

3

FUNDAMENTOS DE MATEMÁTICAS

En este capítulo, nos adentraremos en los fundamentos matemáticos necesarios para entender y aplicar herramientas de análisis y predicción en series temporales. En particular, estudiaremos en detalle la descomposición de series temporales e introduciremos los procesos estocásticos, los cuales son fundamentales en el análisis de series y la predicción en contextos de incertidumbre. También estudiaremos sus propiedades principales, y en particular, los procesos estocásticos estacionarios. Después nos adentraremos en los modelos de predicción utilizados en la experimentación, y explicaremos en qué consiste realizar una prueba estadística para comprobar una propiedad de una serie temporal y describiremos formalmente las dos que hemos utilizado en el análisis posterior. Para terminar, introduciremos los fundamentos matemáticos en los que se basan las redes neuronales. Estos conceptos son clave para poder entender y aplicar técnicas avanzadas de análisis y predicción en el campo de la ciencia de datos.

3.1 DESCOMPOSICIÓN DE UNA SERIE TEMPORAL

Una serie temporal es una secuencia ordenada de observaciones registradas en intervalos de tiempo, normalmente constantes. En esta sección consideraremos una serie temporal como una función dependiente del tiempo, es decir, y_t, donde t es el tiempo.

La descomposición matemática usual determina que una serie temporal se puede dividir en cuatro componentes, definidas a continuación:

- La tendencia (T_t) refleja el progreso de la serie a largo plazo. No tiene por qué ser lineal, y puede ser a la baja, a la alza o invariable. Según el tipo de tendencia, se puede modelar por una recta o con alguna otra curva, como por ejemplo una variante de la exponencial.
- La ciclicidad (C_t) representa variaciones repetidas pero no periódicas, es decir, que no tienen una longitud en tiempo constante. Estas variaciones normalmente se deben a factores externos impredecibles, como por ejemplo una acción humana.

- La estacionalidad (S_t) incluye las variaciones periódicas de longitud constante. Las causas de estas variaciones también son periódicas, como por ejemplo la temperatura, el día de la semana, la hora del día o períodos de vacaciones.
- El ruido o residuo (R_t) abarca todas las variaciones debidas a influencias completamente aleatorias.

Con la finalidad de determinar el tipo de modelo que debemos utilizar es fundamental establecer la estructura subyacente de la serie temporal y para ello usamos un proceso estadístico llamado descomposición. Al descomponer una serie temporal, podemos predecir de manera separada cada una de ellas y ajustarnos mejor a las propiedades particulares de cada componente, lo que nos permite hacer predicciones más precisas.

Existen distintos métodos para descomponer una serie temporal y todos ellos tienen un objetivo común: aislar cada componente de la serie temporal con la mayor precisión posible. La mayoría de los enfoques de descomposición consideran los componentes de tendencia y ciclo como un único componente determinado tendencia-ciclo (T_t).

El proceso de descomposición es empírico y consiste en retirar primero la componente tendencia-ciclo y más adelante separar la componente de estacionalidad. La componente de ruido se asume como aleatoria, y como tal no se puede prever, pero sí identificar.

La representación matemática general de cualquier método de descomposición es la siguiente:

$$y_t = f(T_t, S_t, R_t)$$

donde y_t es el valor de la serie temporal en el tiempo t, T_t es la componente tendencia-ciclo en el tiempo t, S_t es la componente de estacionalidad en el tiempo t, y R_t es la componente residual en el tiempo t.

Los dos enfoques principales de descomposición son la descomposición aditiva y la descomposición multiplicativa.

El modelo aditivo establece la serie temporal como una suma de las tres componentes, es decir:

$$y_t = T_t + S_t + R_t$$

Y el modelo multiplicativo considera el producto de las componentes:

$$y_t = T_t * S_t * R_t$$

También podemos obtener otros enfoques realizando una combinación entre el aditivo y el multiplicativo, sin embargo, la descomposición presentada puede no ser la correcta para todas las series temporales, tal y como veremos más adelante. Esto es debido a que pueden existir ciertos componentes o tendencias en la serie no consideradas por la descomposición usual.

Una vez que hemos comprendido las diferentes componentes de las series temporales y cómo éstas influyen en su comportamiento, es importante entender cómo se generan estas series. Para ello, es necesario introducir los procesos estocásticos, elementos matemáticos que facilitan el análisis del comportamiento aleatorio de una serie temporal. Al analizar los procesos estocásticos en el contexto de las series temporales, podremos comprender mejor la variabilidad y las fluctuaciones que observamos en los datos. Además, los procesos estocásticos nos permitirán modelar y predecir el comportamiento futuro de las series temporales, objetivo principal de este trabajo.

3.2 DEFINICIÓN Y PROPIEDADES DE LOS PROCESOS ESTOCÁSTICOS

Comenzaremos la sección introduciendo la relación entre las series temporales y los procesos estocásticos, y terminaremos mediante el estudio de la estacionariedad, una de las propiedades fundamentales de los procesos estocásticos.

El primer paso en el análisis de series temporales es establecer un contexto matemático adecuado. Para tratar con la naturaleza impredecible de las futuras observaciones de una serie temporal, podemos considerar cada observación de la serie y_t como una realización de una variable aleatoria Y_t, donde una realización es el resultado de una repetición de esa variable. Así, en la variable aleatoria *lanzar una vez un dado* una realización sería el número que nos haya salido en ese único lanzamiento. Es necesario notar que para la redacción de esta sección nos hemos basado principalmente en el libro **Time Series: Theory and Methods** [4].

3.2.1 *Definición de procesos estocásticos*

En primer lugar vamos a definir dos conceptos clave.

Definición 3.2.1. (Espacio Probabilístico) Llamamos espacio probabilístico a la terna formada por un espacio muestral, Ω, el álgebra de sucesos $\mathcal{F}$, y una función de probabilidad P, es decir a $(\Omega, \mathcal{F}, P)$.

Definición 3.2.2. (Variable Aleatoria) Sea $(\Omega, \mathcal{F}, P)$ un espacio probabilístico. Una variable aleatoria es una función medible:

$$\begin{aligned} X\colon \Omega &\to E \\ t &\mapsto X_t. \end{aligned}$$

donde E es un espacio medible.

En este estudio vamos a limitarnos a las variables aleatorias con valores reales, pues son lo que verifican las series temporales. Por tanto, consideraremos una variable aleatoria como una función

$$\begin{aligned} X\colon \Omega &\to \mathbb{R} \\ t &\mapsto X_t. \end{aligned}$$

Definición 3.2.3. (Proceso Estocástico) Un proceso estocástico es un conjunto de variables aleatorias $\{Y_t, t \in T\}$ definidas en un espacio de probabilidad $(\Omega, \mathcal{F}, P)$.

Podemos considerar una serie temporal $\{y_t, t \in T_0\}$ como las realizaciones de un conjunto de variables aleatorias $\{Y_t, t \in T_0\}$. La intuición nos puede guiar hacia el modelado de los datos como la realización de un proceso estocástico $\{Y_t, t \in T\}$, donde $T_0 \subseteq T$.

Es importante notar que para cada $t \in T$ fijo, la función $Y_t(\cdot)$ es una función en Ω. Por otra parte, para cada $\omega \in \Omega$ fijo, la función $Y_\cdot(\omega)$ es una función sobre el conjunto T. De esta manera, la realización de un proceso estocástico $\{Y_t, t \in T\}$ es el conjunto de funciones $\{Y_\cdot(\omega), \omega \in \Omega\}$.

Igual que teníamos una función de distribución con una variable aleatoria, es necesario definir la función de distribución de un proceso estocástico.

Definición 3.2.4. (Funciones de Distribución de un Proceso Estocástico) Sea $\mathcal{T}$ el conjunto de todos los vectores $\{t = (t_1, ...t_n)' \in T^n\colon t_1 < t_2 < ... < t_n, n = 1, 2, ...\}$. Entonces el conjunto de funciones de distribución del proceso estocástico $\{Y_t, t \in T\}$ es el conjunto de funciones $\{F_t(\cdot), t \in \mathcal{T}\}$ definidas para cada $t = (t_1, ...t_n)'$ como

$$F_t(y) = P(Y_{t_1} \leq y_1, ..., Y_{t_n} \leq y_n), \qquad y = (y_1, ...y_n)' \in \mathbb{R} \tag{3.2.1}$$

En el estudio de series temporales, el conjunto índice T suele ser un conjunto de puntos en el tiempo, normalmente: $\{0, \pm 1, \pm 2, ...\}$, $\{1, 2, 3, ...\}$, $[0, \infty)$ o $(-\infty, \infty)$. De ahora en adelante consideraremos los procesos estocásticos con esta estructura.

3.2.2 *Estacionariedad*

Una de las propiedades fundamentales de las series temporales, y por tanto de los procesos estocásticos, es la estacionariedad. La estacionariedad de una serie temporal indica que las propiedades estadísticas de la serie no cambian con el tiempo, es decir, la media y varianza de los datos en diferentes periodos de tiempo son constantes. En otras palabras, aunque la serie temporal puede tener cambios y variaciones a lo largo del tiempo, estos cambios siguen una misma tendencia o patrón que se mantiene constante a lo largo de toda la serie.

A continuación lo vemos de una manera más formal, pero antes es necesario introducir un concepto nuevo: la función de autocovarianza. La autocovarianza es un concep-

to estadístico que permite medir la covarianza de un proceso estocástico en diferentes puntos del conjunto T, y por tanto, en distintos puntos de la línea temporal.

Definición 3.2.5. (Función de Autocovarianza) Si $\{Y_t, t \in T\}$ es un proceso estocástico tal que $Var(Y_t) < \infty$ para cada $t \in T$, entonces la función de autocovarianza $\gamma(\cdot, \cdot)$ de $\{Y_t\}$ está definida por:

$$\gamma_Y(r, s) = Cov(Y_r, Y_s) = \mathbb{E}[(Y_r - \mathbb{E}Y_r)(Y_s - \mathbb{E}Y_s)], \qquad r, s \in T \tag{3.2.2}$$

Ya estamos en condiciones de presentar la definición formal de estacionariedad:

Definición 3.2.6. (Estacionariedad) Se dice que la serie temporal $\{Y_t, t \in T\}$ es estacionaria si

$$\begin{aligned} &(i) \quad \mathbb{E}|Y_t|^2 < \infty \quad \forall t \in T \\ &(ii) \quad \mathbb{E}Y_t = m \quad \forall t \in T \\ &(iii) \quad \gamma_Y(r, s) = \gamma_Y(r + t, s + t) \quad \forall r, s, t \in T \end{aligned}$$

Comentario 1. Si el proceso estocástico $\{Y_t, t \in T\}$ es estacionario, se verifica que $\gamma_Y(r, s) = \gamma_Y(r - s, 0)$ para todo $r, s \in T$. Por tanto, podemos redefinir la función de autocovarianza como una función de una sola variable

$$\gamma_Y(h) \equiv \gamma_Y(h, 0) = Cov(Y_{t+h}, Y_t) \quad \forall t, h \in T \tag{3.2.3}$$

La propiedad que acabamos de definir se puede restringir aún más. Para ello, tenemos que definir un concepto nuevo, la distribución de probabilidad conjunta:

Definición 3.2.7. (Distribución de probabilidad conjunta) Sean $X_1, ..., X_n$ variables aleatorias definidas en un espacio de probabilidad $(\Omega, \mathcal{F}, P)$. La distribución conjunta de esas variables aleatorias es la función

$$P_{X_1,...X_n}(B_1, ..., B_n) = P\{X_1 \in B_1, ..., X_n \in B_n\} \quad B_i \in \mathbb{R} \forall i = 1, ..., n \tag{3.2.4}$$

Al tratarse de variables aleatorias reales, podemos observar que su distribución conjunta no es más que la distribución del vector aleatorio $(X_1, ..., X_n)$.

Definición 3.2.8. (Distribuciones finito-dimensionales de un proceso estocástico) Sea $\{Y_t, t \in T\}$ un proceso estocástico, entonces las distribuciones conjuntas de $Y_{t_1}, ..., Y_{t_n}$, con $t_i \in T, \forall i = 1, ..., n$ son las distribuciones finito-dimensionales del proceso.

Además de la distribución conjunta, se habla de la función de distribución conjunta y de la función de densidad conjunta, no entraremos más en detalle pues no es el tema principal de esta sección.

Podemos restringir la propiedad de estacionariedad con el concepto de estacionariedad estricta. La estacionariedad estricta significa intuitivamente que los gráficos de la serie temporal a lo largo de dos intervalos de tiempo de igual longitud deben exhibir características estadísticas similares.

Definición 3.2.9. (Estacionariedad estricta) La serie temporal $\{Y_t, t \in T\}$ es estrictamente estacionaria si las distribuciones conjuntas de $(Y_{t_1}, ..., Y_{t_k})$ y de $(Y_{t_1+h}, ..., Y_{t_k+h})$ son las mismas para cualquier entero positivo k y para todo $t_1, ...t_k \in T$ y $h \in \mathbb{Z}$.

A continuación vemos un resultado muy interesante:

Proposición 3.2.1. *Si un proceso estocástico $\{Y_t, t \in T\}$ es estrictamente estacionario, y verifica que $\mathbb{E}|X_t|^2 < \infty$, entonces es estacionario.*

Demostración. Si tomamos $k = 1$ en la definición (3.2.9), obtenemos que las distribuciones de Y_t son idénticas para todo $t \in T$. Si $\mathbb{E}|Y_t|^2 < \infty$, esto implica que $\mathbb{E}Y_t$ y $Var(Y_t)$ son constantes.

De la misma manera, si tomamos $k = 2$ en la definición (3.2.9), deducimos que (Y_{t_1}, Y_{t_2}) y (Y_{t_1+h}, Y_{t_2+h}) tienen la misma distribución conjunta para cualquier $t_1, t_2 \in T$ y $h \in \mathbb{Z}$.

Esto quiere decir que tienen la misma covarianza. Por lo tanto, se verifican las tres condiciones de estacionariedad y podemos decir que el proceso es estacionario. □

Sin embargo, la implicación contraria no se verifica, como contraejemplo podemos estudiar la secuencia de variables aleatorias independientes X_t, donde cada una sigue una distribución exponencial con media 1 si t es impar, y una distribución normal con media 1 y varianza 1 cuando t es par. Observamos que X_t es estacionario con $\gamma_X(0) = 1$ y $\gamma_X(h) = 0$ si $h \neq 0$. Sin embargo, como X_1 y X_2 tienen distribuciones distintas, no puede ser estrictamente estacionario.

Los procesos estocásticos estacionarios son fundamentales en el análisis de series temporales. Sin embargo, muchas series temporales observadas son claramente no estacionarias. En estos casos la teoría de procesos estocásticos estacionarios se puede utilizar de distintas maneras distintas, y nosotros nos centraremos en dos de ellas. La primera alternativa se basa en la descomposición de la serie temporal en tendencia, estacionalidad y residuo. Se intenta estimar y extraer los componentes determinísticos de tendencia y estacionalidad, con el objetivo de que la componente residual sea un proceso estocástico estacionario, y por tanto sea modelable para realizar las predicciones oportunas.

La segunda utiliza una transformación sobre la serie completa. Esta transformación puede ser aplicar una función logarítmica para que se estabilicen tanto la media como la varianza, o puede ser diferenciando la serie temporal. Para diferenciar una serie temporal se calculan las diferencias entre observaciones. La diferenciación tiene como resultado la estabilización de la media y la varianza, así como la reducción o eliminación de los componentes de tendencia y estacionalidad.

3.2.3 *Diferenciación*

A continuación exponemos el proceso de diferenciación, pues es uno de los que vamos a usar más adelante para convertir la serie temporal en una serie estacionaria. Tal y como hemos mencionado, este método consiste en calcular la diferencia entre distintas observaciones, y puede ser de dos tipos:

- Diferenciación simple, donde se calcula la diferencia entre dos observaciones contiguas.
- Diferenciación estacional, en la que se calcula la diferencia entre un valor y su valor estacional anterior, es decir, la diferencia entre observaciones con una distancia s, que normalmente equivale al periodo de estacionalidad de la serie.

Cuando se tienen que aplicar los dos tipos de diferenciación, no es relevante cuál se hace primero. Sin embargo, si los datos tienen una componente estacional muy fuerte, es recomendable empezar por la diferenciación estacional, pues es la que asegura que la serie temporal resultante es estacionaria. A continuación lo comprobamos formalmente.

Suponemos que tenemos una serie temporal X_t, que se puede descomponer mediante el modelo aditivo, es decir, se puede expresar como

$$X_t = T_t + S_t + R_t \tag{3.2.5}$$

donde T_t es la componente de tendencia-ciclo, S_t es la componente de estacionalidad con periodo d y R_t es la componente residual.

Definimos el operador de diferenciación Δ_d:

$$\Delta_d X_t = X_t - X_{t-d} = (1 - B^d)X_t \tag{3.2.6}$$

donde B es otro operador definido por $B^d X_t = X_{t-d}$.

Las potencias de este operador se definen de una manera intuitiva, esto es,

$$\Delta_d^j(X_t) = \Delta_d(\Delta_d^{j-1}(X_t)), \quad j \geq 1$$

Si aplicamos el operador Δ_d sobre el modelo $X_t = T_t + S_t + R_t$ obtenemos

$$\Delta_d X_t = T_t - T_{t-d} + S_t - S_{t-d} + R_t - R_{t-d} \qquad = T_t - T_{t-d} + R_t - R_{t-d} \tag{3.2.7}$$

Observamos que hemos obtenido una descomposición aditiva de la serie temporal en tendencia y residuo, sin la componente de estacionalidad, que se anula por ser periódica.

Volvemos a usar el operador Δ_d con $d = 1$:

$$\Delta X_t = X_t - X_{t-1} = (1 - B)X_t \tag{3.2.8}$$

Podemos observar que al aplicar el operador Δ sobre una función lineal $T_t = at + b$ obtenemos una constante, esto es $\Delta T_t = a$.

De esta manera, una componente tendencia-ciclo que se pueda modelar con un polinomio de grado k se puede reducir a una constante mediante la aplicación de la potencia k del operador Δ, es decir, Δ_k.

Por lo tanto, si partimos del modelo obtenido en la expresión 3.2.7, con

$$T_t - T_{t-d} = \sum_{j=0}^{k} a_j t^j$$

, obtenemos:

$$\Delta^k X_t = k! a_k + \Delta_k (R_t - R_{t-d}) \tag{3.2.9}$$

Si además suponemos que la componente residual es un proceso estacionario con media cero, podemos afirmar que X_t es un proceso estocástico estacionario con media $k! a_k$.

Esto último prueba que dada una serie temporal X_t, si aplicamos tanto el operador Δ_d como Δ de manera iterativa, obtendremos una serie $\Delta^k X_t$ que se puede modelar como la realización de un proceso estocástico estacionario.

3.3 TÉCNICAS DE PREDICCIÓN DE SERIES TEMPORALES

En esta sección vamos a describir los modelos principales utilizados a la hora de hacer predicciones. Para poder explicar formalmente en qué consisten es necesario introducir varios conceptos previos, que suponen la continuación natural de la teoría de procesos estocásticos de la sección anterior. Más adelante describiremos los modelos ARIMA y sus variantes, y terminaremos con la descripción de Prophet.

3.3.1 *Conceptos previos*

Comenzaremos introduciendo la versión más simple de los procesos estocásticos estacionarios: los procesos de ruido blanco. Son los procesos que están formados por una secuencia de variables aleatorias no correladas entre sí, con media cero y varianza finita. Idealmente la componente residual resultante de la descomposición de la serie temporal es un proceso de ruido blanco.

Definición 3.3.1. (Proceso de Ruido Blanco) Diremos que el proceso estocástico $\{Z_t\}$ es ruido blanco con media 0 y varianza σ^2, notado por

$$\{Z_t\} \sim WN(0, \sigma^2)$$

si y sólo si $\{Z_t\}$ tiene media 0 y su función de autocovarianza es

$$\gamma(h) = \begin{cases} \sigma^2, & \text{si } h = 0 \\ 0, & \text{si } h \neq 0 \end{cases}$$

Seguidamente vamos a introducir los procesos estocásticos $ARMA(p,q)$. Estos tipos de procesos dan lugar a los modelos $ARMA(p,q)$ y a sus variantes, modelos que contienen parámetros ajustables a los datos de entrenamiento, normalmente mediante el método de los mínimos cuadrados, explicado en detalle más adelante. El interés detrás de estos modelos radica en que una vez se han ajustado los parámetros, la expresión del modelo se puede utilizar para realizar predicciones.

Definición 3.3.2. (Proceso $ARMA(p,q)$) El proceso estocástico $\{Y_t, t = 0, \pm 1, \pm 2...\}$ es un proceso ARMA(p,q) si $\{Y_t\}$ es estacionario y para cada t se verifica

$$\phi(B)(Y_t - \mu) = \theta(B)Z_t \tag{3.3.1}$$

donde $\{Z_t\} \sim WN(0, \sigma^2)$, μ es la media de $\{Y_t\}$, ϕ y θ son polinomios de grado p y q, respectivamente:

$$\phi(z) = 1 - \phi_1 z - ... - \phi_p z^p \tag{3.3.2}$$

$$\theta(z) = 1 + \theta_1 z - ... + \theta_q z^q \tag{3.3.3}$$

y B es el operador definido por $BX_t = X_{t-1}$.

El polinomio ϕ se denomina polinomio autorregresivo (AR) y trata de capturar la dependencia del valor actual de la serie en los valores pasados. El polinomio θ es el polinomio de media móvil (MA), y es el encargado de capturar la dependencia del valor actual con los errores o residuos pasados, que conforman el ruido blanco considerado en 3.3.1.

Una manera alternativa de definir un proceso $ARMA(p,q)$ es mediante la siguiente expresión:

$$Y_t = \mu + \sum_{k=1}^{p} \phi_k Y_{t-k} + Z_t + \sum_{i=1}^{q} \theta_i Z_{t-i} \tag{3.3.4}$$

De ahora en adelante consideraremos que los polinomios AR y MA no tienen factores en común. Esto no supone una pérdida de generalidad pues de tener factores en común, se cancelarían en la expresión 3.3.1 y obtendríamos un proceso de grado inferior sin factores comunes. Junto con los procesos $ARMA(p,q)$ podemos definir los procesos $AR(p)$ y$MA(q)$, que simplemente son los equivalentes a $ARMA(p,0)$ y $ARMA(0,q)$, respectivamente.

A continuación vamos a profundizar en los procesos $AR(p)$, pues su estudio nos permite establecer una forma empírica de comprobar la estacionariedad de un proceso $ARMA(p,q)$ arbitrario.

Proposición 3.3.1. *Un proceso $AR(1)$ es estacionario si las raíces $z_1, ...z_p \in \mathbb{C}$ de su polinomio característico verifican $|z_i| > 1 \quad \forall i = 1, ..., p$.*

Demostración. Vamos a realizar la demostración del caso base $AR(1)$, pues para realizar la demostración del caso general es necesario precisar de conocimientos muy avanzados de los procesos estocásticos, y no es el objeto principal de este estudio.

Sea Y_t un proceso $AR(1)$ arbitrario:

$$Y_t = \mu + \phi_1 Y_{t-1} + Z_t \tag{3.3.5}$$

donde $\{Z_t\}$ es un proceso de ruido blanco y μ, ϕ_1 son los parámetros del proceso.

Vamos a expresar Y_t en función de Y_{t-2} y Z_{t-1}

$$Y_t = \mu + \phi_1(\mu + \phi_1 Y_{t-1} + Z_{t-1}) + Z_t = \mu + \mu\phi_1 + \phi_1^2 Y_{t-2} + Z_t + \phi_1 Zt - 1 \tag{3.3.6}$$

Si repetimos este proceso t veces, obtenemos

$$\begin{aligned} Y_t &= \phi_1^t Y_{t_0} + \mu(1 + \phi_1 + \phi_1^2 + ...) + Z_t + \phi_1 Z_{t-1} + \phi_1^2 Z_{t-2} \\ &= \phi_1^t Y_{t_0} + \mu \sum_{i=0}^{t-1} \phi_1^i + \sum_{i=0}^{t-1} \phi_1^i Z_{t-i} \end{aligned}$$

A continuación vamos a calcular la esperanza, la varianza y la covarianza y comprobaremos qué condiciones deben cumplir para asegurar la estacionariedad del proceso.

Empezaremos por la esperanza:

$$\begin{aligned} \mathbb{E}Y_t &= \mathbb{E}[\phi_1^t Y_{t_0} + \mu \sum_{i=0}^{t-1} \phi_1^i + \sum_{i=0}^{t-1} \phi_1^i Z_{t-i}] \\ &= \mathbb{E}[\phi_1^t Y_{t_0}] + \mathbb{E}[\mu \sum_{i=0}^{t-1} \phi_1^i] + \mathbb{E}[\sum_{i=0}^{t-1} \phi_1^i Z_{t-i}] \\ &= \phi_1^t \mathbb{E}Y_0 + \mu \sum_{i=0}^{t-1} \phi_1^i \end{aligned}$$

La varianza es:

$$Var(Y_t) = \mathbb{E}\Big[Y_t - \mathbb{E}(Y_t)\Big]^2$$

$$
\begin{aligned}
&= \mathbb{E}\Big[Y_t - (\phi_1^t \mathbb{E}Y_0 + \mu \sum_{i=0}^{t-1} \phi_1^i)\Big]^2 \\
&= \mathbb{E}\Big[\phi_1^t Y_0 + \sum_{i=0}^{t-1} \phi_1^i Z_{t-i} - \phi_1^t \mathbb{E}Y_0\Big]^2 \\
&= \mathbb{E}\Big[\phi_1^t (Y_0 - \mathbb{E}[Y_0]) + \sum_{i=0}^{t-1} \phi_1^i Z_{t-i}\Big]^2 \\
&= \phi_1^{2t} \mathbb{E}\Big[Y_0 - \mathbb{E}[Y_0]\Big]^2 + 2\phi_1^t (Y_0 - \mathbb{E}[Y_0]) \sum_{i=0}^{t-1} \phi_1^i \mathbb{E}[Z_{t-i}] + \mathbb{E}\Big[\sum_{i=0}^{t-1} \phi_1^i Z_{t-i}\Big]^2 \\
&= \phi_1^{2t} Var(Y_0) + \sum_{i=0}^{t-1} \phi_1^{2i} Var(Z_{t-i}) \\
&= \phi_1^{2t} Var(Y_0) + \sum_{i=0}^{t-1} \phi_1^{2i} \sigma^2 \\
&= \phi_1^{2t} Var(Y_0) + \sigma^2 \sum_{i=0}^{t-1} \phi_1^{2i} \\
&= \begin{cases} \phi_1^{2t} Var(Y_0) + \sigma^2 \dfrac{1 - \phi_1^{2t}}{1 - \phi_1^2} & \|\phi_1\| \neq 1 \\ Var(Y_0) + \sigma^2 t & \|\phi_1\| = 1 \end{cases}
\end{aligned}
$$

Para poder obtener la última expresión, ha sido necesario utilizar la incorrelación del ruido blanco tanto temporal como con el proceso Y_t.

Por último, calculamos la autocovarianza:

$$
\begin{aligned}
Cov(Y_t, Y_{t+k}) &= \mathbb{E}\Big[[Y_t - \mathbb{E}Y_t][Y_{t+k} - \mathbb{E}Y_{t+k}]\Big] \\
&= \mathbb{E}\Big[[Y_t - \mathbb{E}Y_t][Y_{t+k} - \mathbb{E}Y_{t+k}]\Big] \\
&= \mathbb{E}\Big[\Big[\phi_1^t (Y_0 - \mathbb{E}[Y_0]) + \sum_{i=0}^{t-1} \phi_1^i Z_{t-i}\Big]\Big[\phi_1^{t+k} (Y_0 - \mathbb{E}[Y_0]) + \sum_{i=0}^{t+k-1} \phi_1^i Z_{t+k-i}\Big]\Big] \\
&= \phi_1^t \phi_1^{t+k} \mathbb{E}\Big[(Y_0 - \mathbb{E}[Y_0])^2\Big] + \mathbb{E}\Big[(\sum_{i=0}^{t-1} \phi_1^i Z_{t-i})(\sum_{i=0}^{t+k-1} \phi_1^i Z_{t+k-i})\Big] \\
&= \phi_1^{2t+k} Var(Y_0) + \mathbb{E}\Big[\phi_1^k (\sum_{i=0}^{t-1} \phi_1^i Z_{t-i})^2\Big] \\
&= \phi_1^{2t+k} Var(Y_0) + \phi_1^k \sum_{i=0}^{t-1} \phi_1^{2i} Var(Z_{t-i}) \\
&= \phi_1^{2t+k} Var(Y_0) + \phi_1^k \sigma^2 \sum_{i=0}^{t-1} \phi_1^{2i}
\end{aligned}
$$

$$= \begin{cases} \phi_1^{2t+k} Var(Y_0) + \sigma^2 \phi_1^k \dfrac{1-\phi_1^{2t}}{1-\phi_1^2} & \|\phi_1\| \neq 1 \\ Var(Y_0) + \sigma^2 t & \|\phi_1\| = 1 \end{cases}$$

Tal y como podemos observar, en los tres momentos calculados obtenemos una dependencia de t, por lo que el proceso Y_t no es estacionario. Si suponemos que $\|\phi_1\| < 1$ y que Y_0 sigue una distribución con media nula y varianza constante es inmediato que:

- $\lim_{t\to\infty} \mathbb{E}Y_t = \frac{\mu}{1-\phi_1}$
- $\lim_{t\to\infty} Var(Y_t) = \frac{\sigma^2}{1-\phi_1^2}$
- $\lim_{t\to\infty} Cov(Y_t, Y_{t+k}) = \sigma^2 \frac{\phi_1^k}{1-\phi_1^2}$

Claramente, estos tres momentos hacen que Y_t verifique las condiciones de estacionariedad. Para comprender la relación que tiene el parámetro ϕ_1 con el polinomio autorregresivo, vamos a expresar el modelo utilizando el operador de retardo B:

$$Y_t = \mu + \phi_1 Y_{t-1} + Z_t = \mu + \phi_1 B(Y_t) + Z_t \tag{3.3.7}$$

y obtenemos la siguiente expresión

$$(1 - \phi_1 B) Y_t = \mu + Z_t \tag{3.3.8}$$

Y $\phi(B) = 1 - \phi_1 B$ es el polinomio autorregresivo. En este caso el polinomio tiene una única solución

$$\phi(B) = 0 \Rightarrow 1 - \phi_1 B = 0 \Rightarrow B^* = \frac{1}{\phi_1} \tag{3.3.9}$$

Como hemos visto, el proceso Y_t será estacionario si $\|\phi_1\| < 1$, y esto es equivalente a $\|B^*\| > 1$. □

Esta propiedad será especialmente interesante a la hora de realizar pruebas estadísticas sobre la serie temporal, como veremos más adelante.

3.3.2 *Modelos de predicción autorregresivos de media móvil*

Vamos a presentar finalmente los modelos principales de predicción basados en los procesos estocásticos que acabamos de introducir, los procesos $ARMA(p, q)$.

Los procesos $ARMA(p, q)$ permiten definir los modelos de predicción autorregresivos de media móvil, $ARMA(p, q)$. Para ello, definimos el modelo según la expresión:

$$Y_t = \mu + \sum_{k=1}^{p} \phi_k Y_{t-k} + Z_t + \sum_{i=1}^{q} \theta_i Z_{t-i} \tag{3.3.10}$$

donde $\{Z_t\} \sim WN(0, \sigma^2)$, μ es la media de $\{Y_t\}$ y ϕ y θ son los polinomios autorregresivos (AR) y de media móvil (MA).

Este modelo define un conjunto de parámetros que se pueden ajustar iterando a través del conjunto de datos de entrenamiento. Sin embargo, es necesario asegurar que el espacio de búsqueda de los valores de los parámetros no es demasiado grande.

Para restringir ese espacio de búsqueda se define una nueva propiedad de un proceso $ARMA(p, q)$, la invertibilidad. Diremos que un proceso MA(q) es invertible si todas las raíces del polinomio característico, esto es, las raíces de la ecuación $\theta(z) = 0$ tienen norma menor que 1 en $\mathbb{C}$.

Se puede comprobar que un proceso MA(q) invertible es equivalente a un proceso $AR(\infty)$. Por lo tanto, un modelo $ARMA(p, q)$ modela un proceso estocástico estacionario e invertible. Si no impusiéramos esta última condición el espacio de búsqueda de parámetros del modelo $MA(q)$ sería del orden de 2^q, lo cual es mejor evitar.

Modelo autorregresivo integrado de media móvil

Como variación del modelo anterior podemos considerar el modelo autorregresivo integrado de media móvil, $ARIMA(p, d, q)$, que incluye un paso anterior de diferenciación de orden d de la serie temporal, por lo tanto $ARIMA(p, 0, q)$ equivale a $ARMA(p, q)$. La diferenciación se encarga de eliminar la tendencia y la estacionalidad presentes en la serie, convirtiéndola en una serie temporal estacionaria, y por tanto modelable por $ARMA(p, q)$.

Modelo estacional autorregresivo integrado de media móvil

Sin embargo, si una serie temporal tiene una componente estacionario demasiado fuerte, un modelo ARIMA puede no ser capaz de modelarla correctamente. En estos casos, un modelo de ARIMA estacional (SARIMA) puede ser conveniente [14]. Este modelo parte del modelo $ARIMA(p, d, q)$ e incorpora nuevos factores estacionales, determinados por:

- P: componente autorregresivo estacional
- Q: componente de media móvil estacional
- D: componente de diferenciación estacional

La notación utilizada para este modelo es $SARIMA(p, d, q) \times (P, D, Q)_s$, donde s es el periodo de estacionalidad. La expresión que lo define es la siguiente

$$\Phi(B^s)\phi(B)\Delta_s^D\Delta^d(X_t - \mu) = \Theta(B^s)\theta(B)Z_t \tag{3.3.11}$$

O alternativamente:

$$Y_t = \mu + \sum_{k=1}^{p} \phi_k Y_{t-k} + \sum_{i=1}^{q} \theta_i Z_{t-i} + \sum_{k=1}^{P} \gamma_k Y_{t-sk} + \sum_{i=1}^{Q} \eta_i Z_{t-si} + Z_t \tag{3.3.12}$$

donde $\{Z_t\} \sim WN(0, \sigma^2)$, μ es la media de $\{Y_t\}$.

Se puede ver claramente que es una extensión de los modelos anteriores incorporando la estacionalidad. Esto quiere decir que permite modelar series estacionarias, así como series estacionales y no estacionarias.

3.3.3 *Prophet*

Este modelo fue introducido por Facebook [28], su objetivo original era pronosticar datos diarios con estacionalidad semanal y anual, teniendo en cuenta el efecto de los días festivos. Para ello, utiliza una descomposición de la serie temporal en tres componentes principales: la tendencia, la estacionalidad, y los festivos. La expresión de la serie temporal es la siguiente

$$y_t = g_t + s_t + h_t + \epsilon_t \tag{3.3.13}$$

donde

- g_t modela la tendencia o los cambios no periódicos en la serie temporal
- s_t representa la estacionalidad, los cambios periódicos en la serie
- h_t representa el efecto de los días festivos que ocurren en horarios irregulares a través de distintos días
- ϵ_t es la componente residual

Para modelar la **tendencia** se establecen dos modelos distintos, en función de si el sistema se puede saturar cuando se supere cierto umbral. En nuestro caso, por la naturaleza del problema, solamente nos interesa el segundo modelo, en el que el sistema no presenta dicha saturación. Este modelo se llama **tendencia lineal con puntos de cambio**. Aunque no queremos adentrarnos en mucho detalle, mencionaremos ciertos aspectos característicos. Este modelo tiene en cuenta la tasa de crecimiento y un vector de ajustes en ciertos puntos para la tasa de crecimiento. También incluye un parámetro de desplazamiento que conecta los puntos finales de cada segmento.

Para representar la **estacionalidad** se utilizan series de Fourier:

$$s_t = \sum_{n=1}^{N} (a_n cos(\frac{2\pi nt}{L}) + b_n sin(\frac{2\pi nt}{L})) \tag{3.3.14}$$

donde L es el periodo de la estacionalidad.

En este caso ajustar el modelo es equivalente a estimar los parámetros $\beta = [a_1, b_1, ..., a_n, b_n]$. Para ello se construye el vector de estacionalidad para cada valor de t en los datos de entrenamiento:

$$X(t) = [cos(\frac{2\pi(1)t}{L}), ..., sin(\frac{2\pi(1)t}{L})] \tag{3.3.15}$$

y entonces se modela la componente estacional como

$$s_t = X(t)\beta \tag{3.3.16}$$

Las vacaciones y días festivos son variables en función de la ubicación, pues todas las festividades tienen un componente cultural. Para ello, Prophet incluye un calendario por defecto en cada país y además permite crear un calendario particular dependiendo de la aplicación que se le quiera dar al modelo. Para estimar los efectos de las festividades en la predicción de la serie temporal se asume que los días festivos son independientes entre sí.

Suponemos que tenemos L celebraciones, y para cada día festivo i sea D_i el conjunto de fechas pasadas y futuras de ese día, y añadimos un parámetro κ_i, que representa el cambio en la predicción para un día festivo concreto. Aunque normalmente la fecha de una celebración no cambia, en ciertas festividades el día no se mantiene constante a lo largo de los años.

Definimos entonces una función que indica si un tiempo t es parte de alguno de los días festivos considerados:

$$Z(t) = [1(t \in D_1), ..., 1(t \in D_L)] \tag{3.3.17}$$

Entonces el modelo del efecto de estos días es

$$h_t = Z(t) * \kappa \tag{3.3.18}$$

Con Prophet el problema de predicción se convierte en un problema de ajuste de una curva, a diferencia de los modelos que veíamos con anterioridad, en los que se considera de manera explícita la dependencia temporal de los datos, al definir la serie temporal como un proceso estocástico. Esto ofrece ciertas ventajas que resultan muy prácticas a la hora de realizar predicciones. La primera es la mejora de la flexibilidad, pues se permite tener distintos periodos de estacionalidad y la capacidad de ajustar la componente de tendencia según las necesidades del problema. Además, tenemos la posibilidad de incluir datos que no estén uniformemente distribuidos a lo largo del tiempo y por último, la rapidez del entrenamiento comparado con otros modelos de series temporales, lo que lo hace más eficiente en términos de tiempo y recursos.

3.4 PRUEBAS ESTADÍSTICAS

A continuación estudiaremos más conceptos estadísticos, en este caso relacionados con la inferencia. Repasaremos en qué consiste hacer una prueba estadística y describiremos en detalle las dos pruebas que hemos utilizado en el análisis de las series temporales.

3.4.1 *Descripción del proceso de prueba estadística*

Comenzaremos introduciendo la motivación tras las pruebas estadísticas, para ello es necesario repasar ciertos conceptos básicos que introducen de manera natural la inferencia estadística.

En estadística, llamamos **población** al conjunto de eventos que comparten una o varias de las características que se están estudiando, es el grupo total que se desea analizar con el objetivo de inferir ciertas propiedades estadísticas.

Por esta razón, es fundamental definir correctamente la población de interés de cara a que los resultados de los análisis estadísticos sean precisos. En el contexto de este estudio, la población se puede entender como el conjunto de observaciones de la serie temporal a lo largo del tiempo. Cada toma de datos representa una medición de la variable en cuestión en un momento específico en el tiempo, y por tanto la población sería el conjunto de todas las mediciones realizadas desde el inicio hasta el final del período de tiempo considerado.

Es poco práctico considerar el conjunto total de eventos posibles, por lo se utilizan las muestras estadísticas. Aunque hay distintas maneras de definir este concepto, consideraremos una **muestra estadística** como una secuencia $X_1, ..., X_n$ de variables aleatorias independientes e idénticamente distribuidas. Esto no es más que considerar una secuencia finita de eventos, pero de cara a realizar el estudio estadístico resulta más adecuada la primera interpretación.

Definición 3.4.1. (Inferencia estadística) La inferencia estadística es el proceso mediante el cual se utiliza una muestra para inferir propiedades de la población de la que se extrajo.

En el problema de inferencia estadística una propiedad o conjunto de ellas que determina la distribución conjunta de $X_1, ..., X_n$ se llama **parámetro de la distribución** [7]. Al conjunto Ω de todos los valores que puede tomar un parámetro θ se le denominará **espacio de parámetros**.

Aunque nuestro objeto de estudio no sea la inferencia, es necesario introducir un último concepto para poder exponer el proceso completo de prueba estadística: se trata del concepto de estadístico.

Definición 3.4.2. (Estadístico) Llamamos estadístico T a la función real de n variables $T = \phi(X_1, ...X_n)$. Un estadístico es una variable aleatoria en sí mismo y su distribución de probabilidad se conoce como distribución muestral del estadístico.

Los estadísticos se utilizan para poder estimar un parámetro de la distribución a partir de los datos muestrales disponibles. La idea es que el valor del estadístico proporciona una estimación razonable del valor del parámetro correspondiente de la población.

Uno de los métodos más comunes para realizar una inferencia estadística es la evaluación de hipótesis. Llamamos **hipótesis** a una afirmación sobre un parámetro de una población. A partir de una hipótesis en un problema de este estilo se pueden definir lo que denominamos las **hipótesis complementarias**: la hipótesis nula (H_0) y la hipótesis alternativa (H_1).

Si θ es un parámetro de la población, notaremos la hipótesis nula como $H_0 : \theta \in \Theta_0$ y la hipótesis alternativa como $H_1 : \theta \in \Theta_0^c$, donde Θ_0 es un subconjunto del espacio de parámetros Ω y Θ_0^c es su complementario.

En un problema de evaluación de hipótesis, tras analizar la muestra de la población, se tiene que tomar la decisión de aceptar H_0 como verdadero con cierto margen de confianza, o de rechazar H_0 y por tanto aceptar H_1 como verdadero. Definimos dos tipos de errores a la hora de evaluar una hipótesis,

- Error de tipo I, se da cuando se toma la decisión de rechazar H_0 y H_0 era verdad.
- Error de tipo II, se da cuando no se rechaza H_0 y en realidad se tenía que haber rechazado y por tanto, aceptar H_1.

Los pasos establecidos por [22] exponen perfectamente el proceso de evaluación de hipótesis:

Paso 1. Establecer el enunciado de las dos hipótesis complementarias. Este paso es fundamental pues hay que tener en cuenta el contexto y naturaleza del problema. Es un paso complejo y normalmente se recomienda enfocar en primer lugar la hipótesis alternativa H_1, pues es la más importante desde el punto de vista práctico. El enunciado de la hipótesis alternativa debe expresar las situaciones que queremos identificar a través de la evaluación de hipótesis.

Paso 2. Calcular un estadístico T que cumpla dos propiedades fundamentales. La primera es que su comportamiento tiene que variar de manera significativa cuando H_0 es verdad a cuando H_1 lo es; y la segunda es que debe ser posible calcular su distribución de probabilidad asumiendo que H_0 es verdad.

Paso 3. Elegir una región crítica, es decir, establecer qué valores de T nos indican de manera fiable que H_1 es verdad en vez de H_0. Normalmente se dan tres tipos de regiones, la derecha, donde rechazamos H_0 si el valor de T es mayor o igual que un valor crítico c_d; la izquierda, donde rechazamos H_0 si el valor de T es menor o igual que otro valor crítico c_i, o ambas, donde se rechaza H_0 si el valor de T es mayor o igual que un valor crítico c_d o si el valor de T es menor o igual que c_i. En el caso de que el valor de T se encuentre fuera de la región crítica, no podemos rechazar H_0. Es importante notar que no rechazar H_0 no implica aceptar H_0.

Paso 4. Decidir el tamaño de la región crítica. De manera formal supone definir el nivel de significancia de la prueba α, que representa el riesgo que se asume al rechazar H_0 cuando H_0 sea verdad.

Paso 5. Es un paso que no siempre se considera, pero es importante tener en cuenta en qué punto concreto se sitúa el valor de T, pues nos puede dar información de la cantidad de evidencia de la que disponemos para rechazar o aceptar H_0. De esta

manera está fuera de la región crítica, pero muy cerca de la frontera, se podría decir que hay alguna evidencia de que deberíamos rechazar H_0.

3.4.2 *Pruebas estadísticas aplicadas a series temporales*

Dentro del contexto anterior vamos a presentar en detalle dos pruebas estadísticas importantes que se basan en la evaluación de hipótesis. La primera prueba es el test de Dickey-Fuller [8], que se utiliza para determinar si una serie temporal es estacionaria. La segunda prueba es la prueba de Ljung-Box [3], que se utiliza para determinar si una serie temporal tiene autocorrelación significativa.

En este estudio en particular, utilizaremos la prueba de Ljung-Box para analizar los residuos de las distintas descomposiciones de series temporales, que como ya hemos estudiado, en una descomposición correcta se tratarían de ruido blanco, y por tanto no presentarían autocorrelación.

Estas pruebas son fundamentales en el análisis de series temporales, ya que nos permiten tomar decisiones adecuadas en cuanto al modelado de las series.

Augmented Dickey Fuller

En primer lugar vamos a estudiar el test original de Dickey-Fuller, pues supone la base teórica para el test aumentado (ADF). Este test se basa en el hecho de que la estacionariedad en un proceso autorregresivo depende del módulo de las raíces del polinomio autorregresivo, tal y como hemos estudiado con anterioridad.

Como resultado, obtenemos que si un proceso autorregresivo tiene alguna raíz ρ tal que $\|\rho\| = 1$, entonces podemos afirmar que el proceso no es estacionario. Las pruebas estadísticas cuyo objetivo es buscar una raíz que cumpla esa condición se llaman pruebas de raíz unitaria.

El test de Dickey-Fuller es una prueba de raíz unitaria que parte de un modelo $AR(1)$, con la siguiente estructura básica:

$$Y_t = \phi Y_{t-1} + Z_t \tag{3.4.1}$$

Además del modelo básico, normalmente se consideran dos alternativas:

- La primera incorpora una constante μ, y su objetivo es modelar el comportamiento de una serie temporal con media no nula

$$Y_t = \mu + \phi Y_{t-1} + Z_t \tag{3.4.2}$$

- La segunda incorpora también un componente dependiente del tiempo, y su objetivo es incluir la tendencia a la hora de modelar el comportamiento de la serie

$$Y_t = \mu + \beta t + \phi Y_{t-1} + Z_t \tag{3.4.3}$$

Las hipótesis complementarias en este caso son:

- H_0: $\phi = 1$
- H_1: $\phi \neq 1$

La evaluación de hipótesis se realiza ajustando mediante el método de mínimos cuadrados Y_t en cualquiera de sus variantes y comprobando si se verifica H_0 [11].

Sin embargo, con el test de Dickey-Fuller se restringe demasiado el orden del proceso autorregresivo. Para poder incluir procesos de mayor orden se desarrolló el test aumentado de Dickey-Fuller(ADF por sus siglas en inglés). Este test parte de un modelo autorregresivo $AR(p)$

$$Y_t = \mu + \beta t + \phi Y_{t-1} + \phi_1 \Delta Y_{t-1} + ... + \phi_p \Delta Y_{t-p} \tag{3.4.4}$$

El estadístico que se utiliza normalmente es

$$DF_\phi = \frac{T(\hat{\phi} - 1)}{1 - \hat{\phi_1} - ... - \hat{\phi_p}} \tag{3.4.5}$$

Si el valor del estadístico es mayor que el umbral crítico correspondiente, se puede rechazar la hipótesis H_0. En otras palabras, si el valor del estadístico es mayor que el umbral, se puede concluir que la serie temporal no tiene una raíz unitaria y, normalmente será estacionaria.

Ljung Box

Esta prueba se utiliza para determinar si las observaciones de una serie temporal son aleatorias o si por lo contrario presentan cierta autocorrelación a lo largo del tiempo. En nuestro estudio la utilizaremos para comprobar si los residuos resultantes de las distintas descomposiciones de la serie temporal presentan o no autocorrelación. Si los residuos no presentan autocorrelación, se consideran ruido blanco y se puede asumir que el modelo utilizado para descomponer la serie temporal es adecuado.

Las hipótesis complementarias son:

- H_0: los datos no están autocorrelados
- H_1: los datos están autocorrelados

En esta prueba no tenemos unos valores umbral predefinidos que nos permitan tomar la decisión de rechazar o aceptar H_0, sino que el estadístico que usamos sigue una distribución concreta. A través de los valores de la distribución con cierto margen de confianza tomamos la decisión.

La distribución utilizada es la distribución de Pearson (o χ^2) con $k \in \mathbb{N}$ grados de libertad. Esta distribución es la suma del cuadrado de k variables aleatorias independientes que siguen una distribución normal. Esto es, si tenemos $Z_1, ..., Z_k$ variables aleatorias, donde $Z_i \backsim N(0,1) \quad \forall i = 1, ..., k$, entonces la variable aleatoria X definida como

$$X = Z_1^2 + Z_2^2 + ... + Z_k^2$$

sigue una distribución χ^2, notado como $X \backsim \chi^2(k)$.

Para realizar la prueba estadística definimos el estadístico siguiente:

$$Q = n(n+2) \sum_{k=1}^{h} \frac{\rho_k^2}{n-k} \tag{3.4.6}$$

donde

- ρ_k es la autocorrelación en los datos con retardo k, esto es la medida de la correlación entre los valores de la serie con un desfase de k períodos.
- n es el tamaño de la muestra.
- h es el número de retardos o desfase a la hora de calcular la autocorrelación que se quieren probar.

Tras calcular el valor de Q con los datos, se compara con el valor de $\chi^2_{1-\alpha,h}$, el cuantil α de la distribución de Pearson con h grados de libertad. La región crítica definida para poder rechazar H_0 es

$$Q > \chi^2_{1-\alpha,h}$$

Si el valor obtenido para Q está dentro de esa región, estamos en disposición de rechazar la hipótesis de independencia y por tanto, podemos afirmar que existe cierta autocorrelación en los datos.

3.5 CONCEPTOS MATEMÁTICOS DE LAS REDES NEURONALES

Hemos discutido algunos de los métodos clásicos de predicción de series temporales, que se basan principalmente en la estadística y el análisis de las propiedades la serie temporal. Sin embargo, en las últimas décadas, el aprendizaje profundo ha emergido como una técnica poderosa para el análisis y la predicción de series temporales. En esta sección, introduciremos algunos conceptos previos necesarios para comprender matemáticamente la arquitectura y el entrenamiento de las redes neuronales, que veremos más adelante en este trabajo.

3.5.1 *Diferenciabilidad*

Los conceptos matemáticos que vamos a introducir giran en torno a la diferenciabilidad de funciones. Esto es porque para entrenar una red neuronal, es necesario ajustar los pesos de las conexiones entre las unidades o neuronas de cada capa, y este proceso, denominado *backpropagation*, se basa en el cálculo diferencial para obtener la derivada de la función de error con respecto a los pesos de la red.

Supongamos que V y W son espacios normados arbitrarios.

Definición 3.5.1. (Función diferenciable) Una función $f\colon A \subseteq V \to W$ es diferenciable en el punto $a \in \mathring{A}$ si existe una aplicación lineal y continua $g \in L(V, W)$ que verifique las propiedades siguientes, que son equivalentes:

$$\lim_{x \to a} \frac{\|f(x) - f(a) - g(x-a)\|}{\|x - a\|} = 0 \tag{3.5.1}$$

$$\lim_{x \to a} \frac{f(x) - f(a) - g(x-a)}{\|x - a\|} = 0 \tag{3.5.2}$$

A continuación vamos a estudiar tres propiedades de las funciones diferenciables, no demostraremos estos resultados pues no es el objeto principal de este estudio.

El primero de ellos establece una relación unívoca entre una función f y su **diferencial** en a, $g(a)$, notado $Df(a)$, que tiene como expresión:

$$Df(\mathbf{a})(\mathbf{v}) = \lim_{t \to 0} \frac{f(\mathbf{a} + t\mathbf{v}) - f(\mathbf{a})}{t} \tag{3.5.3}$$

Proposición 3.5.1. *Si f es diferenciable en a, entonces la función g anterior es única.*

El segundo establece la relación entre la diferenciabilidad y la continuidad de una función.

Proposición 3.5.2. *Si f es diferenciable en a, entonces es continua en a.*

Por último, vemos que resulta indiferente la elección de norma en 3.5.1, mientras sean normas equivalentes. Este resultado es especialmente significativo al hablar de espacios normados finitos, en los que todas las normas son equivalentes.

Proposición 3.5.3. *La diferenciabilidad de f en a, así como su diferencial $Df(a)$, se conservan al sustituir las normas de V y W por otras equivalentes a ellas.*

El siguiente resultado que vamos a estudiar es el resultado en el que se basa el proceso de *backpropagation*, que estudiaremos más adelante:

Teorema 3.5.1. *(Regla de la cadena) Sean V, W, Z tres espacios normados, y sean $M \subseteq V$ y $N \subseteq W$ subconjuntos abiertos. Sean dos funciones $f : M \to N$ y $h : N \to Z$. Si f es diferenciable en un punto $a \in M$, y h es diferenciable en $b = f(a)$, entonces $h \circ f$ es diferenciable en a, con*

$$D(h \circ f)(a) = Dh(b) \circ Df(a) = Dh(f(a)) \circ Df(a) \tag{3.5.4}$$

La regla de la cadena es esencial para el entrenamiento de redes neuronales porque permite calcular de manera eficiente el gradiente de una función compuesta. En el caso de las redes neuronales, la función objetivo que se busca minimizar está compuesta por múltiples capas de unidades interconectadas. Por tanto, para calcular el gradiente de la función objetivo respecto a los pesos de las neuronas en cada capa, se requiere aplicar la regla de la cadena de manera recursiva. A continuación vamos a introducir el concepto de gradiente, y terminaremos esta sección con el algoritmo de optimización del mismo.

Vamos a suponer de nuevo que V y W son dos espacios normados, y tomamos un abierto $M \subseteq V$ para definir $f : M \to W$. Sea $u \in V$ tal que $\|u\| = 1$, y fijamos $r\mathbb{R}^+$ tal que $B(a, r) \subset M$ y, para cada u de la forma anterior, consideramos la función ϕ_u definida como sigue:

$$\phi_u \colon (-r, r) \to W, \phi_u(t) = f(a + tu) \quad \forall t \in (-r, r) \tag{3.5.5}$$

Definición 3.5.2. (Derivada direccional) Dado $u \in V$ tal que $\|u\| = 1$, diremos que f es derivable en la dirección u, en el punto a, cuando la función ϕ_u es derivable en 0. En ese caso, al vector derivada $\phi'_u(0)$ lo llamamos derivada direccional de f en a, según la dirección u, y lo denotamos por $f'_u(a)$.

$$f'_u(a) = \varphi'_u(0) = \lim_{t \to 0} \frac{\varphi_u(t) - \varphi_u(0)}{t} = \lim_{t \to 0} \frac{f(a + tu) - f(a)}{t} \tag{3.5.6}$$

De la definición anterior podemos deducir que si f es diferenciable en un punto $a \in M$, entonces f es direccionalmente derivable en a con

$$f'(a) = Df(a)(u) \quad \forall u \in V : \|u\| = 1 \tag{3.5.7}$$

Definición 3.5.3. (Derivada parcial) Si tomamos ahora $V = \mathbb{R}^N$, junto con su base usual $\{e_1, ..., e_N\}$, entonces la derivada direccional de f en a , según la dirección e_k con $k = 1, ..., N$, se denomina derivada parcial de f con respecto a la k-ésima variable en el punto a, y se expresa como

$$\frac{\partial f}{\partial x_k}(a) = f'_{e_k}(a) = \lim_{t \to 0} \frac{f\,(a + te_k) - f(a)}{t} \tag{3.5.8}$$

Si lo anterior se verifica $\forall k = 1, ..., N$, diremos que f es parcialmente derivable en a, y entonces tenemos N derivadas parciales, que definen el **vector gradiente**, notado por:

$$\nabla f(\mathbf{a}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{a}), \frac{\partial f}{\partial x_2}(\mathbf{a}), \cdots, \frac{\partial f}{\partial x_N}(\mathbf{a}) \right) \tag{3.5.9}$$

Los conceptos anteriores se pueden desarrollar en profundidad, pero en este caso se han querido estudiar de una manera superficial con el único objetivo de garantizar la comprensión del algoritmo del descenso del gradiente, que vamos a introducir a continuación.

Algoritmo del descenso del gradiente

El algoritmo del descenso del gradiente es un método de optimización que se utiliza en el entrenamiento de redes neuronales. La idea esencial es encontrar el mínimo global de una función a través de la iteración de una fórmula matemática que se basa en la tasa de cambio de la función. Para hacerlo, el algoritmo comienza en un punto aleatorio de la función y calcula la tasa de cambio o gradiente en ese punto. Luego, se mueve en la dirección en la que la función disminuye más rápidamente, que es la dirección opuesta a la tasa de cambio, repitiendo este proceso hasta que llega a un mínimo global.

En el caso concreto de las redes neuronales, la función que se está optimizando es el error en un conjunto de datos de entrenamiento, y el objetivo es minimizarla para que la red pueda hacer predicciones precisas en datos nuevos.

Formalmente, se basa en el hecho de que si una función multivariable $F(x)$ donde $x \in \mathbb{R}^N$, es diferenciable en un entorno de un punto a, entonces $F(x)$ decrece más rápido si va desde a en la dirección del gradiente negativo de F en a, esto es, $-\nabla F(a)$.

Por lo tanto, obtenemos la fórmula matemática mencionada anteriormente:

$$a_{n+1} = a_n - \gamma \nabla F(a_n) \tag{3.5.10}$$

donde γ es un parámetro denominado tasa de aprendizaje o *learning rate*. En el entrenamiento de la red, a_n contiene información sobre los pesos de las conexiones entre las neuronas, y la tasa de aprendizaje determina la magnitud del paso que se toma en cada iteración para actualizarlos.

Si el *learning rate* es demasiado pequeño, el modelo convergerá muy lentamente hacia el mínimo global, lo que puede llevar a un largo tiempo de entrenamiento. Por otro lado, si es demasiado grande, el modelo puede divergir y nunca converger al mínimo global. Por lo tanto, elegir una tasa de aprendizaje adecuada es crucial para obtener una buena precisión.

Hemos visto los conceptos clave de cálculo diferencial y cómo se aplican al algoritmo del descenso del gradiente, que es una técnica esencial para el entrenamiento de redes neuronales. Ahora que tenemos una comprensión básica de cómo se optimizan los pesos de una red neuronal, podemos pasar a discutir la arquitectura básica de una red neuronal y explicar en detalle qué son los pesos ya mencionados y cómo se actualizan en el proceso de entrenamiento utilizando el algoritmo de *backpropagation*.

4

FUNDAMENTOS DE INFORMÁTICA

En este capítulo exploraremos el ajuste por mínimos cuadrados, del cual ya hemos visto algún uso en los modelos de series temporales introducidos en la sección anterior.

Discutiremos qué medidas de rendimiento existen para evaluar la calidad de las predicciones, lo que nos permitirá comparar diferentes algoritmos y enfoques y determinar cuál es el mejor para nuestro caso de uso concreto.

Después, profundizaremos en dos algoritmos de Machine Learning utilizados fuera del contexto de las series temporales: Support Vector Regression y K-Nearest Neighbours. Y finalmente, nos adentraremos en el estudio de la arquitectura y entrenamiento de las redes neuronales, y las matemáticas detrás de ellas, lo que nos permitirá comprender cómo funcionan y por qué son efectivas en la predicción de series temporales.

En resumen, este capítulo proporcionará una comprensión más profunda de algunos de los conceptos fundamentales del aprendizaje supervisado y cómo se aplican en la práctica.

4.1 AJUSTE DE ALGORITMOS

En secciones anteriores de este trabajo, hemos hablado sobre algunos de los métodos más comunes utilizados para realizar predicciones, como la regresión lineal y los modelos ARMA. Tal y como ya sabemos, en los problemas de regresión, el objetivo es encontrar una relación entre las variables independientes y la variable objetivo para poder hacer predicciones precisas.

Para lograr esto, se utiliza un proceso de ajuste de los datos. La técnica más conocida para realizarlo es la de los mínimos cuadrados, que minimiza la suma de los cuadrados de los residuos. En esta sección, estudiaremos en detalle esta técnica, pues resulta fundamental para el estudio de algoritmos de aprendizaje automático.

4.1.1 *Mínimos cuadrados*

Para estudiar en detalle esta técnica vamos a presentar el contexto general del problema. Sean (x_n, y_n) observaciones donde $x_n \in \mathbb{R}^N$. El objetivo es encontrar una función $f(x)$ con una forma concreta que se ajuste de la mejor manera a los datos y que sea fiable para poder hacer predicciones. La estructura de $f(x)$ depende del problema y se fija con anterioridad, pero como caso general podemos considerar que $f(x)$ es la combinación lineal de un conjunto de funciones $f_i(x)$ linealmente independientes

$$f(x) = \sum_{i=1}^{m} \alpha_i f_i(x) \tag{4.1.1}$$

En este contexto, encontrar la función $f(x)$ es equivalente a encontrar el valor óptimo de los coeficientes $\alpha_i \quad \forall i = 1, ..., m$.

Definimos el concepto de error o residuo, e_k, que tal y como su nombre indica es la diferencia entre el valor real de la variable objetivo y_k de una observación concreta y su valor predicho, $f(x_k)$, esto es

$$e_k = y_k - f(x_k) \tag{4.1.2}$$

Cuando usamos el ajuste por mínimos cuadrados consideramos que la mejor alternativa para $f(x)$ es la que minimiza el error cuadrático medio, que se define como

$$E(f) = \sqrt{\frac{\sum_{i=1}^{n} (e_i)^2}{n}} \tag{4.1.3}$$

Equivalentemente, podemos minimizar $E^*(f) = \frac{\sum_{i=1}^{n} (e_i)^2}{n}$. Esto es debido a que los valores de $\alpha_i \quad \forall i = 1, ..., m$ que minimizan $E(f)$ también minimizan $E^*(f)$.

Vamos a proceder a ejemplificar cómo se realizaría el ajuste de una función $f(x)$ en el caso general $f(x) = \sum_{i=1}^{m} \alpha_i f_i(x)$.

El error a minimizar es el siguiente:

$$E^*(f) = \frac{\sum_{i=1}^{n} (y_i - \sum_{j=1}^{m} \alpha_j f_j(x))^2}{n} \tag{4.1.4}$$

Observamos que el error $E^*(f)$ se puede considerar como una función sobre los parámetros $\alpha_i \quad \forall i = 1, ..., m$, es decir $E^*(f) = E^*(\alpha_1, ..., \alpha_m)$. Por lo tanto, el problema se soluciona encontrando un mínimo global de esta función en $\mathbb{R}^N$.

Para encontrar un mínimo global de $\mathbb{R}^N$ es necesario probar que la función es convexa, y por tanto cualquier mínimo local será un mínimo global.

Por lo tanto, basta con encontrar un mínimo local, y para ello calculamos el gradiente de la función $E^*(\alpha)$ y comprobamos en qué puntos se anula, es decir:

$$\nabla E^*(\alpha) = (\frac{\partial E}{\partial \alpha_1}(\alpha), ..., \frac{\partial E}{\partial \alpha_m}(\alpha)) = (0, ..., 0)$$

Para cada $i = 1, ..., m$ tenemos que resolver

$$\frac{\partial E}{\partial \alpha_i}(\alpha) = \sum_{k=1}^{n} 2(y_k - \sum_{j=1}^{m}(\alpha_j f_j(x_k)))(-f_i(x_k)) = 0 \tag{4.1.5}$$

Esta expresión da lugar a un sistema de m ecuaciones con m incógnitas denominado *Ecuaciones Normales de Gauss*, y para cada $i = 1, ..., m$ tiene la siguiente forma:

$$\begin{aligned}
&\sum_{k=1}^{n} 2(y_k - \sum_{j=1}^{m}(\alpha_j f_j(x_k)))(-f_i(x_k)) && = 0 \\
&2(-\sum_{k=1}^{n}(y_k f_i(x_k)) + \sum_{k=1}^{n}\sum_{j=1}^{m}(\alpha_j f_j(x_k) f_i(x_k))) && = 0 \\
&\sum_{k=1}^{n}\sum_{j=1}^{m}(\alpha_j f_j(x_k) f_i(x_k))) && = \sum_{k=1}^{n}(y_k f_i(x_k))
\end{aligned}$$

Bastará con resolver el sistema para encontrar los valores óptimos de $\alpha_1, ..., \alpha_m$ que identifican unívocamente a la función $f(x)$.

Llegados a este punto podemos decir que hemos explorado los fundamentos matemáticos del ajuste de mínimos cuadrados y cómo se aplica en los problemas de regresión. Si bien existen otras técnicas, el ajuste de mínimos cuadrados es uno de los más utilizados y efectivos, y resulta fundamental comprenderlo correctamente.

4.2 EVALUACIÓN DE MODELOS

En esta sección vamos a estudiar las diferentes medidas de rendimiento utilizadas al realizar predicciones en series temporales. Estudiar estas medidas es un paso esencial del proceso de modelado debido a que permiten evaluar la eficacia de los distintos modelos y determinar su capacidad para hacer predicciones precisas. Algunas de las medidas más comunes incluyen el error absoluto medio (*MAE*, por sus siglas en inglés), el error porcentual absoluto medio (*MAPE*) y la raíz del error cuadrático medio (*RMSE*), entre otras. Además, estudiaremos la técnica de validación cruzada,

que es una metodología comúnmente utilizada para evaluar el rendimiento de un modelo de una manera más precisa.

4.2.1 *Medidas de rendimiento*

Vamos a estudiar en detalle varias medidas que evalúan el rendimiento de los modelos de predicción en series temporales mediante fórmulas estadísticas. Es importante notar que cada medida solo proporciona una proyección de los errores del modelo, enfatizando ciertas características del rendimiento del modelo, por lo que no existe una única medida que sea siempre mejor que el resto. La alternativa usual es trabajar con una colección diversa de medidas, que nos permita considerar más factores y así poder tomar decisiones informadas.

El procedimiento habitual para evaluar un modelo de predicción se divide en tres etapas: preparación de datos, entrenamiento y evaluación.

En la preparación de datos se realiza todo el preprocesamiento necesario (imputación de datos y eliminación de datos atípicos, entre otros) y se separan los datos en los conjuntos de entrenamiento y evaluación. Este paso es fundamental pues es el que garantiza la ausencia de sesgo en la evaluación del modelo, pues ésta se realiza sobre datos que el modelo no ha visto anteriormente. Es natural observar que en función de la cantidad y naturaleza de los datos presentes en cada conjunto, el modelo final se comporte de una manera distinta.

En la etapa de entrenamiento, se ajustan los parámetros del modelo iterando sobre los datos de entrenamiento. Una vez que el modelo está entrenado, se puede empezar a evaluar el modelo. En esta última etapa se utilizan los datos de evaluación para analizar la capacidad del modelo para hacer predicciones precisas en datos no vistos anteriormente.

Podemos considerar una clasificación de las medidas de rendimiento [13]:

- **Medidas dependientes de la escala**, en las que la medida del error está expresada en las mismas unidades que los datos de origen.
- **Medidas de porcentaje**, que expresan el error como un porcentaje.

Las primeras normalmente se utilizan por su simpleza y facilidad de cálculo, sin embargo, no resultan adecuadas para comparar distintas series temporales debido a que la escala no tiene por qué ser la misma. Este problema se soluciona utilizando las segundas, aunque en este caso hay que prestar atención a las observaciones de la serie que sean igual a cero, pues entonces el resultado de la medida pierde consistencia.

Comenzamos estudiando las medidas dependientes de la escala más utilizadas a día de hoy:

- **Error Absoluto Medio** (MAE): se calcula tomando la media de los valores absolutos de las diferencias entre el valor real y_k y el valor predicho $f(x_k)$

$$MAE = \frac{\sum_{i=1}^{n} |y_i - f(x_i)|}{n} \tag{4.2.1}$$

  El único problema que plantea esta medida es que los valores poco frecuentes o atípicos no se ven penalizados.
- **Error Cuadrático Medio** (MSE): calcula la media de los cuadrados de las diferencias entre el valor real y_k y el valor predicho $f(x_k)$

$$MSE = \frac{\sum_{i=1}^{n} (y_i - f(x_i))^2}{n} \tag{4.2.2}$$

  Esta métrica sí que penaliza los valores atípicos debido a que al elevar al cuadrado el residuo, los de los valores atípicos tienen un peso mucho mayor. Por lo tanto, estos valores afectan significativamente al resultado de la métrica, lo que se traduce en un valor peor para la misma.
- **Raíz del error cuadrático medio** (RMSE), como su propio nombre indica, se calcula tomando la raíz del valor de MSE

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - f(x_i))^2}{n}} \tag{4.2.3}$$

Como medidas de porcentaje vamos a estudiar dos:

- **Error Porcentual Absoluto Medio** (MAPE), se calcula tomando la media de los valores absolutos de las diferencias entre el valor real y_k y el valor predicho $f(x_k)$ dividido por el valor real

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - f(x_i)|}{y_i} \tag{4.2.4}$$

  Existe también la posibilidad de eliminar la multiplicación por 100 y obtener el resultado como un valor entre 0 y 1. Las ventajas de utilizar esta medida son la independencia de la escala y su sencilla interpretabilidad. Sin embargo, tiene varias desventajas, como por ejemplo en el caso de tener valores de la serie muy próximos a 0, entonces el valor de la medida no es fiable, pues será infinitamente grande. De todas formas, esta es la medida de porcentaje más usada.
  Otra desventaja es que penaliza más los errores en los que $f(x_k)$ es mayor que y_k (errores negativos), que los errores positivos. Esto hace que si únicamente tomamos esta medida como referencia, obtengamos predicciones sesgadas.
- **Error Porcentual Absoluto Medio Simétrico** (sMAPE), es una variación del sMAPE que soluciona el problema de penalización sesgada, introduciendo una normalización dentro del cálculo de la media

$$sMAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - f(x_i)|}{(|y_i| + |f(x_i)|)/2} \tag{4.2.5}$$

Esta medida soluciona los problemas principales de MAPE pero su valor fluctúa entre 0 y 200. Por este motivo, existe una formulación alternativa que asegura que el valor de la métrica está en un formato porcentual

$$sMAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - f(x_i)|}{|y_i| + |f(x_i)|} \tag{4.2.6}$$

4.2.2 *Validación cruzada*

Hasta este punto, hemos considerado el procedimiento usual que conlleva separar los datos en un conjunto de entrenamiento y otro de evaluación, y posteriormente calcular el valor de las medidas de rendimiento sobre la totalidad del conjunto de evaluación. Sin embargo, este procedimiento tiene una desventaja: el resultado de la medida de rendimiento es muy variable dependiendo del conjunto que tomemos como entrenamiento. Si entrenamos el mismo modelo sobre distintos conjuntos de entrenamiento del mismo tamaño, la medida de rendimiento cambiará considerablemente. Teniendo en cuenta que estamos utilizando la medida como una base racional de cómo se comporta el modelo con datos no vistos, es necesario que sea lo más consistente posible.

Para solucionarlo existe una alternativa que divide el conjunto de observaciones en k grupos de mismo tamaño, $D_1, ..., D_k$. El primer grupo se toma como conjunto de evaluación y el resto como conjunto de entrenamiento. Se entrena el modelo concreto y se calcula la medida de evaluación, en este caso tomaremos MSE_1. Se repite el procedimiento k veces iterando a través de los grupos $D_2, ..., D_k$, calculando a cada paso el valor de $MSE_2, ..., MSE_k$. Tomamos como medida de evaluación final la media de las medidas calculadas en cada paso:

$$MSE = \frac{\sum_{i=1}^{k} MSE_i}{k} \tag{4.2.7}$$

Con esta técnica podemos obtener una visión más real y precisa del comportamiento del modelo con datos no vistos, de manera que la medida de rendimiento final resulta más fiable.

4.3 ALGORITMOS DE APRENDIZAJE SUPERVISADO

Aunque el objeto de este estudio son las series temporales, es interesante observar que también es posible realizar predicciones de series temporales mediante algoritmos genéricos de aprendizaje supervisado. En este estudio estos algoritmos se han utilizado para el preprocesamiento de los datos, tal y como veremos en el capítulo de Metodología.

Los dos algoritmos utilizados son variaciones de algoritmos cuyo objetivo original era la tarea de clasificación, esto es, asignar una clase o etiqueta a cada muestra de datos. Sin embargo, las versiones que vamos a describir en esta sección son las versiones dedicadas a la regresión, donde la variable objetivo es una variable continua.

4.3.1 *Support Vector Regression*

El primer algoritmo que vamos a estudiar es el algoritmo de Regresión de Vectores de Soporte, SVR. Su versión original fue desarrollada por Vladimir Vapnik en [30] y fue denominada Máquina de Vectores de Soporte (*Support Vector Machine*, SVM). Antes de estudiar en detalle los conceptos teóricos que hay detrás de este algoritmo, vamos a intentar describir de la manera más sencilla posible la intuición que lleva a él.

Comenzamos imaginando la regresión lineal más simple posible, donde tenemos una variable unidimensional x y la variable objetivo y, y n datos de entrenamiento. El objetivo en este problema es encontrar una recta que se ajuste lo mejor posible a los datos de entrenamiento.

En la regresión con vectores de soporte (SVR) en vez de ajustar una recta, el objetivo es ajustar un *tubo*, es decir, ajustamos una recta a los datos, con cierto margen de error ϵ, que se representa como dos rectas paralelas a la original. Se trata de encontrar una función $f(x)$ de manera que $f(x_n)$ diste de y_n menos que ϵ. Si una observación y_n verifica $|y_n - f(x_n)| \leq \epsilon$ entonces el residuo r_n es nulo. Las observaciones que verifiquen $|y_n - f(x_n)| = \epsilon$ se denominan **vectores de soporte**.

Además, la función $f(x)$ debe ser lo más plana posible. Esta condición puede confundir a simple vista, pero asegura que la función final sea robusta ante cambios en la variable de entrada x_n.

Es importante hacer dos observaciones, la primera es que la variable de entrada x_n no tiene por qué ser unidimensional, y de hecho en la mayoría de los casos no lo es. Esto quiere decir que en vez de una recta, estamos ajustando una superficie, en concreto un hiperplano, que por definición tendrá dimensión $n-1$ si la dimensión del espacio es n. En el caso bidimensional, el hiperplano es una recta, y se trata de la situación que acabamos de ver. La segunda observación es que los datos no siempre van a poder ser modelables por una función lineal, por lo que $f(x)$ puede no serlo. Esta es la gran diferencia que presenta este algoritmo con respecto a la regresión lineal.

Para describir el marco teórico del algoritmo, estudiamos el problema original, y después lo reformularemos para poder presentar las estructuras que permiten que $f(x)$ sea no lineal. Partimos de un conjunto de datos de entrenamiento, pares $(x_i, y_i) \quad \forall i = 1, ..., n$, donde $x_i \in \mathbb{R}^N \quad \forall i = 1, ..., n$.

Formulación primal

En la primera formulación del problema se asume que los daos son modelables mediante una función lineal, entonces buscamos una función $f(x)$ de la forma:

$$f(x) = x^T\beta + b \tag{4.3.1}$$

donde $\beta \in \mathbb{R}^N$.

Definimos una función de *coste*, que penaliza el error en la predicción, y es de la forma $c(x, y, f(x))$. Teóricamente los datos de entrenamiento se obtienen siguiendo una distribución de probabilidad $P(x, y)$, por lo que la función f se puede obtener como la función que minimice el funcional:

$$R[f] = \int c(x, y, f(x))dP(x, y) \tag{4.3.2}$$

Sin embargo, la distribución de probabilidad es desconocida, por lo que utilizamos una aproximación empírica del funcional:

$$\hat{R}[f] = \frac{\sum_{i=1}^{n} c(x_i, y_i, f(x_i))}{n} \tag{4.3.3}$$

Tal y como veíamos, es necesario asegurar también que la función f es lo más plana posible. Esto es equivalente a que el valor de $\frac{||\beta||}{2}$ sea lo más bajo posible. Normalmente se incluye un parámetro $\lambda > 0$, que se encarga de controlar el efecto de esta condición. De esta manera, el funcional a minimizar es el siguiente

$$\tilde{R}[f] = \lambda\frac{||\beta||}{2} + \frac{\sum_{i=1}^{n} c(x_i, y_i, f(x_i))}{n} \tag{4.3.4}$$

La función de coste más usual se define como sigue

$$\begin{aligned} c(x, y, f(x)) &= |y - f(x)|_\epsilon \\ &= \begin{cases} 0 & \text{si} \quad |y - f(x)| \leq \epsilon \\ |y - f(x)| & \text{si} \quad |y - f(x)| > \epsilon \end{cases} \end{aligned}$$

Existen multitud de funciones de coste que se pueden utilizar, la única condición que deben cumplir es la propiedad de convexidad. Este requisito es necesario pues estamos ante un problema de optimización, y es necesario asegurar la existencia y unicidad de un mínimo, así como poder encontrar mínimos globales a partir de los locales.

Idealmente, buscamos que en todos los puntos se verifique $c(x, y, f(x)) = 0$, sin embargo, esto no siempre es posible. Para lidiar con los puntos en los que esta condición

es imposible, incluimos dos parámetros ξ_n y ξ_n^* que representan un margen flexible, y que permiten que $c(x, y, f(x)) = 0$ aunque $|y - f(x)| > \epsilon$.

No obstante, es necesario controlar hasta qué punto se aumenta el margen de error permitido utilizando ξ_n y ξ_n^*, pues podríamos llegar a la situación en la que cualquier valor obtenido como error se admitiría como idóneo, y por tanto con coste cero. Para solucionar este problema, se añade un parámetro C que penaliza las observaciones que se encuentran fuera del margen de error ϵ. Este parámetro se fija antes de solucionar el problema de optimización.

Con todo lo estudiado, estamos en disposición de formular el problema general, se trata de encontrar:

$$min \quad \lambda \frac{||\beta||}{2} + C \sum_{i=1}^{n} (\xi_n + \xi_n^*) \tag{4.3.5}$$

tal que:

$$\begin{aligned} &\forall n: \quad y_n - (x_n^T \beta + b) \leq \epsilon + \xi_n \\ &\forall n: \quad (x_n^T \beta + b) - y_n \leq \epsilon + \xi_n^* \\ &\forall n: \quad \xi_n \geq 0 \quad \xi_n^* \geq 0 \end{aligned}$$

Formulación dual

Esta segunda formulación busca la obtención de una función de Lagrange con la función objetivo y las restricciones que hemos visto. Para hacerlo, incorpora una serie de variables $\alpha, \alpha^*, \eta, \eta^*$ de manera que obtenemos una función $L(\beta, b, \xi, \xi^*, \alpha, \alpha^*, \eta, \eta^*)$.

No vamos a entrar en la formalidad de las propiedades siguientes, pero la función L tiene un punto de silla respecto a las variables de la fórmula primal. Esto quiere decir que las derivadas parciales respecto de las variables β, b, ξ, ξ^* se deben anular. Con esta condición obtenemos que los parámetros η, η^* se pueden expresar en función de α, α^*. La función de Lagrange final nos proporciona la fórmula dual en forma de un problema de optimización sobre los parámetros α, α^*:

$$min \quad L(\alpha) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) x_i^T x_j + \epsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i^* + \alpha_i) \tag{4.3.6}$$

tal que

$$\sum_{i=1}^{n} (\alpha_n - \alpha_n^*) = 0$$

$$\forall n: \quad 0 \leq \alpha_n \leq C$$
$$\forall n: \quad 0 \leq \alpha_n^* \leq C$$

Podemos observar que el parámetro β se puede expresar como una combinación lineal:

$$\beta = \sum_{i=1}^{n} (\alpha_n - \alpha_n^*) x_n \tag{4.3.7}$$

Y entonces la función objetivo será

$$f(x) = \sum_{i=1}^{n} (\alpha_n - \alpha_n^*) x_n^T x + b$$

En [17] se establecieron ciertas condiciones complementarias que aseguran que se alcanza una solución óptima. Aunque no vamos a estudiarlas en detalle, es interesante conocer la intuición detrás de ellas. Vienen a indicar que todas las observaciones que entren de manera estricta en el tubo ϵ verifican que $\alpha_n = 0$ y $\alpha_n^* = 0$. Si $\alpha_n \neq 0$ o $\alpha_n^* \neq 0$ entonces la observación n se llama vector de soporte.

Relación no lineal

Como ya hemos mencionado, existen conjuntos de datos que no se pueden modelar mediante relaciones lineales. En este caso, podemos extender la formulación lineal mediante la inclusión de funciones no lineales que sí que puedan modelar los datos.

Si observamos la función $L(\alpha)$ en la formulación dual, utilizamos el producto escalar definido como

$$< x_i, x_j >= x_i^T x_j \tag{4.3.8}$$

Al sustituir el producto escalar por una función $G(x_i, x_j)$, llamada núcleo, podemos modelar relaciones no lineales. Un ejemplo de función no lineal $G(x_i, x_j)$ es el núcleo gaussiano, definido como

$$G(x_i, x_j) = exp(-||x_i - x_j||^2) \tag{4.3.9}$$

De esta manera, el problema de optimización dual no lineal es:

$$min \quad L(\alpha) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) G(x_i, x_j) + \epsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i^* + \alpha_i) \tag{4.3.10}$$

tal que

$$\sum_{i=1}^{n}(\alpha_n - \alpha_n^*) = 0$$
$$\forall n: \quad 0 \leq \alpha_n \leq C$$
$$\forall n: \quad 0 \leq \alpha_n^* \leq C$$

Aunque no sea un algoritmo específico para la predicción de series temporales, SVR puede resultar interesante por diversos motivos. El principal es su capacidad para modelar datos no lineales, pues este tipo de patrones están muy presentes en las series temporales. Además, es un algoritmo robusto a los valores atípicos o *outliers*, que también son un problema muy común en el modelado de series. Por último notar que SVR es un algoritmo muy flexible por la cantidad de parámetros que pueden ajustarse, y tiene una gran capacidad de manejar datos con una dimensionalidad grande, lo cual nos será especialmente útil en la imputación de datos, tal y como veremos más adelante.

4.3.2 *K-Nearest Neighbours*

El segundo algoritmo que vamos a estudiar es uno de los más simples a la hora de solucionar problemas de clasificación y de regresión. Sigue una intuición muy sencilla: el valor de la variable objetivo de una observación depende de las observaciones más cercanas.

En la práctica, para cada observación x, se calculan las k observaciones más cercanas a ella dentro del conjunto de entrenamiento, que son los pares $(x_1, y_1), ..., (x_k, y_k)$, y el valor de y será la media de los valores $y_i \quad \forall i = 1, ..., k$.

Teniendo en cuenta que $x \in \mathbb{R}^N$, podemos utilizar la distancia euclídea definida como

$$d(a, b) = \sqrt{\sum_{i=1}^{N}(a_i - b_i)^2} \qquad a, b \in \mathbb{R}^N \tag{4.3.11}$$

Es importante notar que la eficiencia del algoritmo depende enteramente del valor escogido para el parámetro k. En general, el valor óptimo depende del equilibrio entre sesgo y varianza. Un valor pequeño de k hace que el modelo sea flexible, y por tanto tenga un sesgo bajo y varianza elevada, pues la predicción en cierta región de $\mathbb{R}^N$ depende de un número pequeño de observaciones. Si utilizamos valores altos para k, obtendremos un modelo menos variable, pues la predicción en una región depende de un número mayor de valores, y no es tan susceptible a cambios.

En esta sección se ha presentado el funcionamiento de distintos algoritmos de Machine Learning. A continuación vamos a exponer en detalle la arquitectura y el proceso

de entrenamiento de las redes neuronales, ya introducidas de manera superficial en el capítulo anterior.

4.4 REDES NEURONALES

Tras haber introducido en el capítulo anterior los conceptos básicos de diferenciabilidad, en esta sección procedemos a introducir en detalle en qué consiste la arquitectura de una red neuronal. Después, estudiaremos cómo funciona el entrenamiento o ajuste de los pesos de la red, mediante *backpropagation*.

4.4.1 *Arquitectura de una red neuronal*

Para explicar correctamente las componentes de una red neuronal, vamos a utilizar la estructura expuesta en [16], simplificada, que identifica las siguientes componentes:

- Un conjunto de unidades de procesamiento, o neuronas.
- Un estado de activación para cada unidad a_k, que equivale a la salida de cada neurona.
- Conexiones entre unidades. Cada conexión está definida por un peso w_{jk}, que representa el efecto de la unidad j en la unidad k.
- Una función de propagación, que determina el efecto combinado de todas las unidades con conexión a la unidad k, denotada por s_k.
- Una función de activación $\mathcal{F}_k$, que dada la entrada efectiva s_k y el estado de activación anterior a_k, calcule el nuevo estado de activación.
- Una entrada externa para cada unidad, denominada sesgo, y denotada por b_k.

En la Figura 2 podemos ver de manera simplificada y poco rigurosa, la arquitectura de una red neuronal. La intención es proporcionar una visión más clara del funcionamiento de una red.

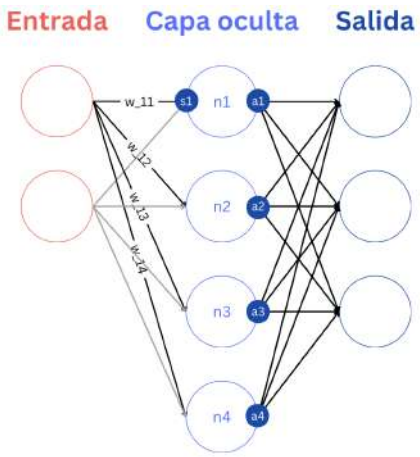

Figura 2: Arquitectura simplificada de una red neuronal

Cabe decir que todas las componentes definidas anteriormente, toman valores a lo largo del tiempo, es decir, que son valores que cambian a medida que el entrenamiento avanza. Por ejemplo, la activación de una unidad a_k, realmente es una función $a_k(t)$.

Cada unidad n_k puede ser de tres tipos: de entrada, de salida, u oculta. Las unidades de entrada son las que reciben la información del exterior, las de salida son las que producen la salida de la red neuronal, y las ocultas son las que se encuentran en las capas intermedias.

La manera más simple de calcular la función de propagación s_k de cada unidad, es realizando una combinación lineal de las unidades que conectan con ella, utilizando los pesos establecidos. Matemáticamente, s_k viene dada por:

$$s_k(t) = \sum_j w_{jk}(t)a_j(t) + b_k(t) \tag{4.4.1}$$

El siguiente paso, es establecer la función de activación $\mathcal{F}_k$, que representa el efecto de la entrada s_k en el estado de la unidad notado por a_k. Para ello, se define una función $\mathcal{F}_k$, que tenga como entrada ambos valores, y que produzca como salida el nuevo estado de activación de la unidad.

$$a_k(t+1) = \mathcal{F}_k\left(a_k(t), s_k(t)\right) \tag{4.4.2}$$

En la práctica, se suele usar una función creciente que tenga como entrada únicamente la activación, esto es:

$$a_k(t+1) = \mathcal{F}_k\left(s_k(t)\right) = \mathcal{F}_k\left(\sum_j w_{jk}(t)a_j(t) + b_k(t)\right) \tag{4.4.3}$$

Normalmente se suelen utilizar como funciones de activación funciones que impongan un límite superior sobre el valor de la activación de la unidad, las más utilizadas se encuentran en la Figura 3.

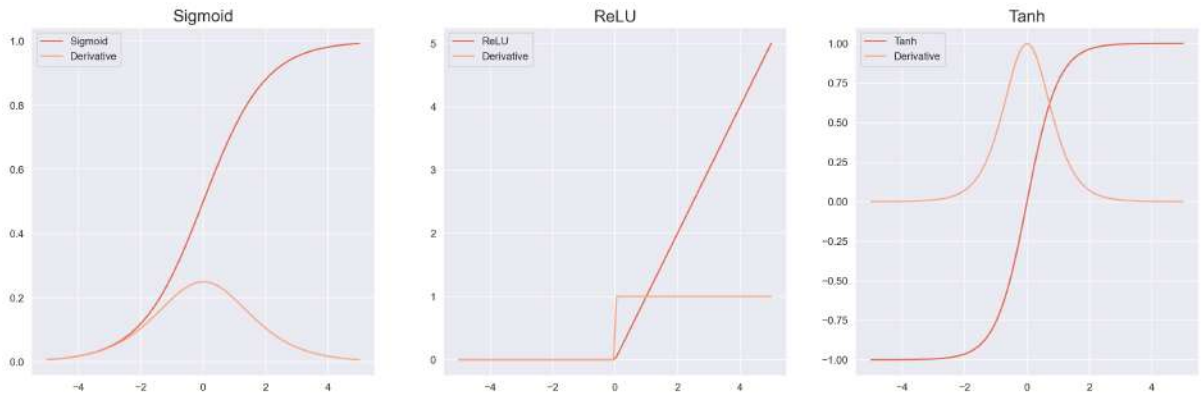

Figura 3: Principales funciones de activación

La primera función es la **función sigmoide**, cuya fórmula matemática es:

$$y(x) = \frac{1}{1+e^{-x}}$$
$$y'(x) = y(x)(1-y(x)) = \frac{e^{-x}}{(1+e^{-x})^2}$$

Esta función se utilizaba comúnmente en el pasado para redes neuronales porque su naturaleza diferenciable permitía que la red neuronal fuera entrenada mediante el algoritmo de *backpropagation*. Sin embargo, su uso se ha reducido en la actualidad debido a ciertos problemas que presenta, como la saturación de gradientes en los extremos de su rango. Este problema ocurre cuando los valores de entrada de una función de activación producen gradientes muy pequeños o cercanos a cero. Esto puede ser un problema en el entrenamiento porque hace que los pesos de las capas anteriores no se actualicen adecuadamente, lo que lleva a una convergencia lenta o incluso a un estancamiento en el aprendizaje.

La segunda función de activación es la denominada **ReLU** o **rectificador**, que matemáticamente es:

$$y(x) = max\{0, x\} \tag{4.4.4}$$

Existe otra versión suavizada, $y(x) = ln(1+e^x)$, pero el objetivo es el mismo: solucionar el problema de la saturación de gradientes en las redes neuronales. A diferencia de la función anterior, la función ReLU es una función de activación lineal que sólo se activa cuando su entrada es positiva, es decir, devuelve la entrada si es mayor que cero y cero en caso contrario. Esto hace que el cálculo del gradiente sea más eficiente y acelera el proceso de entrenamiento de la red neuronal.

Por último veremos la función **tangente hiperbólica**, que como su propio nombre indica, es:

$$y(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$y'(x) = 1 - tanh^2(x)$$

La función tanh tiene una forma similar a la sigmoide, pero su rango de valores se encuentra entre -1 y 1 en lugar de entre 0 y 1. Esta función es útil cuando se desea una salida que oscile entre valores negativos y positivos, y es especialmente útil en redes neuronales que trabajan con datos que se centran en cero. Sin embargo, la función tanh también puede sufrir de saturación de gradientes en los extremos, por lo que habrá que prestar atención a la hora de utilizarla.

4.4.2 *Entrenamiento de una red neuronal*

Tras estudiar la arquitectura básica de una red neuronal, procederemos a explicar en qué consiste el entrenamiento. A diferencia de lo que se vio en secciones anteriores, donde se dividían los datos en un conjunto de entrenamiento y otro de prueba, en las redes neuronales se suelen dividir en tres conjuntos: entrenamiento, validación y prueba.

El **conjunto de entrenamiento** es el que se utiliza para ajustar los pesos de la red neuronal, el **conjunto de validación** se utiliza para ajustar los hiperparámetros y la arquitectura de la red, y el **conjunto de prueba** se utiliza para evaluar el rendimiento final del modelo.

El conjunto de entrenamiento se suele dividir a su vez en subconjuntos de datos, llamados lotes o *batches* , y para entrenar la red se realiza un número fijo de iteraciones o *epochs* que recorren todo el conjunto de entrenamiento y en las que se ajustan los pesos de la red neuronal de acuerdo con el algoritmo del descenso del gradiente.

En el caso en el que se haya dividido el conjunto de entrenamiento en lotes, los pesos de la red se ajustarán cada vez que se termine de procesar un lote. En caso contrario, se actualizarán después de procesar una observación. De ahora en adelante vamos a asumir que se utiliza la subdivisión en lotes. El número de *epochs* necesarios para entrenar una red neuronal depende de la complejidad del problema y de la cantidad de datos disponibles, siendo un parámetro más a ajustar.

Para entrenar una red, en primer lugar, se inicializan los pesos con valores arbitrarios, y se realiza una primera iteración para obtener la salida de todos las observaciones del primer lote. Una vez se ha obtenido la salida, se compara con el valor real del conjunto de entrenamiento mediante una función de pérdida (*Loss function*) y se actualizan los pesos según el descenso del gradiente.

Al proceso de obtener una salida a partir de un subconjunto del conjunto de entrenamiento se le denomina *forward propagation*.

Forward propagation

Supongamos que el primer lote de entrenamiento está formado por $(x_1, y_1), ..., (x_m, y_m)$ observaciones donde $x_i \in \mathbb{R}^N$. Podríamos escribirlo como:

$$\mathbf{X} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{x}(1) & \mathbf{x}(2) & \dots & \mathbf{x}(m) \\ | & | & \cdots & | \end{bmatrix} \tag{4.4.5}$$

donde $X \in \mathbb{R}^{N,m}$.

También podríamos poner de esta forma el conjunto de salidas Y

$$\mathbf{Y} = \begin{bmatrix} y(1) & y(2) & \dots & y(m) \end{bmatrix} \tag{4.4.6}$$

Tal y como veíamos antes, la red neuronal está compuesta por $p \in \mathbb{N}$ capas donde cada capa contiene k_p unidades o neuronas. Vamos a centrarnos en la capa $i \in \mathbb{N}$, que contiene k_i unidades.

Entonces, podemos escribir el conjunto de valores de activación de las unidades de la capa i como una matriz $A^i \in \mathbb{R}^{k_i,m}$. Cada columna representará el valor de la unidad j con una observación del conjunto de entrenamiento, y formalmente, se escribe:

$$\mathbf{A}^i = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{a}^i(1) & \mathbf{a}^i(2) & \dots & \mathbf{a}^i(m) \\ | & | & \cdots & | \end{bmatrix} \tag{4.4.7}$$

Vamos a centrarnos en el primer elemento de la primera columna de A_i, es decir, el elemento $a_1^i(1)$. Este elemento representa la activación de la **primera unidad de la capa** i en la primera observación del conjunto de entrenamiento, esto es, utilizando (x_1, y_1). Sabemos que la activación es el resultado de aplicar una función $\mathcal{F}$ sobre el valor de la función de propagación $s_1^i(1)$.

Formalmente, tendríamos:

$$a_1^i(1) = \mathcal{F}_k\left(s_1^i(1)\right) \tag{4.4.8}$$

donde

$$s_1^i(1) = \sum_{j \in \text{unidades en } i-1} w_{1j}^i(1) a_j^{i-1}(1) + b_1^i(1) \tag{4.4.9}$$

Aunque se ha mantenido la notación anterior, de cara a facilitar la comprensión se puede observar que los subíndices siempre corresponden a la unidad y los superíndi-

ces representan la capa, así, $s_1^i(1)$ es la función de propagación de la primer unidad de la capa i, $w_{1j}^i(1)$ es el peso de la unidad j a la unidad 1 de la capa i, y $b_1^i(1)$ es el sesgo de la primera unidad de la capa i.

Esta forma de calcular $s_1^i(1)$ puede dar lugar a reescribir los pesos de la capa i de forma matricial, y entonces tendríamos $W^i \in \mathbb{R}^{k_i, k_{i-1}}$, formalmente:

$$\mathbf{W}^i(1) = \begin{bmatrix} w_{11}^i(1) & \cdots & w_{1k_{i-1}}^i(1) \\ w_{21}^i(1) & \cdots & w_{2k_{i-1}}^i(1) \\ \vdots & \ddots & \vdots \\ w_{k_i1}^i(1) & \cdots & w_{k_ik_{i-1}}^i(1) \end{bmatrix} \tag{4.4.10}$$

Y entonces el vector $s^i(1) \in \mathbb{R}_i^k$ viene dado por:

$$\begin{aligned} s^i(1) &= W^i(1)a^{i-1}(1) + b^i(1) \\ &= W^i(1)\mathcal{F}(s^{i-1}(1)) + b^i(1) \end{aligned}$$

Si volvemos a 4.4.8, podemos ver que $a_1^i(1)$ es el resultado de aplicar $\mathcal{F}$ sobre cada elemento del vector $s^i(1)$. Si repetimos esto de manera iterativa a través de las p capas, llegamos a producir la salida de la red neuronal para la primera observación, dada por $\hat{y}(1) = a^p(1) = \mathcal{F}(s^p(1)$.

Backpropagation

Una vez se ha obtenido la salida $\hat{y}(1), ..., \hat{y}(m)$ para cada observación dentro del lote, es el momento de actualizar el valor de los pesos, utilizando el descenso del gradiente.

Para ello, utilizamos una función de pérdida, que se encarga de comparar la salida de la red con el valor real, notada por $\mathcal{L}(\hat{y}(i), y(i))$. Se pueden tomar diversas funciones como función de pérdida, por ejemplo, podemos tomar el error cuadrático medio:

$$\mathcal{L}(\hat{y}(i), y(i)) = \frac{1}{|y|} \sum_{i=1}^{|y|} (y(i) - \hat{y}(i))^2 \tag{4.4.11}$$

En el entrenamiento de la red, el objetivo principal es minimizar una función de coste, que denotaremos por J, y que depende de los parámetros de la red $(W^1, ..., W^p, b^1, ..., b^p)$. En el caso de que se actualicen los parámetros después de cada observación, y por tanto, no se utilicen lotes, la función coste será idéntica a la función de pérdida. En otro caso, tendremos:

$$J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}(i), y(i)) \tag{4.4.12}$$

Se podría considerar una dependencia de J de los parámetros $(W^1, ..., W^p, b^1, ..., b^p)$, pues acabamos de comprobar que se puede expresar cualquier salida $\hat{y}(i)$ como una función sobre los pesos y los sesgos de cada capa.

Para minimizar J utilizamos el descenso del gradiente, y por tanto será necesario calcular el gradiente de la función con respecto a los parámetros que queremos optimizar, así, será necesario calcular para cada capa i, las matrices:

$$\frac{\partial J}{\partial \mathbf{W}^i} = \begin{bmatrix} \frac{\partial J}{\partial w^i_{11}} & \cdots & \frac{\partial J}{\partial w^i_{1k_{i-1}}} \\ \frac{\partial J}{\partial w^i_{21}} & \cdots & \frac{\partial J}{\partial w^i_{2k_{i-1}}} \\ \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial w^i_{k_i 1}} & \cdots & \frac{\partial J}{\partial w^i_{k_i k_{i-1}}} \end{bmatrix}$$

$$\frac{\partial J}{\partial \mathbf{b}^i} = \begin{bmatrix} \frac{\partial J}{\partial b^i_1} \\ \frac{\partial J}{\partial b^i_2} \\ \vdots \\ \frac{\partial J}{\partial b^i_k} \end{bmatrix}$$

Por lo tanto, será necesario calcular:

$$(\frac{\partial J}{\partial \mathbf{W}^1}, \frac{\partial J}{\partial \mathbf{b}^1}, ..., \frac{\partial J}{\partial \mathbf{W}^p}, \frac{\partial J}{\partial \mathbf{b}^p}) \tag{4.4.13}$$

Para ello, se procede de atrás hacia delante, utilizando la regla de la cadena. Vamos a considerar la última capa p, con la función de activación sigmoide, recordamos que se verificaba lo siguiente:

$$\begin{aligned} z^p &= \mathbf{W}^p a^{p-1} + b^p \\ a^p &= \sigma(z^p) \\ J(a^p, y) &= -y \log(a^p) - (1-y) \log(1-a^p) \end{aligned} \tag{4.4.14}$$

Y entonces, en esta capa será necesario calcular:

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{W}^p} &= \frac{dJ}{da^p} \frac{da^p}{dz^p} \frac{\partial z^p}{\partial \mathbf{W}^p} \\ \frac{\partial J}{\partial b^p} &= \frac{dJ}{da^p} \frac{da^p}{dz^p} \frac{\partial z^p}{\partial b^p} \end{aligned} \tag{4.4.15}$$

Una vez se hayan calculado las expresiones anteriores, se puede proceder a calcular los gradientes correspondientes a la capa anterior, esto es, $p-1$:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{W}^2} &= \frac{dJ}{dz^3}\frac{dz^3}{d\mathbf{a}^2}\frac{d\mathbf{a}^2}{d\mathbf{z}^2}\frac{\partial \mathbf{z}^2}{\partial \mathbf{W}^2} \\ \frac{\partial J}{\partial \mathbf{b}^2} &= \frac{d\mathcal{L}}{dz^3}\frac{dz^2}{d\mathbf{a}^2}\frac{d\mathbf{a}^2}{d\mathbf{z}^2}\frac{\partial \mathbf{z}^2}{\partial \mathbf{b}^2}\end{aligned} \tag{4.4.16}$$

Dejando aparte la aparente complexidad, es importante observar que para calcular estos gradientes, se utilizan los valores obtenidos en el cálculo de los gradientes de la capa p. Por este motivo se procede de atrás hacia delante.

Una vez se han obtenido todos los gradientes, se puede aplicar el algoritmo del descenso del gradiente para actualizar los parámetros, resultando en $\forall i = 1, ..., p$:

$$\begin{aligned}W^i &= W^i + \gamma\frac{\partial J}{\partial \mathbf{W}^i} \\ b^i &= b^i + \gamma\frac{\partial J}{\partial \mathbf{b}^i}\end{aligned}$$

La actualización de los pesos se repite cada vez que se termine de procesar un lote, e iterando a través de todo el conjunto de entrenamiento tantas veces como *epochs* se hayan fijado. Este proceso permite que la red pueda aprender a partir de los datos de entrada y producir una salida que se ajuste lo mejor posible a las etiquetas de entrenamiento correspondientes.

Es importante notar que aunque se ha estudiado en detalle el proceso de ajuste de pesos según el descenso del gradiente (también denominado SGD), existen otros algoritmos que mejoran la convergencia hacia un mínimo. Estos son el Adaptive Moment Estimation (*Adam*) y el Root Mean Square Propagation (*RMSProp*).

Adam es un optimizador adaptativo que ajusta el *learning rate* en función del historial de gradientes, lo que lo hace más eficiente que el SGD. También tiene un parámetro de momento que ayuda a suavizar el proceso de actualización de los pesos.

RMSProp es similar a Adam en el sentido de que es un optimizador adaptativo, pero utiliza una técnica diferente para ajustar el learning rate. RMSProp calcula un promedio móvil exponencial de los gradientes cuadrados y utiliza este valor para normalizar el gradiente.

En general cada optimizador tiene sus ventajas y desventajas, y la elección depende del problema y los datos específicos que se estén utilizando.

Se ha presentado la arquitectura básica de una red neuronal. Además, se ha explicado cómo se utiliza el algoritmo de backpropagation para entrenar la red neuronal y ajustar los pesos de las conexiones. Esta sección supone el final del capítulo de fundamentos informáticos y nos permite dar paso a la presentación de la metodología seguida en el trabajo.

Parte III

METODOLOGÍA

Presentación de los datos utilizados en el estudio. Descripción del proceso de limpieza, análisis y predicción de series temporales.

5

PRESENTACIÓN DE LOS DATOS

En este capítulo nos centraremos en la presentación de los datos que vamos a utilizar a lo largo de las secciones siguientes. En primer lugar, exploraremos el origen del dataset y cómo fue recopilado. Después realizaremos un análisis preliminar de los datos, presentando cada una de las variables que utilizaremos en nuestro análisis posterior. Este análisis es fundamental para poder comprender la estructura que tienen los datos. Con esta información, estaremos preparados para presentar el algoritmo de imputación de datos que hemos diseñado y profundizar en el análisis concreto de series temporales, cuestiones que veremos en los capítulos siguientes.

5.1 ORIGEN DE LOS DATOS

El origen de los datos es una parte fundamental para cualquier análisis de datos. Los datos pueden provenir de diversas fuentes, como encuestas, dispositivos de monitoreo, registros de transacciones, entre otras. Es importante conocer la fuente de los datos para entender su calidad y validez.

Los datos que usamos en este trabajo provienen del artículo ***A three-year dataset supporting research on building energy management and occupancy analytics*** [19], que describe la creación de un conjunto de datos a partir de un edificio construido en 2015 en Berkeley, California.

El conjunto de datos incluye tanto el consumo de energía del edificio en su totalidad como el consumo fragmentado por zonas y por tipo. También incluye información sobre la operabilidad de los sistemas HVAC (*heating, ventilation and air conditioning*), parámetros ambientales del exterior y del interior y del edificio y distintas cuentas de ocupación, entre otros. Los datos se tomaron durante un periodo de tres años, desde Enero del 2018 hasta Diciembre del 2020, incluyendo precisamente el periodo de la pandemia.

Es necesario notar que se dieron ciertos eventos poco comunes que afectaron a la recolección de datos a lo largo de los tres años. Éstos se pueden resumir en: incendios forestales, confinamiento debido a la pandemia COVID-19 y testeo del Control Predictivo por Modelo (*MPC*, por sus siglas en inglés) para mejorar la eficiencia energética del edificio.

Durante tres semanas entre 2018 y 2020, se cerraron las compuertas de aire para minimizar la entrada del humo causado por los incendios forestales. En Marzo del 2020, la mayoría de los trabajadores del edificio empezaron a trabajar desde casa, lo que redujo considerablemente la ocupación de las plantas estudiadas. Por último, durante cinco semanas en el 2020, se comprobó el Control Predictivo por Modelo para optimizar el sistema de aire acondicionado y por ende reducir el consumo de energía.

5.1.1 *Descripción del edificio*

El edificio estudiado es un edificio de tamaño medio destinado a oficinas, denominado *Building 59*, que está localizado en el Laboratorio Nacional de Lawrence Berkeley, y que abarca 10,4000 m2 de espacios distribuidos a través de cuatro plantas independientes y destinadas a objetivos distintos. Los datos que estudiaremos están tomados de la tercera y cuarta planta, que son las plantas que contienen el espacio de oficinas, cerradas y abiertas, respectivamente.

Las características técnicas del edificio lo convierten en un edificio diseñado para mantener un consumo eficiente de energía. Éstas incluyen una estructura enmarcada en acero con un muro cortina con ventanas integradas y un núcleo de espuma que hace de aislamiento. En las zonas de oficinas, se dispone de un sistema de distribución de aire bajo el suelo y un cielorraso con espacio para distribuir el aire del aire acondicionado.

El edificio está dividido en 57 zonas termales, tanto interiores como exteriores. Nos referimos con el término zona termal exterior a las zonas que tienen paredes y ventanas exteriores. De la misma manera, usaremos el término zona termal interior para todas las demás zonas.

Los datos recogidos en las zonas exteriores se tomaron con sensores incorporados en la pared, y en las zonas interiores se recogieron con 16 sensores que fueron colocados a la altura de los escritorios lo más cerca posible de los trabajadores. Además, para medir la ocupación del edificio, se utilizaron sensores con cámara en las seis entradas y salidas del ala sur del edificio, y se contó el número de conexiones wifi en un momento determinado. Podemos comprobar la ubicación de los sensores en la Figura 4:

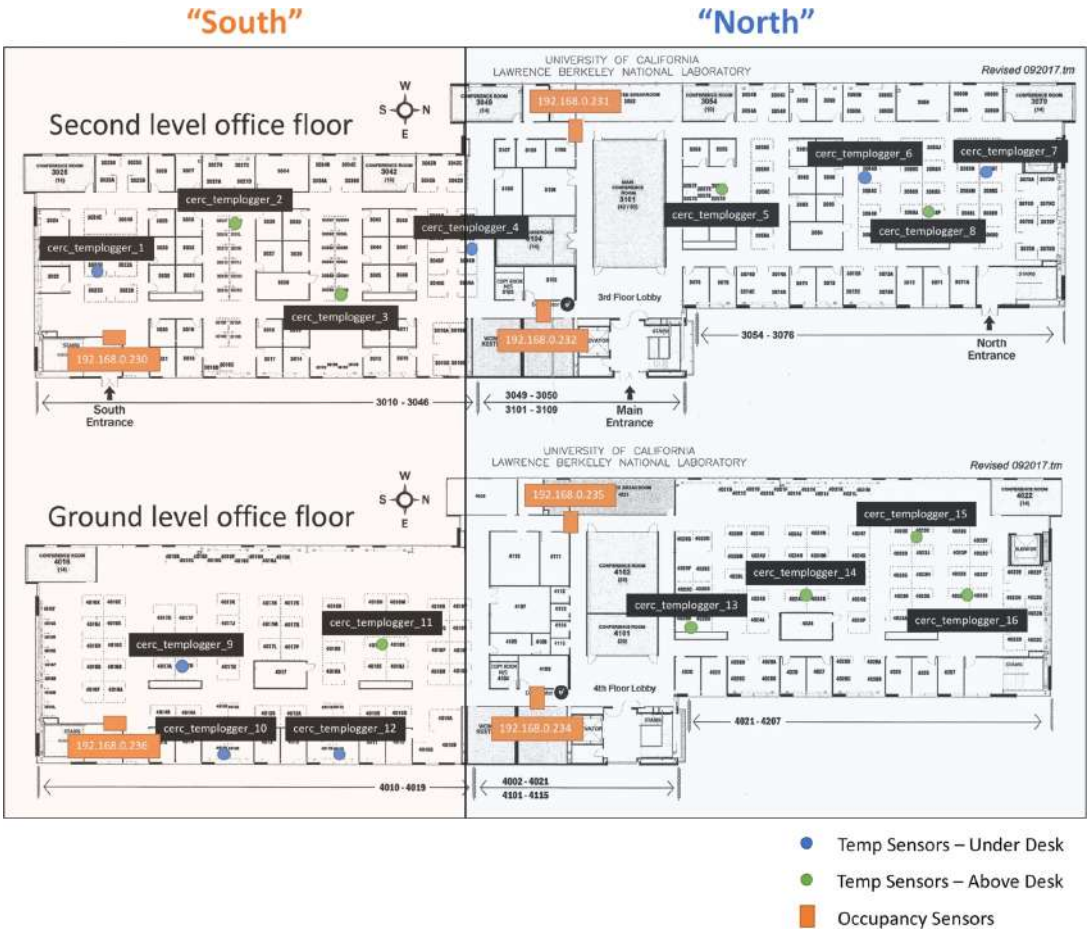

Figura 4: Distribución de los sensores de ocupación y de temperatura

5.1.2 *Descripción de las variables principales*

Los datos de las series temporales obtenidas del edificio se pueden organizar en cinco categorías: datos de consumo de energía, datos de clima exterior, datos de clima interior, datos de operación del sistema de aire acondicionado, y datos de ocupación.

Para este trabajo solamente hemos considerado los tres primeros, debido a la complejidad técnica de los datos de operación de HVAC y a la fragmentación en los datos de ocupación.

En la Tabla 1, podemos ver la información relevante sobre cada una de las variables consideradas.

Datos	Nombre del Fichero	Columna	Descripción	Dim	Unidad	Frecuencia de muestra	Tasa de ausencia
Consumo de energía	ele.csv	mels_S	Consumo de energía misceláneo para el ala Sur	1	kW	15 min	0.33
		mels_N	Consumo de energía misceláneo para el ala Norte	1	kW	15 min	0.20
		lig_S	Consumo de energía para la luz en el ala Sur	1	kW	15 min	0.18
		hvac_S	Consumo del sistema de aire acondicionado en el ala Sur	1	kW	15 min	0.08
		hvac_N	Consumo del sistema de aire acondicionado en el ala Norte	1	kW	15 min	0.08
Datos de clima exterior	site-weather.csv	air_temp_set_1	Temperatura del aire exterior en el sensor 1	1	°C	15 min	0.003
		air_temp_set_2	Temperatura del aire exterior en el sensor 2	1	°C	15 min	0.005
		dew_point_temperature	Temperatura del punto de rocío en el sensor 2	1	°C	15 min	0.011
		relative_humidity_set_1	Humedad relativa en el sensor 1	1	%	15 min	0.009
		solar_radiation_set_1	Radiación solar exterior en el sensor 1	1	W/m2	15 min	0.009
Datos de clima interior	zone_temp_sp_c.csv	zone_*_cooling_sp	Temperatura de refrigeración (setpoint) en la Zona *	41	°F	5 min	0.05–0.07
	zone_temp_sp_h.csv	zone_*_heating_ sp	Temperatura de calefacción (setpoint) en la Zona *	41	°F	5 min	0.05–0.06
	zone_temp_ interior.csv	cerc_templogger_*	Temperatura de la zona interior *	16	°F	10 min	0.01-0.21
	zone_temp_ exterior.csv	zone_*_temp	Temperatura de la zona exterior *	51	°F	1 min	0.15-0.20

Cuadro 1: Información relevante de las variables consideradas

Con la información de la tabla podemos realizar dos observaciones:

1. Hay ciertas variables con una tasa de ausencia relativamente alta, esto nos dice que el conjunto presenta un problema de falta de datos.
2. Tenemos distintas frecuencias de muestras, por lo que será necesario unificarlas antes de realizar el análisis, en la etapa de limpieza de datos, tal y como veremos más adelante.

5.2 ANÁLISIS EXPLORATORIO DE LOS DATOS

Vamos a proceder a estudiar la estructura general de las variables de la tabla en su versión bruta. Aunque los datos presenten un problema de falta de datos, las conclusiones obtenidas en el análisis no cambian, pues la distribución general de los datos es la misma. También realizaremos un estudio de la tasa de ausencia de cada variable, estudiando la distribución de los fallos en los datos. El objetivo de esta fase del trabajo es examinar y resumir los datos, con el fin de comprenderlos mejor y poder obtener información relevante para la toma de decisiones.

Vamos a dividir el estudio en los tres bloques en los que se dividen los datos: datos de consumo de energía, datos de clima exterior y datos de clima interior.

5.2.1 *Datos de consumo de energía*

Dado que esta variable la variable objetivo de este estudio, es vital comprender su comportamiento y los factores que influyen en ella. Por esta razón, el análisis realizado sobre los datos de consumo de energía ha sido especialmente exhaustivo. Tal y como hemos mencionado, estos datos se dividen en cinco variables, en función de la fuente de consumo de la que proviene el gasto y de la zona del edificio en la que se han recopilado:

- mels_S: consumo de energía misceláneo en el ala sur.
- mels_N: consumo de energía misceláneo en el ala norte.
- hvac_S: consumo de energía en aire acondicionado y ventilación en el ala sur.
- hvac_N: consumo de energía en aire acondicionado y ventilación en el ala norte.
- lig_S: consumo de energía por luz en el ala sur.

En la Figura 5 podemos ver en detalle la distribución de cada una de las variables a lo largo de los tres años, agrupadas por día para aumentar la legibilidad del gráfico.

Si estudiamos la proporción del gasto de energía que corresponde a cada fuente de consumo, podemos observar que el gasto en aire acondicionado y calefacción es el más significativo, representando el 80 %, 68 %, y 82 % del gasto total en 2018, 2019 y 2020, respectivamente. En la Figura 6 se presenta el desglose del gasto medio mensual por fuente de consumo. El gasto de HVAC es el mayor de manera consistente a lo largo de los tres años. Cabe destacar que en los meses de la pandemia, se dió

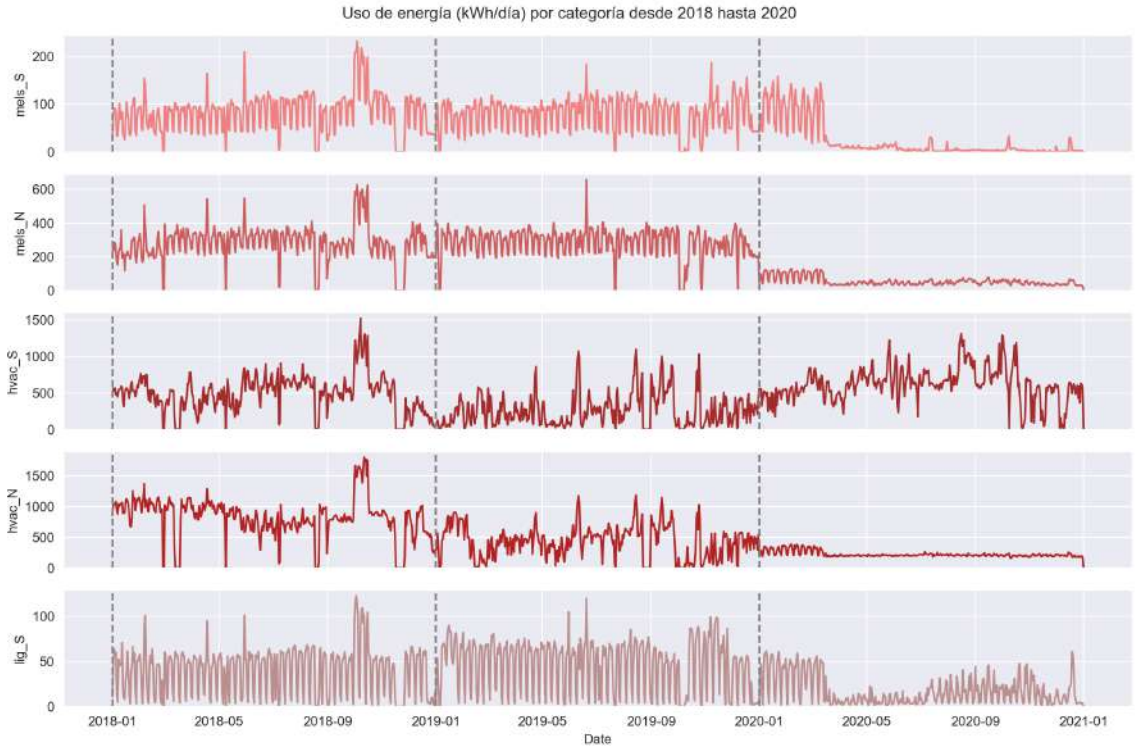

Figura 5: Uso de energía (kWh/día) por fuente de consumo desde 2018 hasta 2020

un descenso considerable en el gasto misceláneo, pero el gasto total de esos meses no se vió afectado en la misma manera. Esto puede explicarse estudiando cómo se comportan las variables en el año 2020 (Figura 5) vemos que el gasto en aire acondicionado y climatización se mantiene igual que en la situación previa a la pandemia. Como este es el gasto más significativo, podemos explicar que el gasto total no se vea influenciado por esa situación extraordinaria.

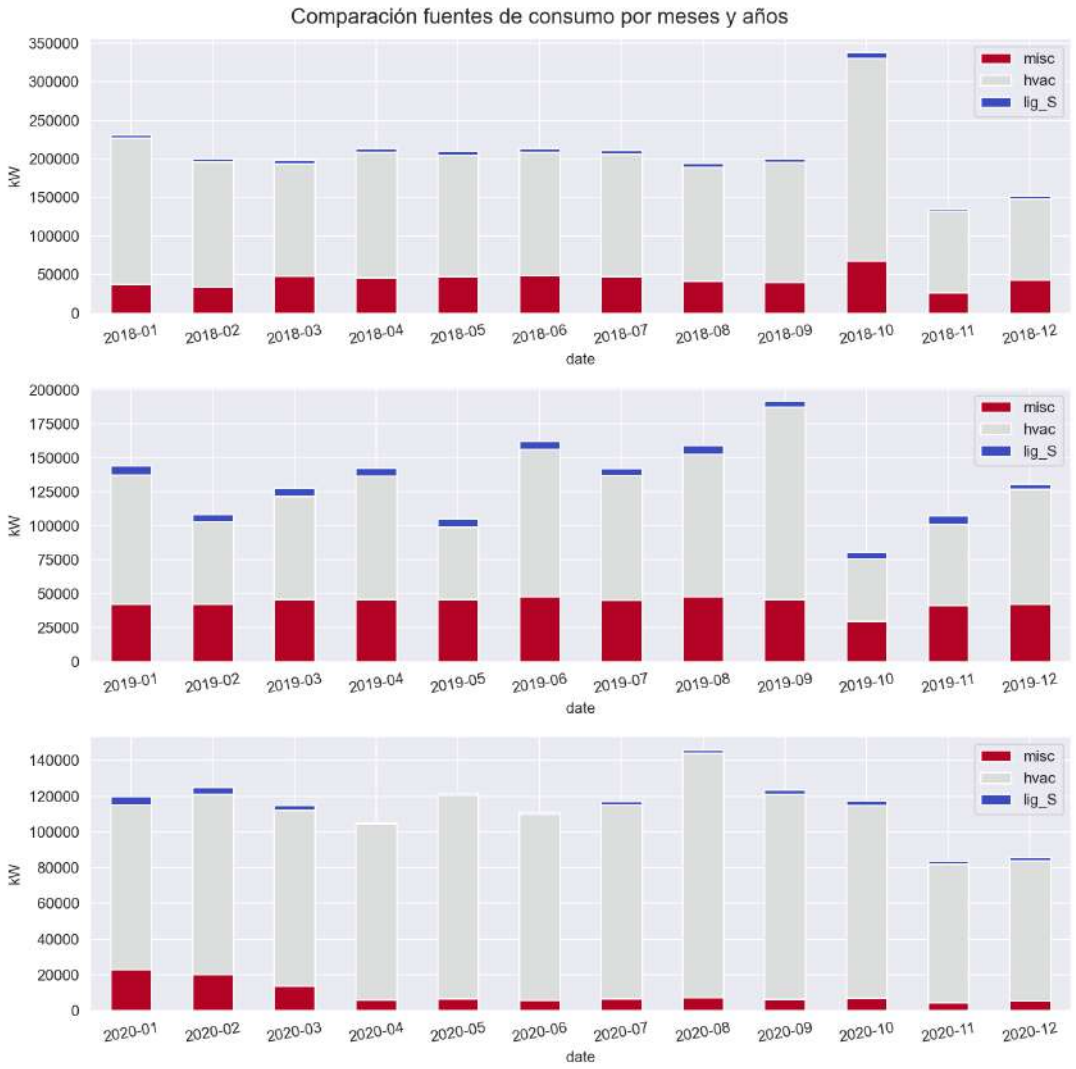

Figura 6: Comparación de fuentes de consumo por meses y años

Otra observación que podemos hacer es que de manera consistente el ala norte es la responsable de un mayor gasto. Esto puede ser debido a que incorpora más metros cuadrados (figura 4), o por otro tipo de cuestiones, como el aislamiento climático o alguna actividad particular que se realice en el ala norte. Para observar la comparación entre el gasto en ambas zonas, vamos a analizar por separado el gasto misceláneo y el gasto en HVAC.

En la Figura 7 vemos el gasto misceláneo por meses y años. Hemos considerado que no era necesario estudiar todos los meses y por esa razón se ve únicamente la mitad. En la Figura se puede constatar que a lo largo de todos los meses analizados, en ambas zonas se produce una subida en el gasto energético. Esto puede deberse a la realización de alguna actividad que conlleve un elevado gasto de energía en las fechas finales de cada mes.

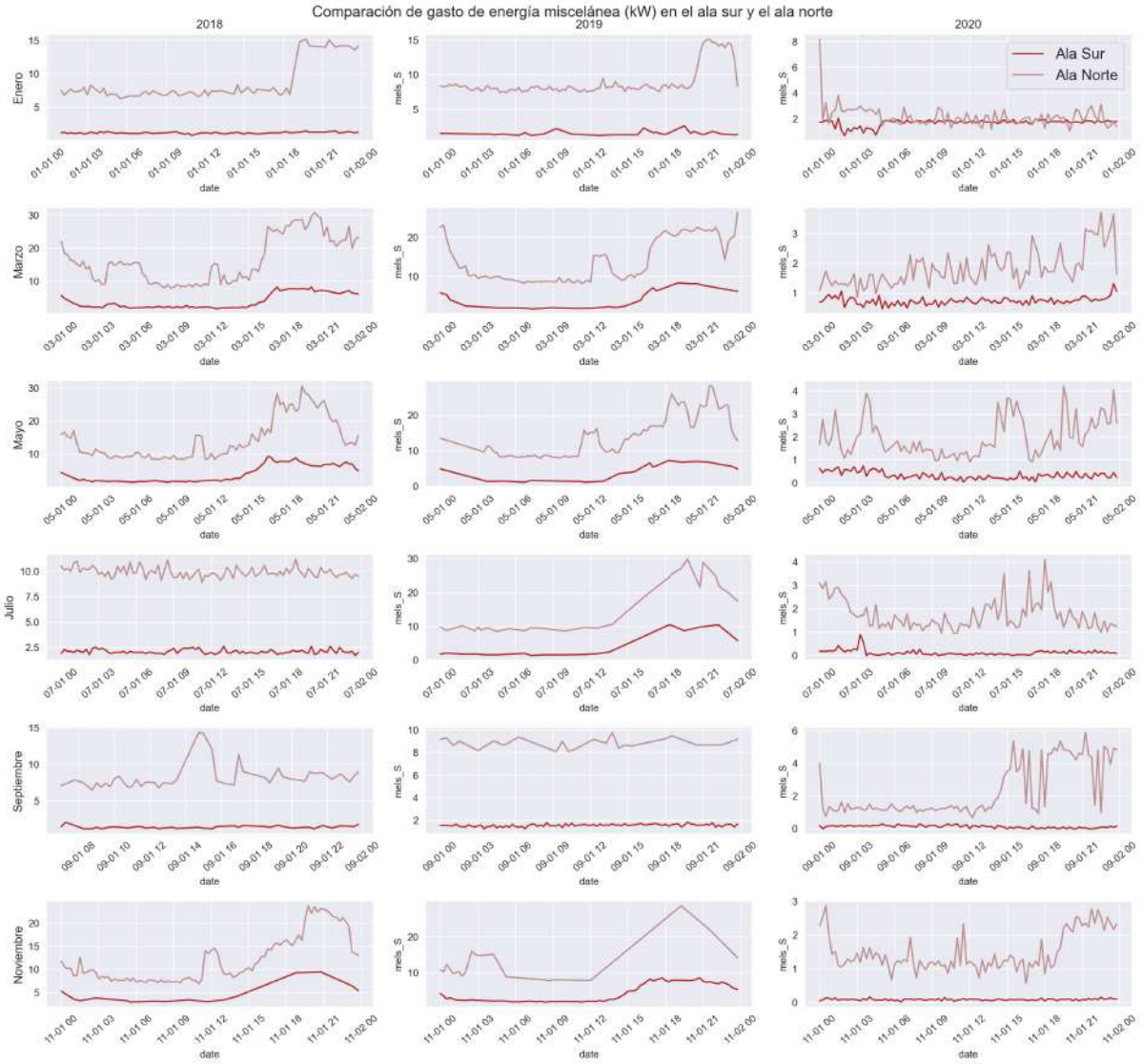

Figura 7: Comparación de gasto misceláneo de energía en alas norte y sur

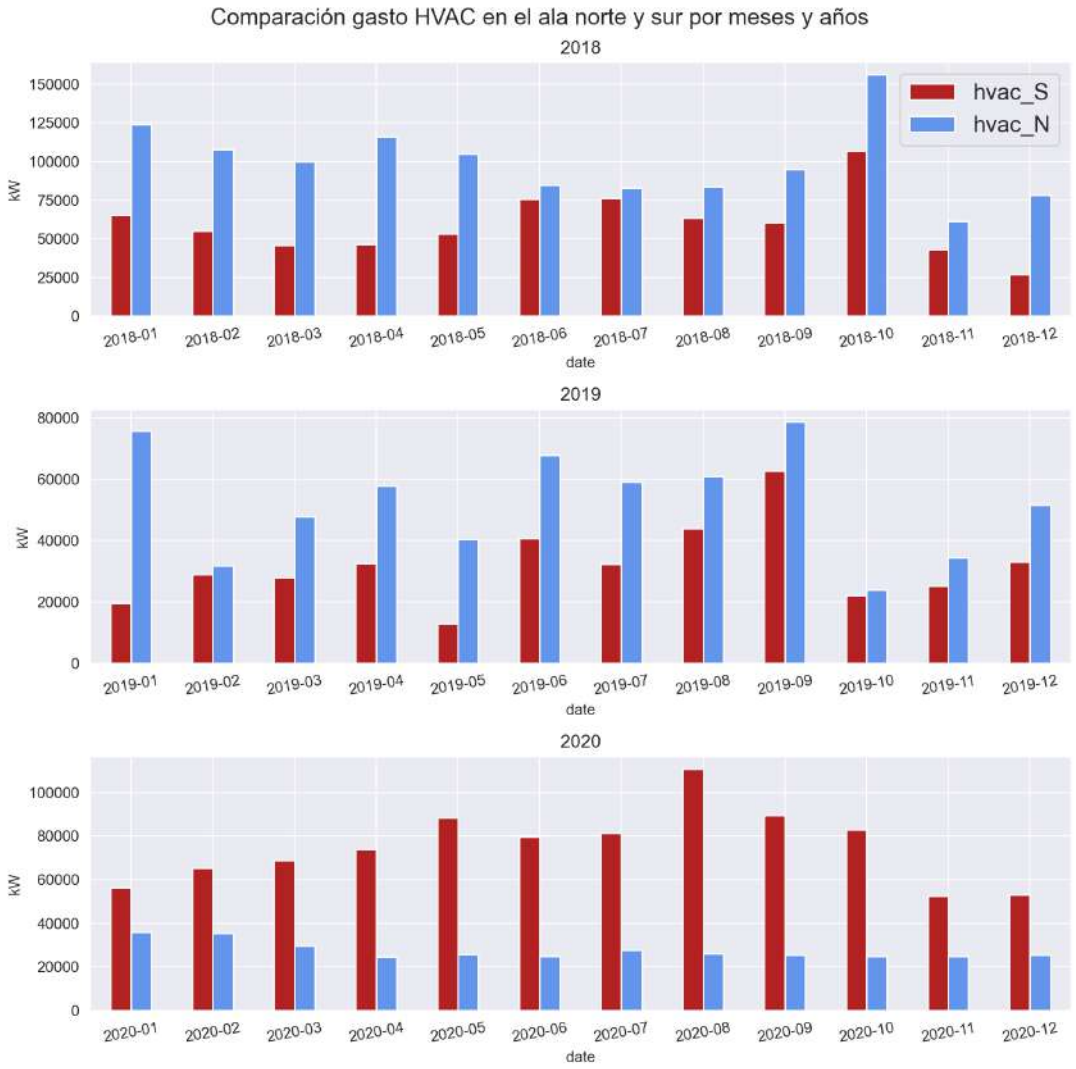

Figura 8: Comparación de gasto total HVAC en el ala norte y sur por mes y año

De manera paralela, podemos fijarnos en la Figura 8, que representa el gasto total en aire acondicionado y calefacción del ala sur y norte. En la Figura se puede ver el cambio del año 2020 con respecto a los dos años anteriores, en el que de manera drástica, el consumo por HVAC en el ala sur empieza a ser mayor que en el ala norte.

Antes de entrar en el análisis de la calidad de los datos, vamos a estudiar cómo cambia el gasto en aire acondicionado a lo largo de un día, comparando la evolución de cada mes. Para hacerlo, se ha calculado el gasto medio en cada hora del día separando el cálculo por mes. Los resultados obtenidos se pueden ver en la Figura 9.

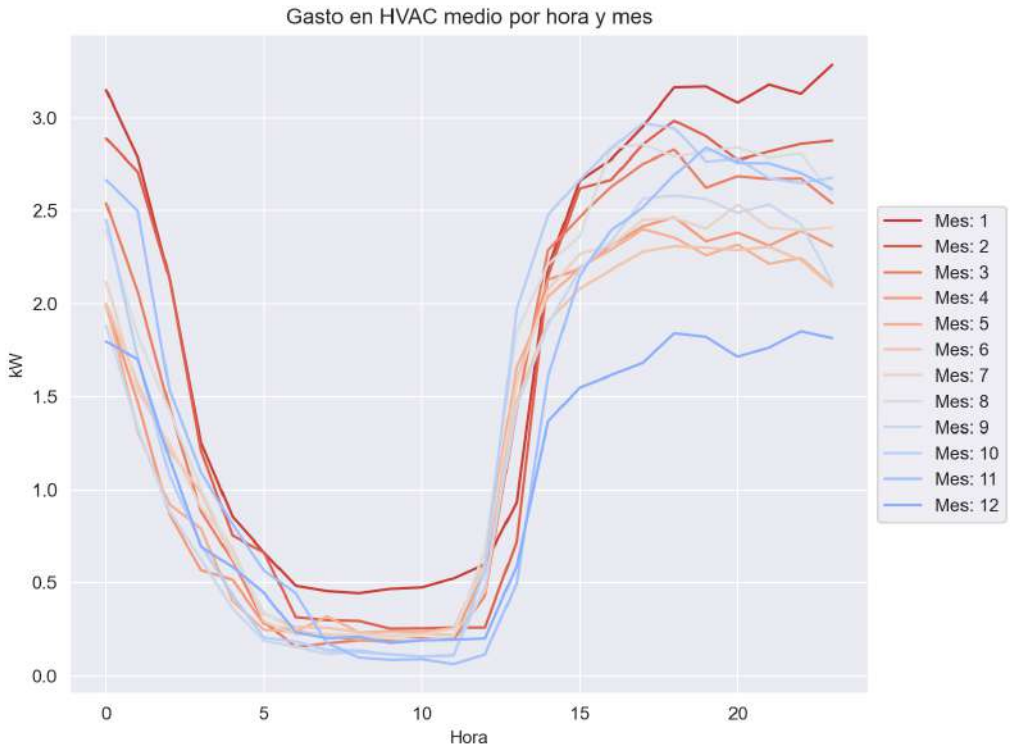

Figura 9: Comparación del gasto total HVAC por horas

Con respecto a la calidad de los datos, es necesario aclarar que existen dos tipos de errores en los datos. El primero es un error del dato en sí, es decir, el valor del dato no es correcto pero existe una observación. El segundo error se da cuando no existe una observación para un instante t determinado en el que debería haber un dato. En este estudio nos hemos fijado en tres aspectos, que se resumen en:

1. El tiempo medio entre observaciones de cualquiera de las cinco variables. Esta cuestión nos ayuda a verificar los errores del primer tipo, pues en el caso de que no exista una observación en un instante t, ninguna variable tendrá datos en ese instante.
2. El número de datos nulos de cada variable. Aunque gran parte de los errores de datos afectan a todas las variables, existen casos en los que solamente un subconjunto de ellas se ve afectada.
3. La relación entre los valores nulos de las distintas variables. Buscamos estudiar si un fallo se da de manera sistemática en un conjunto de variables.

Estas tres cuestiones se estudiarán también en los otros dos bloques de datos.

En particular, en los datos de consumo de energía, el tiempo medio entre observaciones es de 15 minutos y 19 segundos, lo que nos lleva a pensar que existen instantes de tiempo en los que debería haber datos y no los hay, haciendo que la frecuencia real sea mayor de la establecida (15 minutos exactos). Si inspeccionamos los datos, descubrimos que desde el 16 hasta el 27 de Noviembre de 2018 no se da ninguna observación. Este problema se solucionará insertando observaciones donde todas las variables tienen valores nulos, como veremos en el capítulo de imputación de datos.

Para estudiar la segunda cuestión, podemos fijarnos en la Figura 10, en la que se representa el número de datos nulos de cada variable del bloque. Se puede observar claramente cómo las variables correspondientes al sistema HVAC tienen un número de datos nulos considerablemente mayor con respecto al resto de variables. Al analizar de dónde viene este número tan elevado descubrimos que entre el 13 y el 18 de Marzo del 2018 y entre el 23 y el 31 de Agosto del 2019 no hay datos para ninguna de las dos variables. Al ser fallos localizados, podemos suponer que se dio algún fallo en el sistema o incluso que se realizó algún cambio que hizo que se tuviera que interrumpir temporalmente la recolección de datos.

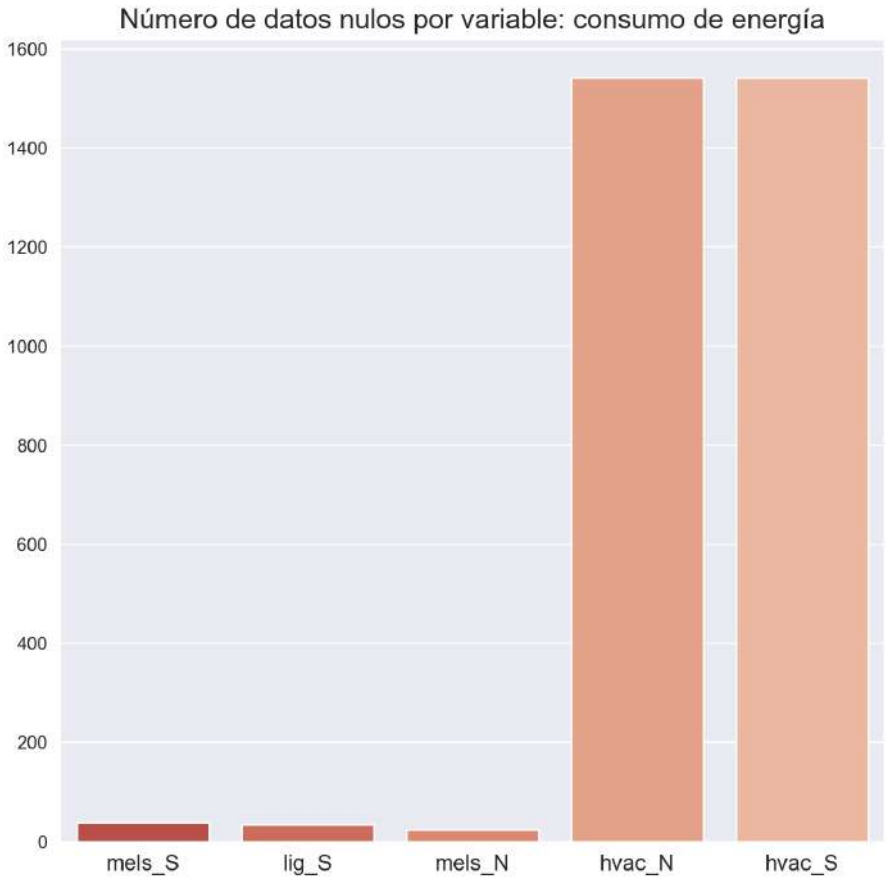

Figura 10: Comparación de datos nulos en consumo de energía

Por último, vamos a analizar la correlación entre la nulidad de las distintas variables, esto es, cómo afecta la presencia o ausencia de datos de una variable en las demás. Los valores posibles de correlación van desde -1, si cada vez que una variable tiene datos la otra no, hasta 1, si siempre que una variable tiene datos, la otra también.

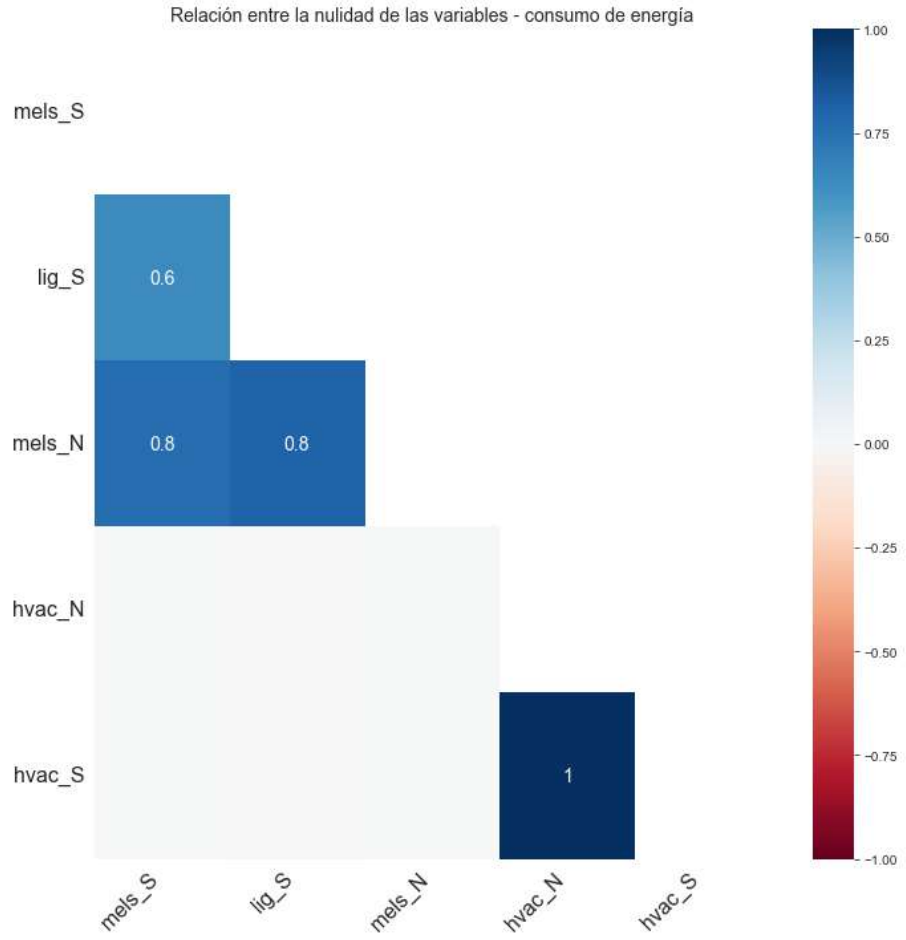

Figura 11: Correlación de nulidad en datos de consumo de energía

En este caso podemos observar que las variables *hvac_S* y *hvac_N* están perfectamente correladas, como era de esperar. Esto favorece la hipótesis de fallo completo del sistema, pues cada vez que una tiene un dato nulo, la otra también lo tiene.

De la misma manera, podemos destacar la correlación entre la nulidad de *mels_S* y *mels_N*, que no es perfecta, pero es bastante elevada.

Consideramos que la información obtenida nos permite tener una comprensión completa de cómo se comporta este conjunto de variables. Aunque todavía no se haya hecho un análisis de las propiedades de las series temporales, hemos encontrado algunas relaciones interesantes entre la nulidad de las variables que nos ayudarán en la fase de preprocesamiento. A continuación seguiremos una estructura parecida para los otros dos bloques de datos.

5.2.2 *Datos de clima exterior*

En este bloque se consideran distintas métricas del clima tomadas mediante dos sensores colocados en el exterior del edificio. Las variables consideradas son las siguientes:

- air_temp_set_1: temperatura del aire exterior según el sensor 1.
- air_temp_set_2: temperatura del aire exterior según el sensor 2.
- dew_point_temperature: temperatura del punto de rocío en el sensor 2
- relative_humidity_set_1: humedad en el sensor 1.
- solar_radiation_set_1: radiación solar exterior en el sensor 1.

De nuevo analizamos la distribución de su valor diario a lo largo de los tres años, en la figura 12:

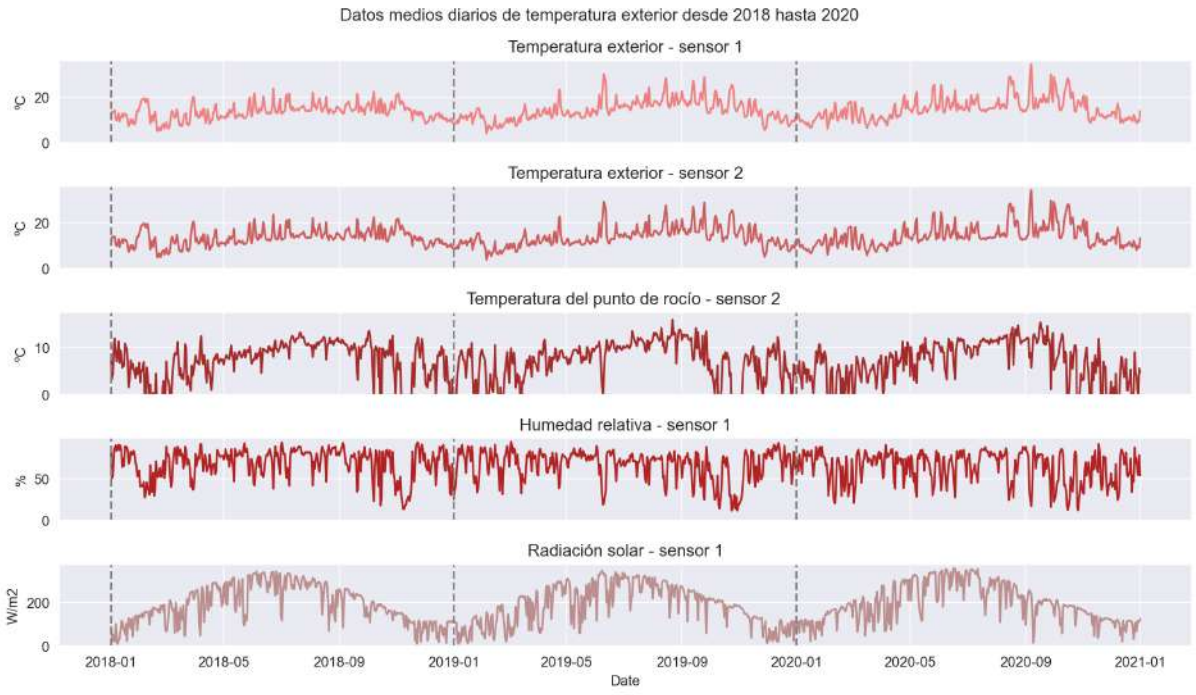

Figura 12: Distribución de las variables de clima exterior de 2018 a 2020

Podemos realizar dos observaciones bastante evidentes. La primera es que la temperatura detectada por los sensores es casi idéntica, lo que hace que aumente nuestra confianza en la calidad de los datos. La segunda observación es la periodicidad de los datos de radiación solar, donde se ve claramente el cambio a través de las distintas estaciones, por lo tanto, esta distribución puede servir como ejemplo de una serie con un componente estacionario anual.

A continuación estudiamos más a fondo la diferencia entre ambos sensores, nos interesa saber si existe alguna tendencia a la hora de recoger los datos. Para ello, analizamos la distribución a lo largo de distintos meses en los tres años (figura 13).

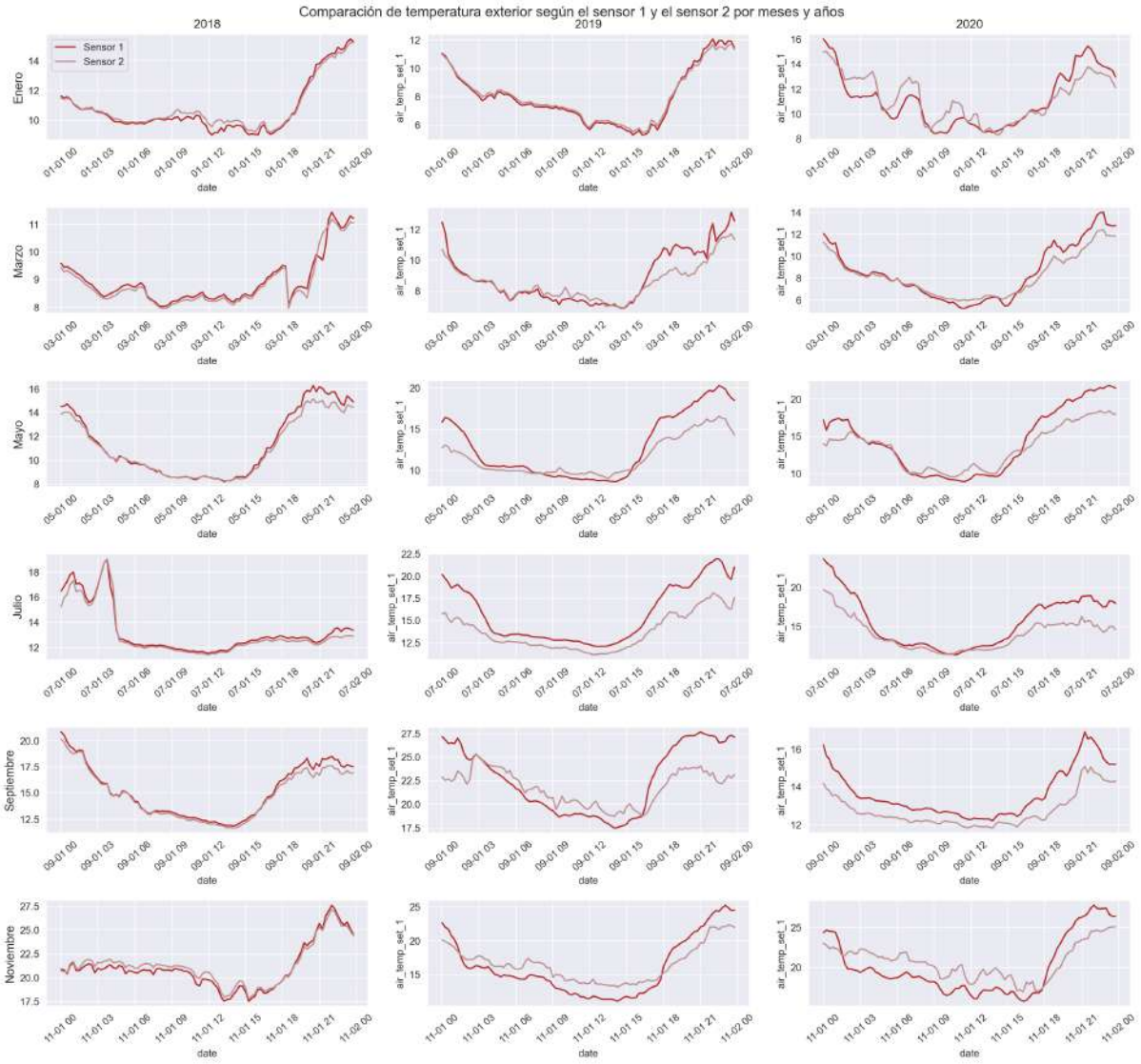

Figura 13: Comparación de la temperatura detectada por el sensor 1 y el sensor 2

Analizando el gráfico podemos concluir que en temperaturas altas, el sensor 1 tiende a leer un valor menor que el sensor 2. Resultaría interesante obtener un tercer registro, para poder comprobar cuál de los dos sensores presenta un desvío en sus mediciones. Por otra parte, se ve claramente que la distribución es muy parecida, y por lo tanto estas dos variables están altamente correladas. Es un aspecto que tendremos que tener en cuenta en los modelos de predicción que veremos más adelante.

Por último, como análisis de calidad de los datos, hemos vuelto a calcular la diferencia media entre tomas de tiempo de cualquiera de las variables, y en este caso hemos obtenido un valor de 15 minutos exactos, lo que nos indica que en el caso de tener errores, éstos son del segundo tipo. Para comprobar este tipo de errores, hemos analizado las variables en búsqueda de datos nulos o no válidos. Al no encontrar ninguno, concluimos que este bloque de datos no presenta inconsistencias. Además, si nos guiamos por el criterio de valores atípicos establecido en [19], podemos afirmar que este bloque no necesita pasar por la fase de preprocesamiento.

5.2.3 *Datos de clima interior*

En este bloque de datos se da un cambio importante en la estructura con respecto a lo que veníamos estudiando. Por una parte, la frecuencia de recolección pasa a ser de 5 y 10 minutos, y por otra parte, cada fichero de datos tiene más de una variable distinta. Cada una de las variables representa los datos obtenidos en una zona termal, por lo que las zonas que recojan los mismos datos o que estén próximas tendrán una distribución similar. Es por este motivo por el que no vamos a estudiar una a una las variables por separado.

La calefacción y la refrigeración son proporcionadas por un sistema de redistribución de aire bajo el suelo que utiliza unidades de aire acondicionado (RTU, por sus siglas en inglés) situadas en el techo de cada zona. Las zonas estudiadas se pueden dividir según la unidad RTU a la que pertenezcan. En la tabla 2 podemos observar dicha división

Ala	RTU	Zonas Termales
Norte	1	36, 37, 38, 39, 40, 41, 42, 64, 65, 66, 67, 68, 69, 70
Norte	2	19, 20, 27, 28, 29, 30, 31, 32, 33, 34, 35, 43, 44, 49, 50, 57, 58, 59, 60, 62, 63, 71, 72
Sur	3	18, 25, 26, 45, 48, 55, 56, 61
Sur	4	16, 17, 21, 22, 23, 24, 46, 47, 51, 52, 53, 54

Cuadro 2: División de zonas termales en función de la unidad RTU a la que pertenecen

A continuación explicaremos qué tipos de datos almacena cada fichero y analizaremos su estructura particular.

5.2.3.1 *Punto de ajuste de la temperatura de refrigeración*

El primer fichero de datos es *zone_temp_sp_c.csv*, que almacena los datos sobre el punto de ajuste de la temperatura de refrigeración de cada zona. El punto de ajuste se toma como una temperatura objetivo, a la cual se deberá mantener la zona. Al ser temperatura de refrigeración, asumimos que el sistema HVAC se activará hasta que la zona particular alcance dicha temperatura.

A continuación vamos a estudiar el punto de ajuste de refrigeración de cuatro zonas distintas a lo largo de los tres años, donde cada una de ellas pertenece a una unidad RTU de la tabla 2. Las zonas escogidas serán las zonas 18, 19, 21 y 36. Podemos verlo en la Figura 14.

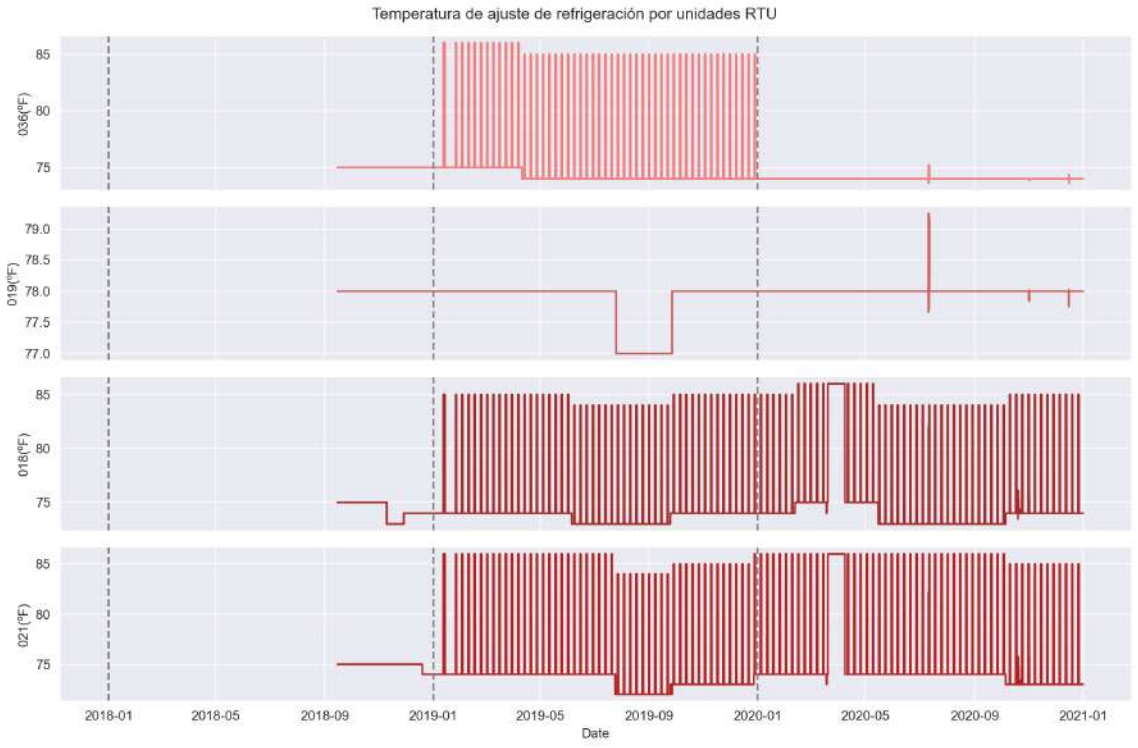

Figura 14: Comparación de la temperatura de ajuste de refrigeración por unidades RTU

Se puede observar cómo los valores de la temperatura de ajuste son prácticamente discretos, y tienen ciertos valores que parecen ser predefinidos. Además, vemos que solamente hay datos desde Septiembre del 2018, este será uno de los aspectos más importantes a la hora de definir el intervalo de datos considerado para los modelos de predicción.

Para ver cómo cambia la temperatura de ajuste a lo largo de las horas del día en cada uno de los meses del año podemos analizar la Figura 15. Tras analizarla, llegamos a la conclusión que la distribución de la zona 18 es distinta. Esto puede ser por dos motivos: el primero es que el sensor de la zona 18 no sea fiable, y el segundo, que el punto de ajuste de esa unidad de RTU siga una distribución distinta a las demás. Podemos concluir, al haber analizado el resto de las zonas de esa unidad RTU, que la distribución es generalizada, y por tanto no se debe a un error del sensor de la zona 18.

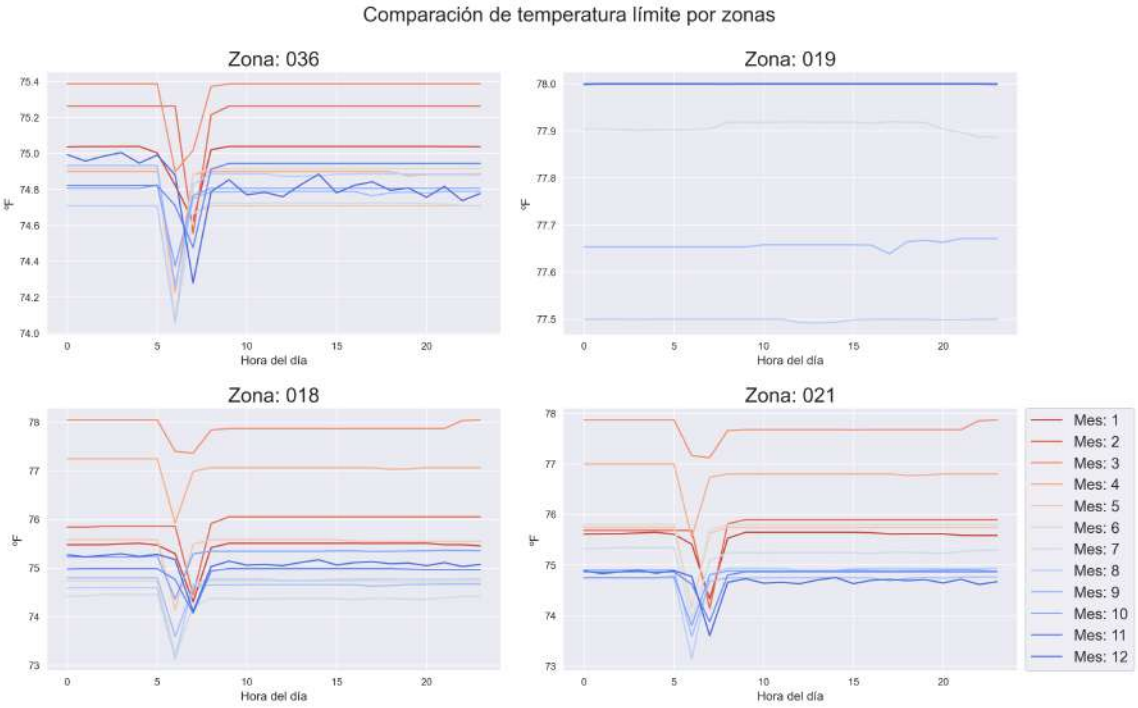

Figura 15: Comparación de la temperatura de ajuste de refrigeración por meses y horas

Finalmente, aplicando el estudio anterior de calidad de datos, obtenemos que la distancia media entre tomas de datos es de 5 minutos y 5 segundos, con lo que deducimos existen errores del primer tipo que vimos. Además, si analizamos la figura 16 podemos confirmar que la mayoría de errores son del primer tipo, pues los números de datos nulos suponen una fracción mínima de los datos totales (en torno al 5 %).

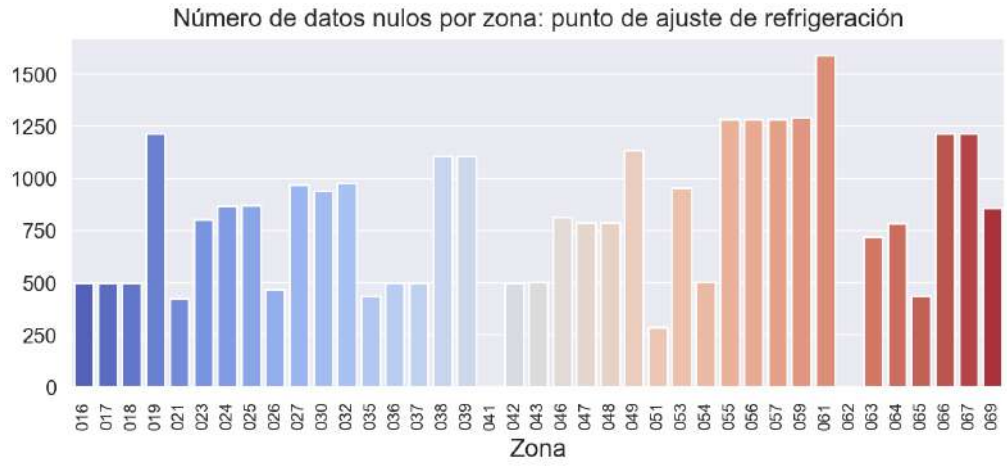

Figura 16: Número de datos nulos por zona - temperatura de ajuste de refrigeración

5.2.3.2 *Punto de ajuste de la temperatura de calefacción*

El siguiente conjunto de datos sigue una estructura paralela al que acabamos de estudiar. La única diferencia es que en vez de representar el punto de ajuste de la refrigeración, representa el punto de ajuste de la calefacción. Esto quiere decir que si la temperatura real está por debajo de la del punto de ajuste, se activará el sistema de calefacción.

De nuevo, comprobamos la distribución de los valores del punto de ajuste de las zonas 18, 19, 21 y 36. Se pueden ver los resultados en la Figura 17.

A simple vista, la distribución que vemos es la misma que en la Figura 14, pero tras fijarnos en detalle comprobamos que los valores de los puntos de ajuste de calefacción son más bajos, como era de esperar.

Dentro de este bloque de datos también podemos analizar la evolución del punto de ajuste de cada unidad RTU a lo largo del día en cada mes. Estos datos están representados en la Figura 18.

Por último, repetimos el análisis de la calidad de los datos, obteniendo de nuevo una distancia media entre observaciones de 5 minutos y 5 segundos, confirmando los errores de primer tipo que encontramos en los datos de punto de ajuste de refrigeración.

La última Figura que vamos a analizar es 19, que representa una vez más el número de datos nulos por variable en este conjunto de datos. Comprobamos que casi todas las variables presentan errores de segundo tipo, si bien éstos conforman una parte muy pequeña de los datos totales.

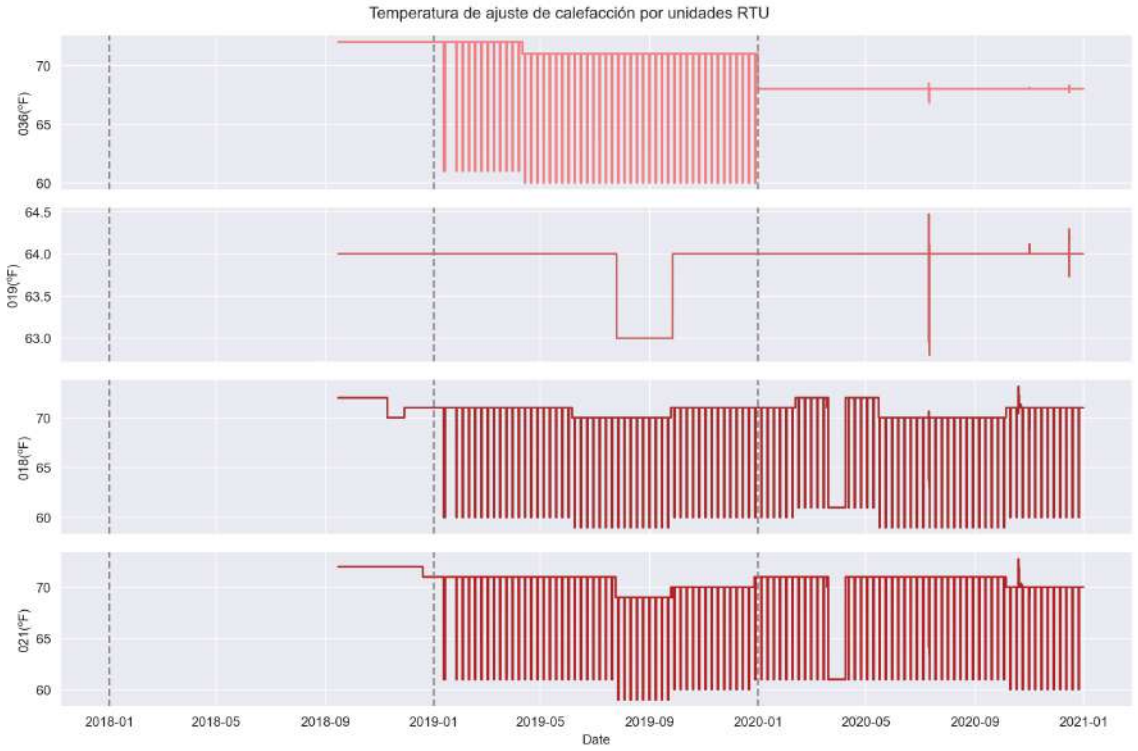

Figura 17: Comparación de la temperatura de ajuste de calefacción por unidades RTU

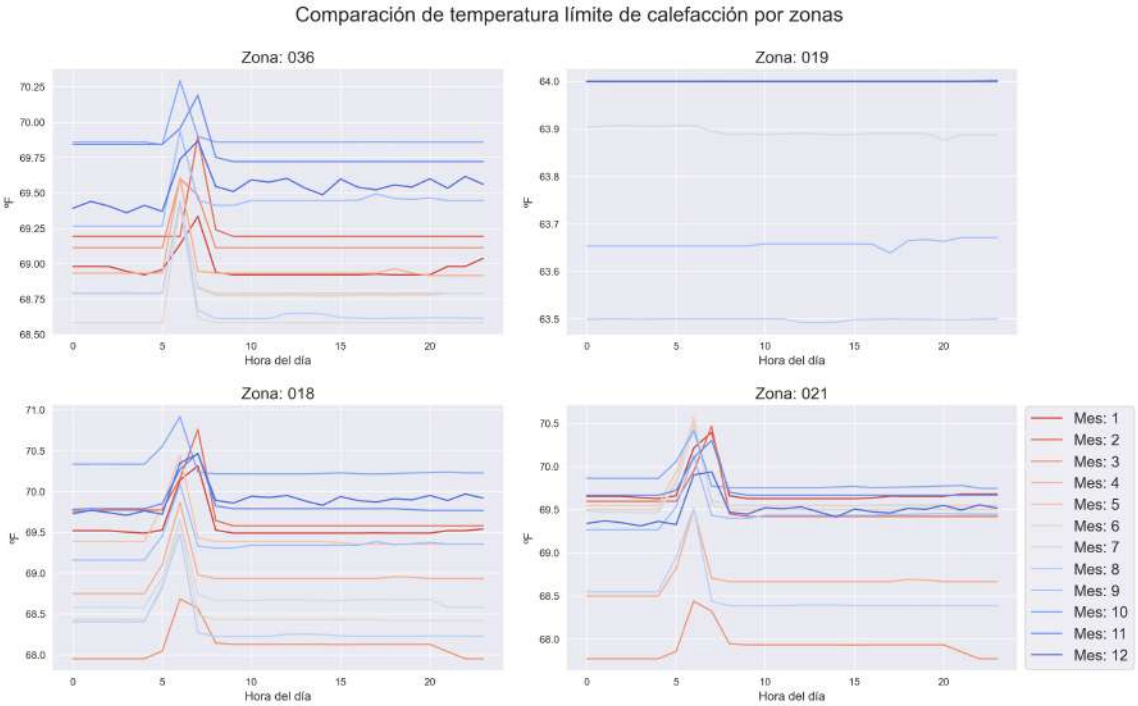

Figura 18: Comparación de la temperatura de ajuste de calefacción por meses y horas

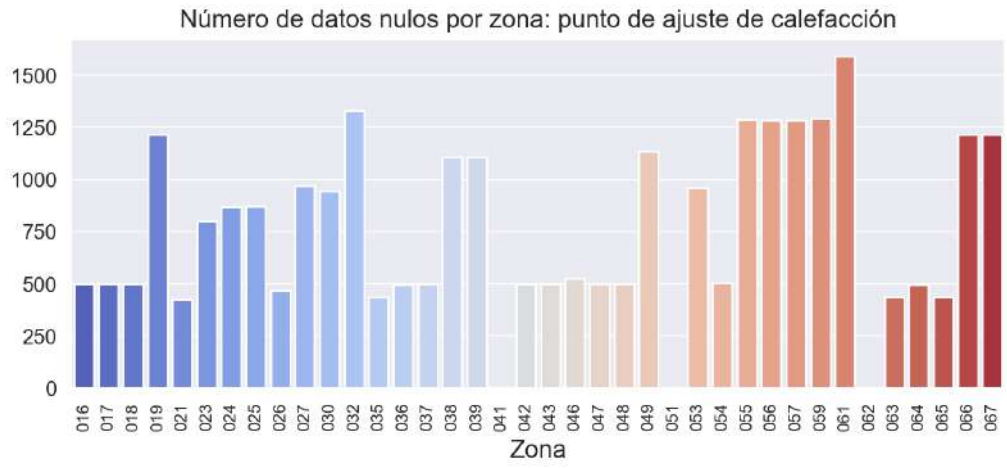

Figura 19: Número de datos nulos en los datos de punto de ajuste de calefacción

5.2.3.3 *Temperatura interior*

Siguiendo el mismo esquema que en los bloques de datos anteriores, vamos a estudiar los datos de temperatura interior. Cabe destacar que la frecuencia establecida para estos datos es de 10 minutos, y que de nuevo tenemos datos multidimensionales, en este caso tenemos 16 variables que representan las lecturas de cada uno de los sensores denominados *cerc_templogger* de la Figura 4, cada uno situado en una zona termal interior.

Podemos agrupar estos sensores en función de la planta en la que se encuentren y en función de si están situados en el ala norte o en el ala sur. De la misma manera que en el apartado anterior, para realizar el análisis vamos a escoger un sensor aleatorio de cada ala dentro de cada piso, obteniendo un total de cuatro sensores.

La distribución de la temperatura interior a lo largo de los tres años se puede ver representada en la Figura 21.

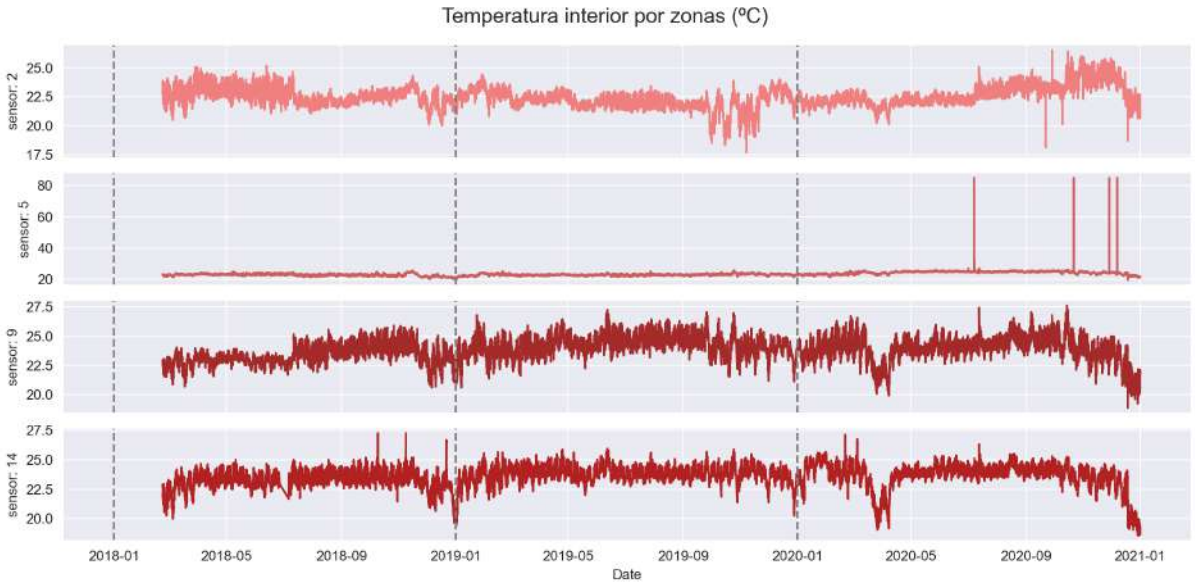

Figura 20: Evolución de temperatura interior 2018-2020

Es necesario destacar que en [19] se define la unidad de estos datos como ºF, sin embargo, debido a que la mayoría de los datos se encuentran en el intervalo de 20 a 25 (Figura 20), podemos afirmar sin duda alguna que los datos están presentados en ºC.

Al observar la evolución de la temperatura recogida por el sensor 5, podemos observar unos valores que están por encima de 80, y que resultan relativamente inconsistentes con los que veníamos observando con anterioridad. Este tipo de valores se pueden clasificar como datos atípicos, y se procesarán de manera adecuada más adelante.

También es interesante observar la evolución de la temperatura interior a lo largo de cada día. Para analizarlo se ha separado los datos por meses, para poder comprobar el efecto estacional (Figura 21).

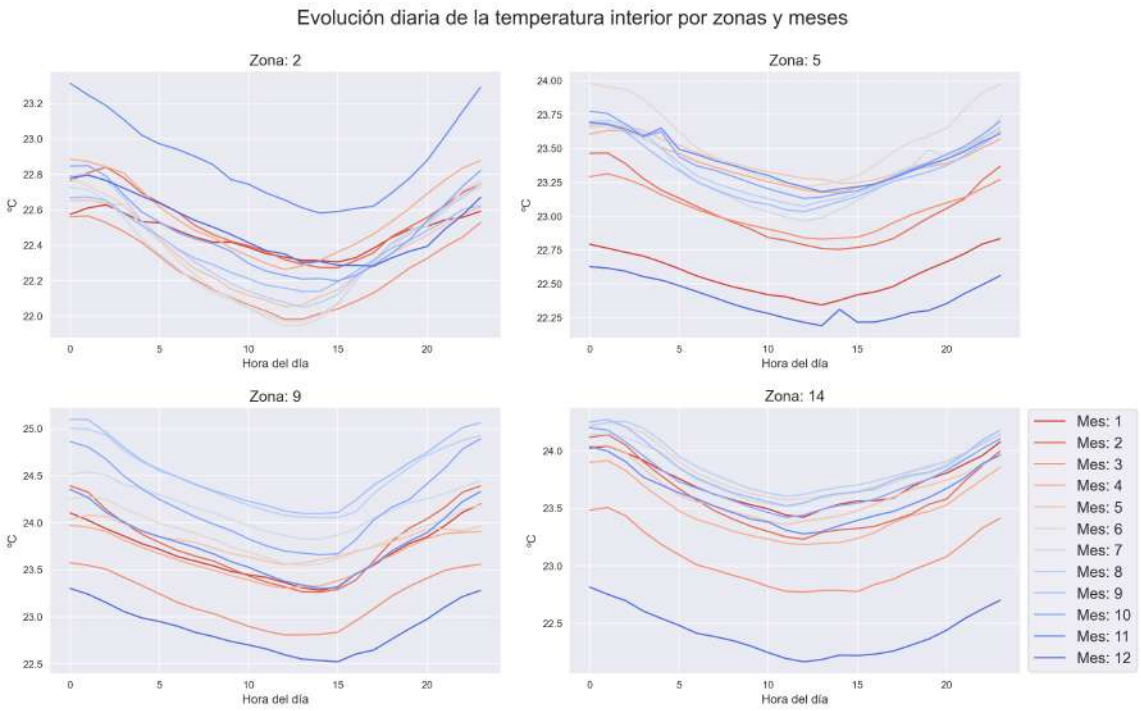

Figura 21: Temperatura interior por horas y meses

Se puede comprobar que la diferencia a través de los meses es bastante clara, aunque es destacable la falta de consistencia sobre los meses más cálidos y más fríos en cada uno de los sensores. Esto puede ser debido a la orientación de la sala en la que se encuentren.

Con respecto a la calidad de los datos, la diferencia media de tiempo obtenida es de 10 minutos y 2 segundos, por lo que podemos deducir que se consideran la mayoría de instantes preestablecidos. Por otra parte, al analizar el número de datos nulos por variable (Figura 22), vemos que existe un sensor (número 15), con alrededor de 25000 datos nulos. Aunque puede parecer excesivo, si analizamos la proporción frente al número de datos totales obtenemos que conforman únicamente un 16 %. Más adelante será necesario tomar la decisión de si mantenemos esta variable o la descartamos por no tener la calidad suficiente.

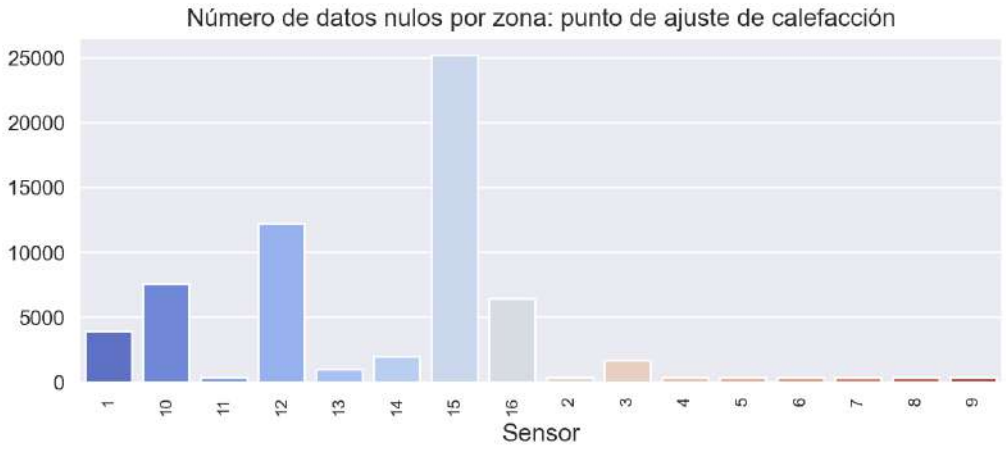

Figura 22: Número de datos nulos por variable - temperatura interior

5.2.3.4 *Temperatura exterior*

El último bloque de datos que vamos a presentar es la temperatura de las zonas termales exteriores. En este bloque la frecuencia de tomas de datos es de 1 minuto, y además incorpora 51 variables, correspondientes cada una a una zona. Para realizar el análisis vamos a mantener las zonas estudiadas en bloques anteriores: 18, 19, 21 y 36. Podemos observar su evolución a lo largo de los tres años en la Figura 23.

Antes de continuar, hay que notar que en estos datos hemos encontrado un error distinto a los considerados anteriormente, y se da cuando para un instante t tenemos varios datos. En este caso particular era necesario solucionarlo antes de generar las gráficas siguientes, y para ello hemos agrupado por t y calculado la media.

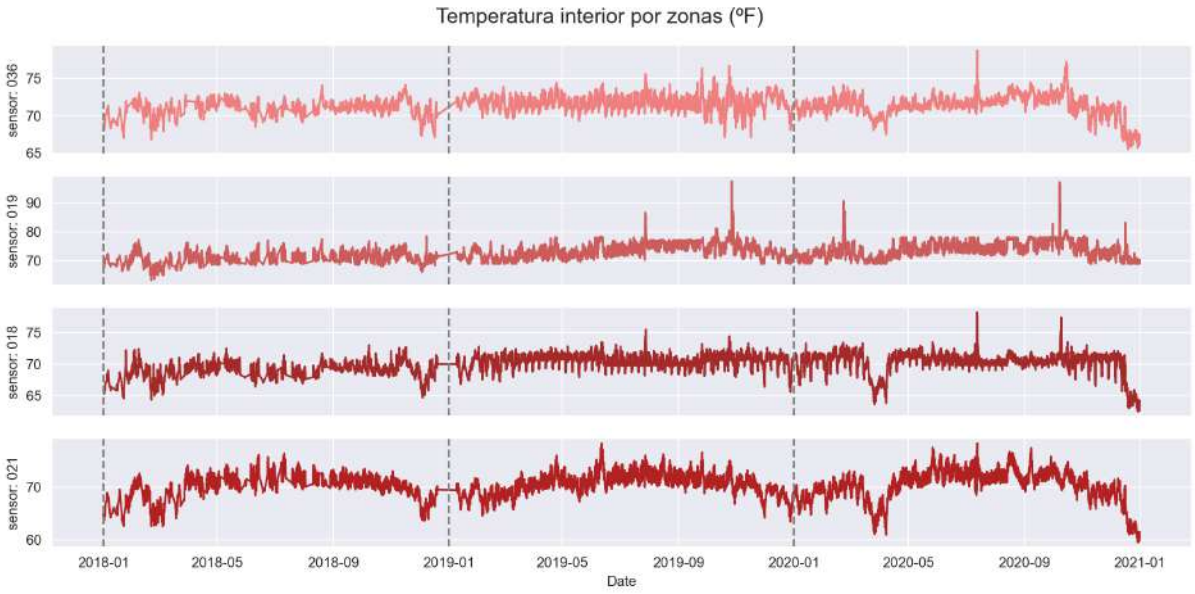

Figura 23: Número de datos nulos por variable - temperatura interior

Es necesario destacar la unidad de medida, que en este caso sí se corresponde a la predeterminada, ºF, al estar la mayoría de los datos entre 70 y 80. Para comprobar el cambio de la evolución de la temperatura a lo largo de los meses, estudiamos la Figura 24. En la gráfica podemos comprobar que la variabilidad diaria es mucho menor, no se destacan especialmente las horas nocturnas, y además, los meses fríos y calientes son consistentes en las cuatro zonas estudiadas.

Por último, al hacer el análisis de la calidad de los datos una vez más, obtenemos que la distancia media entre tomas de datos es de 20 segundos. Esta cifra es considerablemente menor a la establecida (1 minuto), por lo que en un futuro será necesario preprocesarlo correctamente.

Con respecto al número de valores nulos (Figura 25), destaca la zona 70, que presenta 356000 datos nulos. Al comprobar la proporción de los datos nulos frente al total, obtenemos que conforman un 24 % de los datos totales. Esta cifra es significativa, y aunque el problema se pudiera solucionar imputando los datos faltantes, no tenemos suficiente confianza en esta variable, por lo que más adelante la eliminaremos.

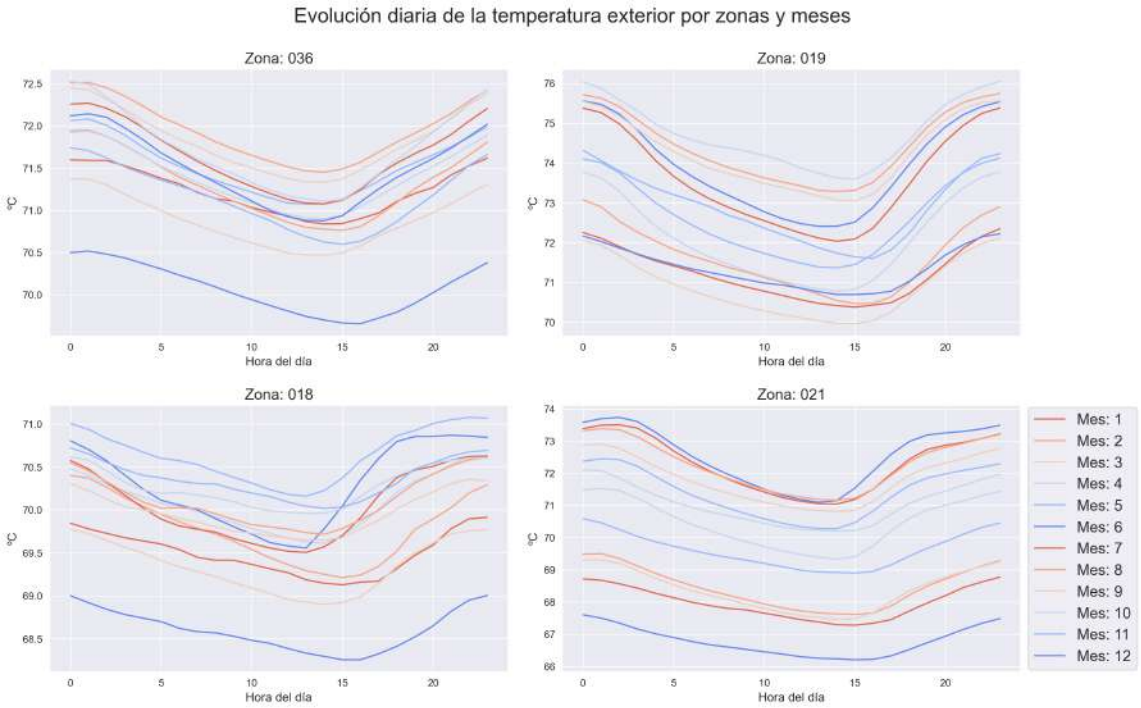

Figura 24: Evolución diaria de la temperatura por zonas (exteriores) y meses

Figura 25: Evolución diaria de la temperatura por zonas (exteriores) y meses

6

PREPROCESAMIENTO DE LOS DATOS

En el análisis y predicción de series temporales, es crucial asegurar que los datos utilizados sean precisos y fiables. En este capítulo se presentará en detalle el proceso que hemos seguido para obtener datos limpios a partir de los datos en brutos estudiados en el capítulo anterior.

En primer lugar, se explicará cómo se han localizado y tratado los distintos errores, tales como la falta de datos del instante de tiempo t o la presencia de datos inválidos, que pueden ser datos nulos o con valores atípicos. Por último, pasaremos a explicar en qué consiste el algoritmo de limpieza y compararemos los datos obtenidos con los originales.

6.1 PROCESAMIENTO DE ERRORES

En el capítulo anterior se introdujeron los dos tipos principales de errores y se hizo referencia a datos no válidos como uno de ellos. Sin embargo, no se proporcionó una definición precisa de lo que se considera un dato no válido. A continuación profundizaremos en este concepto y estableceremos los criterios necesarios para identificar este tipo de errores en cada variable.

6.1.1 *Outliers*

Una dato atípico o **outlier** es aquel que se desvía de manera significativamente del resto de observaciones de la muestra en la que se encuentra [12]. Resulta natural plantear dos alternativas:

1. El outlier es una manifestación de una gran variabilidad aleatoria presente en los datos. En estos casos, los datos atípicos contienen información valiosa de la muestra y no se deben descartar.
2. Por otra parte, el outlier puede ser el resultado de un error o desviación en alguna fase del proceso de obtención de la muestra. Por lo tanto, la información que aporta el dato atípico es información errónea y deberá descartarse.

Es necesario establecer criterios para determinar la existencia y ubicación de este tipo de datos. Aunque existen técnicas estadísticas refinadas, en este estudio vamos a estudiar la más utilizada a día de hoy, por su sencillez y facilidad de comprensión. Esta técnica utiliza un concepto estadístico que no se ha introducido hasta el momento: los cuantiles.

Un cuantil es una medida de posición que generaliza a la mediana. Su definición formal es la siguiente

Definición 6.1.1. Diremos que $c_q \in \mathbb{R}$ es un cuantil de orden q ($q \in (0,1)$) de una variable aleatoria X si

$$P(X \leq c_q) = q \quad \text{y} \quad P(X \geq c_q) = 1 - q \tag{6.1.1}$$

Los cuantiles son medidas que dividen el rango de una distribución de probabilidad en intervalos equiprobables. Por ejemplo, el cuantil de orden 0.5, también conocido como mediana, divide la muestra en dos partes iguales. La mitad de los datos estarán por encima del valor de la mediana y la otra mitad estarán por debajo.

Para poder definir un criterio que separe los outliers en la muestra, utilizaremos un caso particular de cuantil, los cuartiles. Los cuartiles dividen la muestra en cuatro partes iguales. El primer cuartil, Q_1, es el cuantil de orden 0.25, es decir, divide la muestra en dos partes, una con el 25 % de los datos más bajos y otra con el 75 % restante. El segundo cuartil Q_2 es la mediana, que divide la muestra en dos partes iguales. El tercer cuartil, Q_3, es el cuantil de orden 0.75, divide la muestra en dos partes, una con el 75 % de los datos más bajos y otra con el 25 % restante.

Tras haber comprendido este concepto, podemos definir el **rango intercuartil** (IRQ, por su siglas en inglés), que no es más que la diferencia entre el valor de Q_3 y Q_1. El criterio para definir un outlier se define como sigue: si el dato no está en el intervalo $[Q_1 - 1{,}5 \quad IQR, Q_3 + 1{,}5 \quad IQR]$ se considerará un dato atípico.

Esta información se puede ver resumida generando el diagrama de caja asociado a los datos. Este diagrama, también conocido como *boxplot*, es una herramienta gráfica que se utiliza para representar la distribución de un conjunto de datos numéricos.

Consiste en una caja que se extiende desde el primer cuartil hasta el tercer cuartil de los datos, con una línea en medio que representa la mediana. También se incluyen líneas llamadas bigotes que representan el intervalo $[Q_1 - 1{,}5 \quad IQR, Q_3 + 1{,}5 \quad IQR]$.

En esta sección únicamente vamos a estudiar el diagrama de caja de los dos primeros bloques de datos, debido a la amplitud del tercer bloque. Sin embargo, el análisis se realizará también sobre ellos, utilizando el mismo criterio y tratamiento de los datos atípicos.

En la Figura 26 podemos observar el diagrama de caja del primer bloque de datos. Se ve claramente como en todas las variables hay datos que se encuentran fuera del intervalo considerado, por lo que podemos afirmar que son outliers. Sin embargo, es necesario estudiar si se tratan de valores atípicos del primer o del segundo tipo.

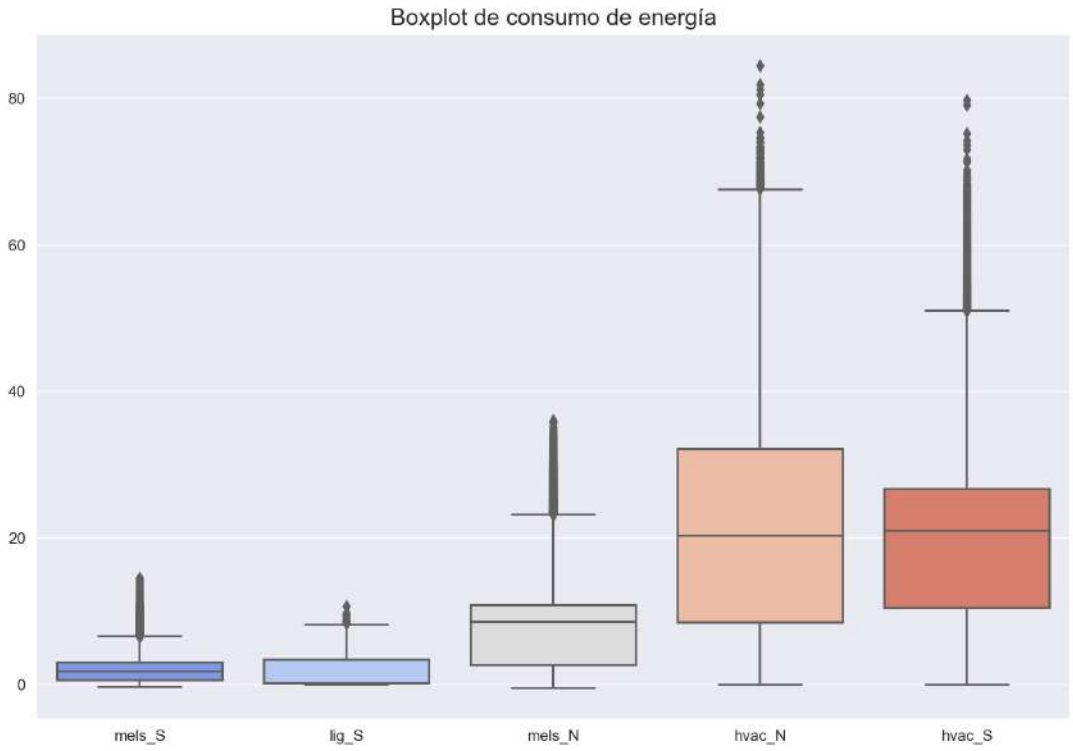

Figura 26: Diagrama de caja de datos de uso de energía

Esta decisión depende casi exclusivamente del contexto, y en este caso resulta complicado dar una respuesta clara pues la información que tenemos respecto del edificio es limitada. En [19] el criterio establecido para localizar datos erróneos en este bloque es que éstos se encuentren por debajo de 0. Tras estudiar la distribución de este tipo de valores en otros conjuntos de datos, vamos a tomar este criterio como bueno, por lo que no será necesario eliminar ningún dato.

En la Figura 27 podemos observar el diagrama correspondiente a los datos de clima exterior. En primer lugar, destaca la distribución de la variable que representa la radiación solar. Al estudiar los datos medios de radiación solar en Berkeley, podemos afirmar que los datos que están fuera del intervalo son correctos, y aunque sean valores extremos, no debemos quitarlos. Para poder analizar el resto de variables, vamos a repetir el diagrama eliminando la radiación solar (Figura 28).

En este último diagrama se puede comprobar que la única variable que no contiene datos atípicos es la que contiene datos de humedad relativa. Para el resto de variables, será necesario realizar un análisis más profundo, teniendo en cuenta los siguientes aspectos:

- La temperatura de punto de rocío no debe ser superior a la temperatura del aire, pues de ser así la humedad relativa sería superior a 100.
- La temperatura máxima en Berkeley, California en cada uno de los tres años analizados está en torno a los 38ºC.

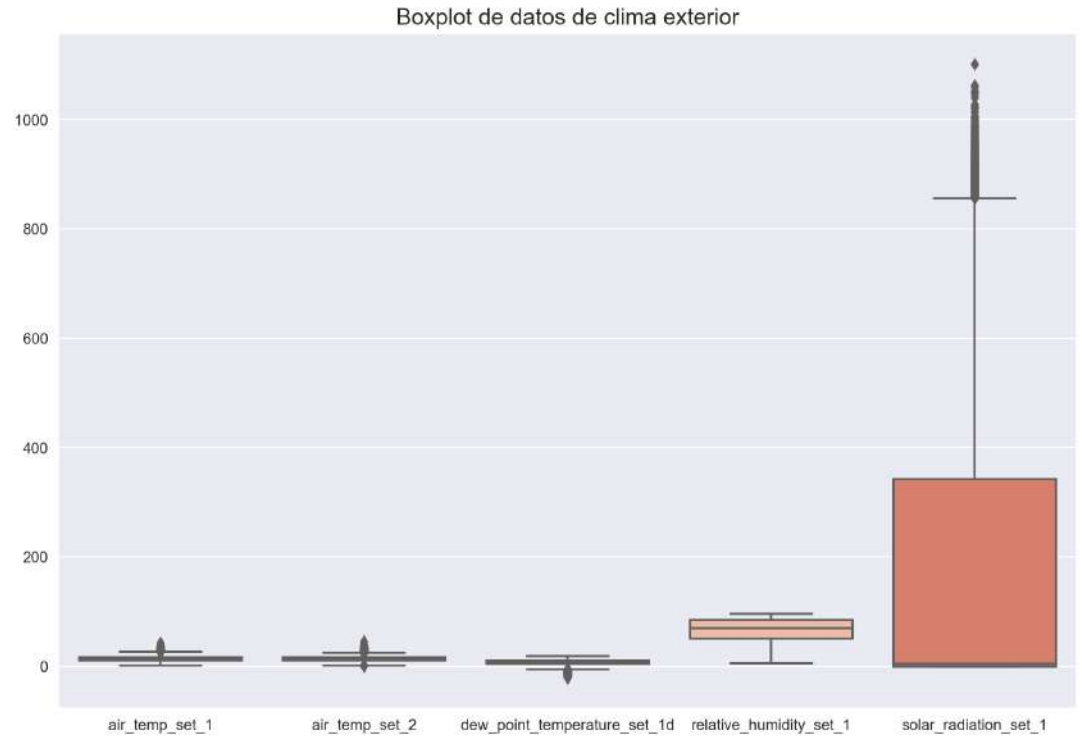

Figura 27: Diagrama de caja de datos de clima exterior

- Una temperatura de punto de rocío se debería dar únicamente en temperaturas bajas.

La primera y la última afirmación tiene que ver con el punto de rocío y es necesario asegurar que se cumple. Al buscar instantes de tiempo en los que la temperatura de punto de rocío sea superior a cualquiera de las mediciones de temperatura del aire, no obtenemos ninguna observación, por lo que la primera afirmación se cumple.

Para la segunda propiedad se puede observar que los valores máximos de la temperatura del aire están en torno a los 46ºC, valor superior a 38ºC. Sin embargo, al contar el número de observaciones que se encuentran por encima de 38ºC descubrimos que únicamente hay 35 entre los dos sensores, lo que representa una ínfima parte del total.

Para solucionar ambos problema, eliminaremos los datos, sustituyéndolos por un valor NaN que se imputará de manera más precisa al ejecutar el algoritmo descrito en la sección siguiente.

Por último estudiamos la relación entre temperatura de punto de rocío negativa y temperatura del aire. Podemos determinar que temperaturas de punto de rocío negativas no implican temperaturas bajas y por tanto, no se cumple la segunda afirmación. Esto supone un problema de calidad de datos, pues las temperaturas de punto de rocío negativas (y por tanto erróneas) conforman aproximadamente un 10 % de las observaciones totales.

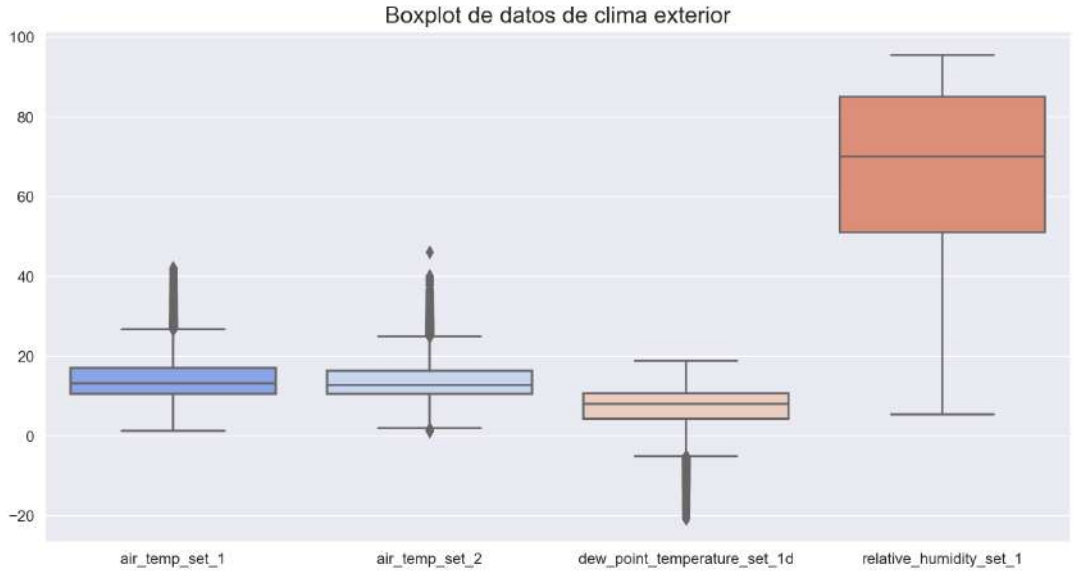

Figura 28: Diagrama de caja de datos de clima exterior sin radiación solar

Sin embargo, existe una relación entre la humedad relativa, la temperatura y la temperatura de punto de rocío. Si consideramos una temperatura T, y humedad relativa H, entonces la temperatura de punto de rocío viene R dada por:

$$R = T - \frac{100 - H}{5} \tag{6.1.2}$$

Utilizaremos esta fórmula para calcular los valores de temperatura punto de rocío utilizando la media de las temperaturas leídas en los sensores, tras imputar los valores mencionados anteriormente.

En la Figura 29 podemos observar la comparación entre los valores calculados y los originales. Se puede comprobar que los valores son prácticamente idénticos, excepto en los valores extremos, donde los datos calculados son más consistentes. De ahora en adelante consideraremos como correctos los calculados, pues cumplen con las restricciones naturales sobre la temperatura de punto de rocío.

Antes de introducir el algoritmo, es necesario determinar cómo se han solucionado los problemas del primer tipo: instantes de tiempo faltantes que hacen que la frecuencia sea distinta de la original. Para resolverlos, primero se han introducido los instantes de tiempo faltantes, con valores nulos en todas las variables. Tras imputar los datos correspondientes, si para un periodo T existe más de una muestra de datos, se calcula la media entre ambas.

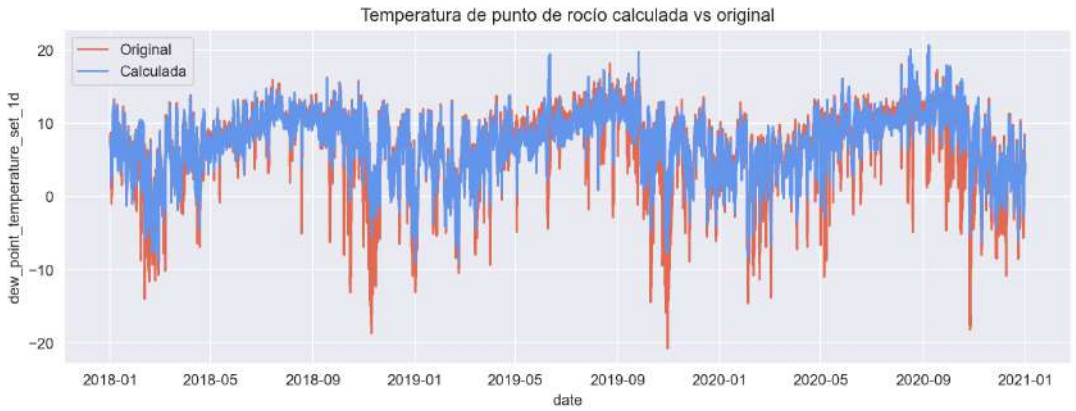

Figura 29: Comparación datos calculados vs originales

6.2 ALGORITMO DE IMPUTACIÓN

En esta sección, se explicará el algoritmo de imputación desarrollado para completar los datos faltantes en el conjunto de partida. Intentamos inicialmente ejecutar el algoritmo de imputación propuesto en [19], pero nos enfrentamos a problemas de memoria que lo hicieron imposible. En consecuencia, decidimos rediseñar el algoritmo de manera que pudiera adaptarse a las limitaciones de nuestro sistema. Se han utilizado técnicas de Machine Learning para imputar los datos y obtener resultados más precisos. El proceso completo de desarrollo junto con todas las pruebas realizadas se encuentran en la parte final de este estudio.

Antes de presentar el algoritmo completo, es necesario introducir el estudio en el que nos hemos basado ([23]) para implementarlo. En él, se propone una manera de imputar datos en series temporales univariables mediante técnicas de Machine Learning realizando una transformación previa que genere datos multidimensionales, el algoritmo se denomina MLBUI por sus siglas en inglés. A continuación lo describimos en detalle.

6.2.1 *Algoritmo de imputación en series univariables*

A pesar de la gran cantidad de variables presentes en el conjunto de datos, se optó por realizar una imputación univariable, lo que significa que los datos se imputan variable por variable, sin tener en cuenta las demás. Esta decisión se tomó por dos razones importantes.

En primer lugar, la frecuencia de las variables no es uniforme, y aunque podríamos unificarlas antes de la imputación, esto podría llevar a la pérdida de información valiosa. En segundo lugar, para cada variable en el conjunto de datos, los datos de su mismo bloque son los que más información pueden proporcionar para imputar los valores nulos, ya que están directamente relacionados con ella. Si examinamos la correlación entre los datos nulos dentro de cada bloque, es posible observar que están

altamente correlacionados. Esto significa que si un valor es nulo en una variable, es muy probable que también lo sea en el resto de ellas, y por tanto, no se deberían utilizar para realizar la imputación.

Partimos de una serie temporal X_t incompleta, o equivalentemente, que presenta huecos formados por datos nulos. Vamos a describir el proceso de imputación de un solo hueco, al ser idéntico en todos los demás. Para imputar todos los huecos de una variable en particular, se aplicará el algoritmo sobre el primero de ellos, y de manera secuencial se irá ejecutando sobre los demás.

Vamos a fijar una notación que facilitará la comprensión del algoritmo. Consideramos t_0 como el índice de la primera observación nula y T como el número de datos nulos presentes en el hueco, es decir, la longitud del hueco. La aproximación descrita se compone de cinco fases:

Primera fase

En esta fase se extraen dos series temporales a partir de la serie original:

- D_{b_t} es la serie temporal correspondiente a los datos anteriores al hueco, es decir, incluye los datos desde el inicio de la serie hasta t_0.
- D_{a_t} es la serie que representa los datos posteriores, comenzando en el primer dato no nulo a partir de t_0.

Es necesario realizar una observación, y es que al ejecutar el algoritmo de manera secuencial sobre los huecos, es decir, empezando por el primero y procediendo en orden desde él, siempre se puede considerar el inicio de D_{b_t} como el inicio de la serie original, es decir, que no habrá huecos anteriores a t_0. No podemos decir lo mismo respecto del punto final de D_{a_t}, pues después del hueco actual puede haber otros, y es fundamental que tanto D_{b_t} como D_{a_t} sean series temporales sin datos nulos.

Para lidiar con esta incertidumbre se establecerán más adelante criterios que generalizan el algoritmo. Por el momento, únicamente es necesario asumir que la longitud de D_{b_t} y de D_{a_t} es superior a T. Vamos a definir N como el índice final de D_{a_t} bajo esta suposición.

Segunda fase

Esta es la fase en la que se realiza una transformación sobre D_{b_t} y D_{a_t} para obtener dos series temporales multivariables, de dimensión T. La idea tras esta transformación es utilizar los T valores anteriores disponibles para predecir el valor de $T+1$.

Por tanto, será necesario generar dos conjuntos de datos para cada una de las series:

- Conjunto T-dimensional (T variables) con los valores que servirán como entrada para el algoritmo de Machine Learning.
- Valores correspondientes a la predicción del algoritmo, que no son más que los valores desde $T+1$ hasta el índice final de cada serie.

El último detalle importante de esta fase es que los valores de D_{a_t} se invierten, es decir, se toman los últimos T valores de la serie para predecir el valor anterior. Esto se consigue invirtiendo la serie temporal antes de realizar la transformación.

Podemos ver este proceso de manera más gráfica en la Figura 30.

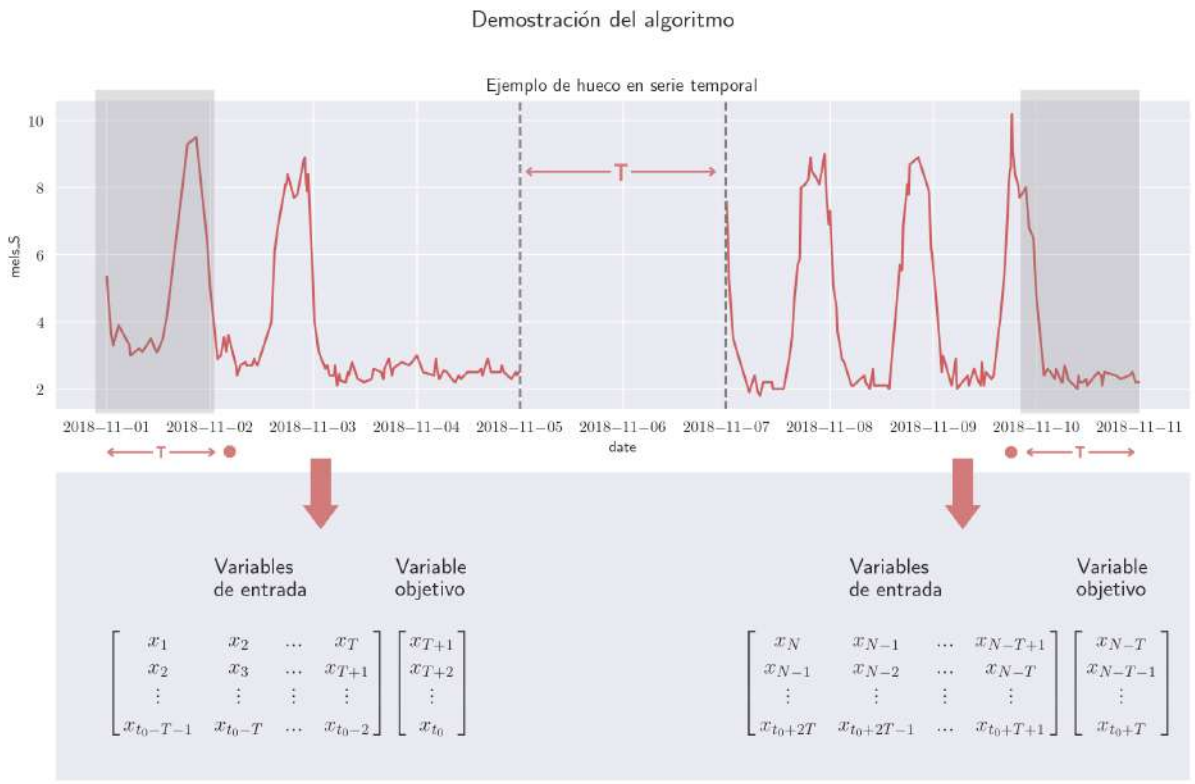

Figura 30: Proceso de obtención de serie multivariable a partir de una serie univariable

Tercera fase

En la tercera fase se entrenan dos modelos de Machine Learning, uno sobre cada conjunto obtenido. Tras realizar varios experimentos, se ha escogido el algoritmo de regresión mediante vectores de soporte (SVR), principalmente por su rapidez frente a otros como por ejemplo *random forest*.

Cuarta fase

Esta es la fase en la que se realizan las predicciones. Para esta fase seguiremos el proceso descrito en la Figura 31. En ella podemos comprobar como la predicción obtenida para cada uno de los instantes se utiliza a la hora de realizar la predicción del instante siguiente. Además, también se puede comprobar que en el conjunto de datos correspondientes a D_{a_t} se realizan las predicciones en el sentido contrario. En nuestro caso, al haber invertido los datos en la segunda fase, las predicciones se realizarían en el mismo sentido.

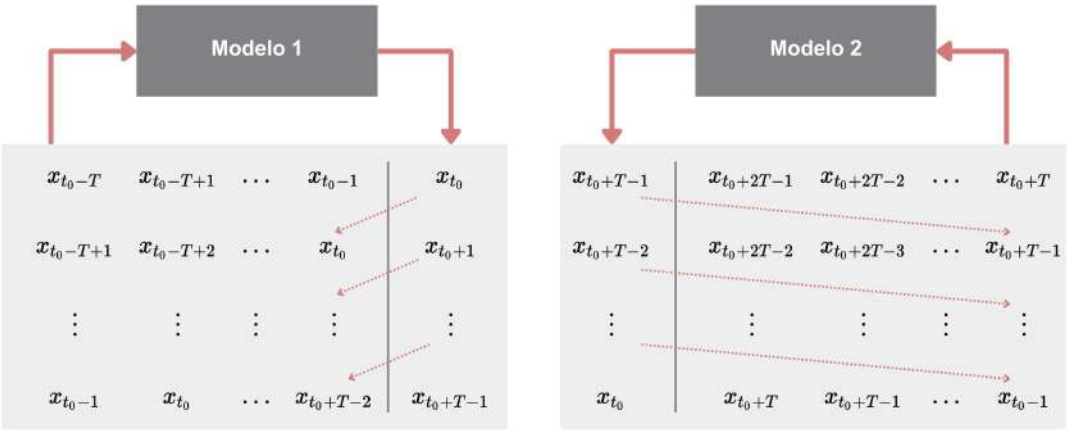

Figura 31: Proceso de obtención de predicciones para imputación de datos

Quinta fase

La quinta fase incluye la imputación en sí. Para obtener un solo dato para cada instante t en el hueco, se calcula la media de los datos predichos por cada uno de los modelos. Se ha escogido la media como operación porque es la que garantiza que los resultados sean estables y no sesgados [25].

6.2.2 *Descripción del algoritmo final*

Tras presentar el fundamento del algoritmo desarrollado, queda pendiente discutir ciertos aspectos que impiden que sea escalable. A continuación se discutirán las cuestiones más problemáticas, exponiendo la solución correspondiente.

Asumimos que tenemos una serie temporal X_t con $m \in \mathbb{N}$ huecos $(H_1, ..., H_m)$. La descripción a alto nivel del algoritmo básico está en el algoritmo 1.

Algorithm 1 Algoritmo de imputación básico

1: **Para cada** M_i, $i \in \{1, ..., m\}$ **hacer**
2: Establecer los valores de t_0, T y N, que corresponden al inicio y longitud del hueco, y al final de los datos posteriores, respectivamente.
3: Obtener D_{b_t} y D_{a_t}.
4: Transformar D_{b_t} y D_{a_t} para convertirlos en conjuntos de datos T-dimensionales.
5: Entrenar un modelo SVR sobre ellos.
6: Realizar T predicciones en cada modelo.
7: Calcular la media de los datos e imputarlos en el lugar correspondiente.
8:

A continuación exponemos uno a uno los problemas que surgen de aplicar el algoritmo anterior directamente sobre los datos en bruto de cualquier serie temporal.

Longitud mínima de D_{b_t} y D_{a_t}

En el apartado anterior se impuso la restricción de que D_{b_t} y D_{a_t} siempre debían tener longitud mayor que T. Esta condición es necesaria para aplicar el algoritmo anterior debido a que de no ser así, no se pueden obtener conjuntos de datos T dimensionales, y por tanto, sería imposible predecir T datos, tal y como necesitamos.

Sin embargo, sería incoherente asumir que siempre se va a cumplir pues existen varias situaciones lógicas en las que no se verifica. Por ejemplo, si el inicio del primer hueco se encuentra en cualquier índice $t < T$, o en el caso de que el final del último hueco esté en una posición p donde $L - p < T$, si L es la longitud total de la serie.

También podemos considerar otro caso más complejo. Suponemos que tenemos un hueco M_1 con inicio t_1 y longitud T_1, es decir que el final del hueco se da en la posición $t_1 + T$. Ahora suponemos que existe un hueco a continuación M_2 con inicio t_2 y longitud T_2, donde $t_2 - (t_1 + T) < T$. Esto quiere decir que el espacio entre ambos huecos, el equivalente a D_{b_t} de M_1 y a D_{a_t} de M_2 es de longitud menor que T. Esta situación se ve ejemplificada en la Figura 32.

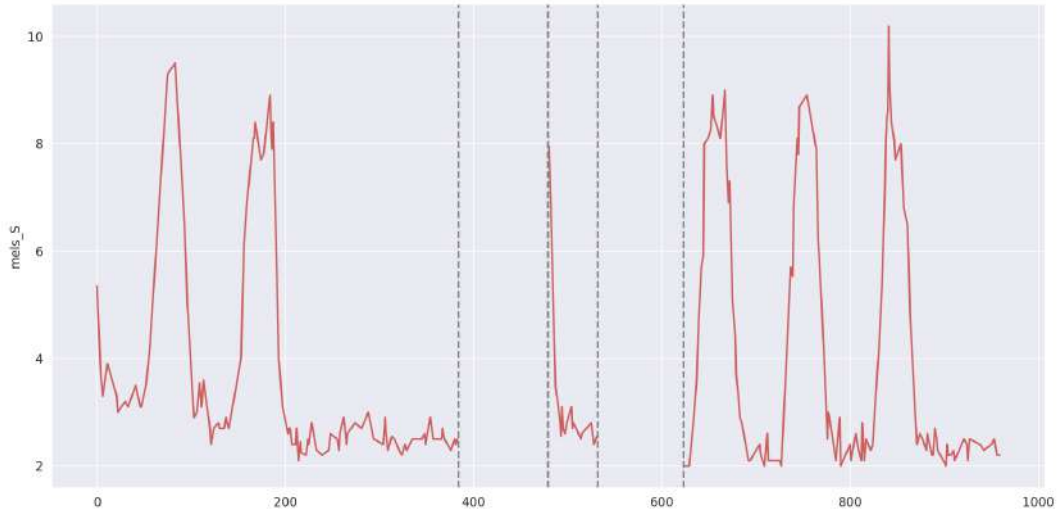

Figura 32: Ejemplo de huecos con poca distancia entre sí

Para idear una solución que se pudiera aplicar a cualquier tipo de serie temporal, hemos dividido el estudio en las dos situaciones mencionadas, pues cubren todas las posibilidades posibles, y se resuelven de manera distinta, tal y como vamos a ver a continuación.

Para la primera situación, que si recordamos incluye las situaciones donde el hueco se encuentra demasiado al principio o al final, hemos estudiado dos alternativas:

1. Utilizar el algoritmo descrito utilizando el conjunto D_{b_t} o D_{a_t}, donde haya datos suficientes (longitud mayor que T).
2. Utilizar algún otro modelo especializado en series temporales.

En el caso de utilizar un modelo especializado en series temporales se tuvo que decidir tanto qué modelo utilizar como cuántos datos utilizar para el entrenamiento, y si se quiere estudiar en más detalle, se puede consultar el capítulo correspondiente de la parte final de experimentos. Tras comparar la precisión de distintas combinaciones sobre huecos de longitud y distribución aleatoria generados artificialmente, se tomó la decisión de utilizar el algoritmo descrito, entrenado con los datos que haya disponibles.

Para la segunda situación, donde existen varios huecos y la distancia entre ellos es demasiado pequeña como para entrenar modelos de ML y realizar predicciones, se ha tomado la decisión de fusionar los huecos. Esto es, en el caso de la Figura 32, se procedería como si únicamente tuviéramos un hueco, H_1 (Figura 33). Después de realizar las predicciones, se imputarían únicamente los datos nulos.

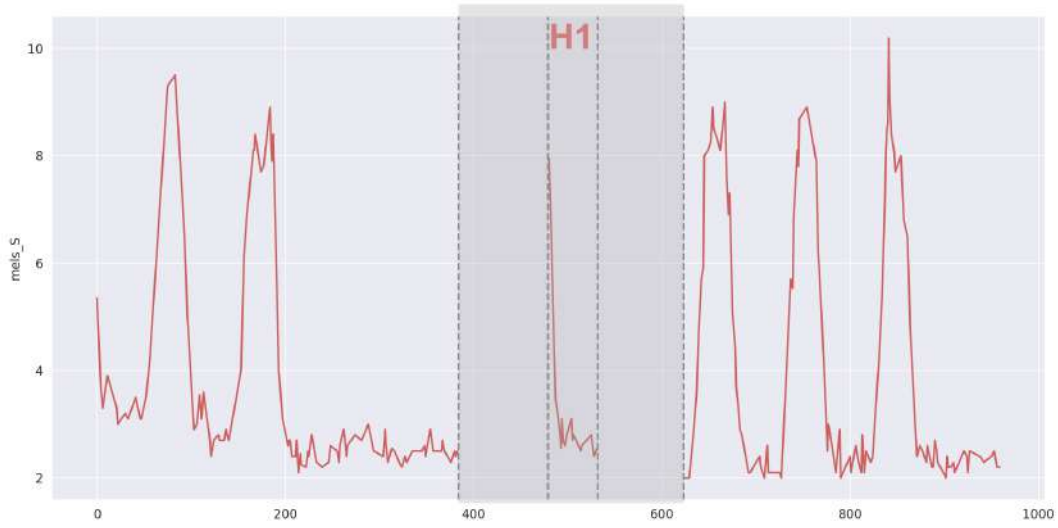

Figura 33: Ejemplo de fusión de huecos con poca distancia entre sí

Esta solución implica que la longitud del nuevo hueco es mucho mayor, al ser ser la suma de los dos huecos más la distancia entre ellos, y por tanto hace que sea más fácil que vuelva a aparecer este problema con futuros errores de datos.

Si esto ocurre y se terminan fusionando demasiados huecos, podría suponer un problema, pues al aplicar el algoritmo original estaríamos perdiendo toda la información que existe entre los huecos. Surge de nuevo la duda de si resultaría mejor utilizar un modelo como Prophet sobre la serie completa e imputar los datos nulos. Para responder a esa pregunta, se realizó un experimento para comparar la precisión de ambas imputaciones de datos sobre huecos cercanos (y por tanto susceptibles de ser fusionados) de longitud aleatoria generados artificialmente. Este experimento se puede consultar en detalle en la sección 9.1.3.

Tras estudiar los resultados, se llegó a la conclusión de que para fusiones de más de 3 huecos se realizaban imputaciones más cercanas a la realidad usando Prophet, y por tanto se escogió este criterio para el algoritmo final.

Longitud máxima de D_{b_t} y D_{a_t}

Además de la restricción ya mencionada sobre la longitud mínima de D_{b_t} y D_{a_t}, no se ha impuesto ninguna restricción más. Esto quiere decir que hasta la fecha, estamos cogiendo todos los datos disponibles para crear D_{b_t} y D_{a_t}. Como veremos a continuación, esto supone un problema a la hora de ejecutar el algoritmo a gran escala.

Si tuviéramos, por ejemplo, una serie con 10000 observaciones, con un único hueco de longitud 2 en una posición central, estaríamos usando miles de instantes previos y posteriores para imputar dos instantes. Esto no resulta lógico ni eficiente, pues el modelo SVR, aunque es relativamente rápido, no puede lidiar con esa dimensión de datos.

Se han establecido dos criterios que solucionan el problema

1. El primero, es separar los huecos por longitud, para evitar el coste de procesamiento que conlleva ejecutar el algoritmo para imputar huecos demasiado pequeños, en los que no se necesita tanta precisión. Se ha determinado que con huecos de longitud menor que una hora (el número de observaciones depende de la frecuencia de la serie tratada), se usará una **interpolación lineal**.
2. El segundo criterio establece la longitud máxima de D_{b_t} y D_{a_t}: **dos días de información por cada día de valores nulos**.

Para imponer el primer criterio nos hemos basado en el paper del dataset original, donde se establecía un criterio parecido.

Para el segundo, se ha estudiado la autocorrelación de varias de las series que tenemos y se ha llegado a la conclusión de que la correlación diaria tiene un peso importante. Esto quiere decir que los datos que se tomen un lunes a las 10:00 están correlados con los del martes a las 10:00.

Vamos a suponer que no se da ninguno de los problemas de datos insuficientes en D_{b_t} o D_{a_t}, es decir que la longitud de ambas es mayor que T. Consideramos D como el número de días presentes en T.

Se realizó un experimento en el que se generaron huecos de longitud aleatoria de manera artificial y por cada día de datos nulos, se comparó la precisión de SVR al tomar un día, dos días y una semana de información previa y posterior. De nuevo, si se quiere entrar en detalle sobre la ejecución y los resultados del experimento, se puede encontrar en la sección 9.1.2.

Después de analizar los resultados, se tomó la decisión ya comentada: tomar dos días de información previa y posterior por cada día de datos nulos.

Para ejemplificarlo, vamos a considerar la variable *hvac_N*, que tiene una frecuencia de 15 minutos. Suponemos que tenemos un hueco de longitud $T = 328$, es decir, $\frac{328}{4} = 82$ horas de hueco, y 3,41 días de datos nulos. Para poder considerar los casos en los que tenemos un número de días menor a 1, vamos a redondear hacia arriba y por tanto procederemos como si tuviéramos 4 días de datos nulos.

En el primer paso del algoritmo, donde se obtienen D_{b_t} y D_{a_t}, se tomarán $4 \cdot 2 = 8$ días de información, o lo que es equivalente, 768 observaciones. Sabemos que tenemos por lo menos $T = 328$ datos previos y posteriores, pero no podemos asegurar que haya 768, por esta razón tomaremos los datos disponibles, con una longitud máxima de 768.

A continuación vamos a definir el algoritmo final utilizado, separando tres fases:

1. Búsqueda de los huecos $M_i, i = 1, ..., m$
2. Interpolación lineal en huecos menores de una hora
3. Fusión de huecos mayores de una hora donde sea necesario, obteniendo $H_i, i = 1, ..., m_1$
4. Imputación secuencial a través de $H_i, i = 1, ..., m_1$.

Para implementar los dos primeros pasos, se iterará a través de toda la serie, almacenando en qué índice comienza y termina un hueco, y clasificándolo en función de si es menor o mayor de una hora. Seguidamente se procederá a realizar una interpolación lineal en los clasificados como *pequeños*.

La implementación del tercer paso es un poco más compleja, y se puede ver representada en el algoritmo 2.

Algorithm 2 Algoritmo de fusión de huecos

Sean $M_i, i = 1, ..., m$ huecos
$i = 0$
Mientras $i < m$
    Obtener inicio $inicio = t_i$, final $final = q_i$ y longitud $long = T_i$ del hueco actual.
    Obtener inicio del hueco siguiente M_{i+1}, $inicio_s = t_{i+1}$
    Obtener longitud siguiente $long_s = T_{i+1}$
    Mientras $inicio_s - final < long$
        Almacenar $inicio$, $final$
        Actualizar $final$=t_{i+1}
        $long = long + long_s$
        Actualizar $i = i + 1$
        Obtener inicio del hueco siguiente M_{i+1}, $inicio_s = t_{i+1}$
        Obtener longitud siguiente $long_s = T_{i+1}$
    Fin
Fin

Tal y como se puede observar en el algoritmo 2, antes de fusionar dos huecos, se almacena el inicio y el final del hueco actual. Esto es para que a la hora de realizar predicciones, se puedan imputar los valores predichos únicamente en las posiciones donde había valores nulos.

Por último, estudiamos la implementación a alto nivel del cuarto paso, representada en el algoritmo 3.

Algorithm 3 Algoritmo de imputación final

Sean H_i, $i \in \{1, ..., m_1\}$ los huecos fusionados
Sea N_i el número de huecos que conforman $H_i, i \in \{1, ..., m_1\}$
Sea k el número de observaciones cada hora
Sea f el índice final de la serie
Para cada H_i, $i \in \{1, ..., m_1\}$ **hacer**
    Obtener los valores de t_0, T
    if $N_i > 3$ **then**
        Imputar datos utilizando Prophet
    else
        Obtener número de días del hueco actual D
        if $t_0 < T$ **o** $f - (t_0 + T) < T$ **then**
            Obtener D_{a_t} con datos no nulos en $[t_0 + T, t_0 + T + 2 \cdot D \cdot 24 \cdot k]$
            Transformar D_{a_t}.
            Entrenar un modelo SVR sobre D_{a_t}.
            Realizar T predicciones e imputar los datos .
        else
            Obtener D_{a_t} con datos no nulos en $[t_0 + T, t_0 + T + 2 \cdot D \cdot 24 \cdot k]$
            Obtener D_{b_t} con datos no nulos en $[t_0 - 2 \cdot D \cdot 24 \cdot k, t_0]$
            Transformar D_{b_t} y D_{a_t}.
            Entrenar un modelo SVR sobre ellos.
            Realizar T predicciones en cada modelo.
            Calcular la media de los datos e imputarlos en el lugar correspondiente.
        end if
    end if

En esta sección hemos presentado en detalle el algoritmo desarrollado para abordar el problema de datos nulos presente en la colección. Es importante destacar que, aunque el algoritmo ha sido desarrollado específicamente para nuestros datos particulares, se ha tenido en cuenta desde el principio la necesidad de hacerlo lo más escalable posible.

Por lo tanto, se espera que esta metodología pueda ser aplicada sobre otro tipo de series temporales. Se recomienda a los interesados que consulten el código completo correspondiente, que se encuentra disponible en el repositorio del proyecto[1].

1 https://github.com/maaguado/time-series-building59

7

ANÁLISIS DE SERIES TEMPORALES

En este capítulo se analizarán las propiedades principales de la serie temporal que actuará como variable objetivo. Asimismo, se estudiará la correlación entre ésta y el resto de variables, con el fin de evaluar su influencia en las predicciones y determinar cuáles son las más relevantes de cara a predecir el consumo de energía.

El análisis de las propiedades de las series temporales es fundamental para poder obtener un modelo que se ajuste de manera precisa a los datos, teniendo en cuenta que diferentes modelos son más eficientes para series con distintas propiedades. En este sentido, aspectos como analizar la estacionariedad de la serie u obtener una descomposición correcta son cruciales pues permiten entender el comportamiento de la serie y su evolución en el tiempo.

7.1 ESTUDIO DE LA VARIABLE OBJETIVO

Vamos a comenzar estudiando las propiedades individuales de la variable objetivo, que será el consumo total de aire acondicionado en el edificio, formado por la suma del consumo en el ala sur (*hvac_S*) y el ala norte (*hvac_N*). En primer lugar, se estudiará la estacionariedad de la serie, que ya hemos introducido formalmente como una propiedad fundamental en las series. Asimismo, se analizarán las distintas componentes presentes en la serie, tales como la tendencia o la estacionalidad.

7.1.1 *Estacionariedad*

Para estudiar la estacionariedad, podemos proceder de distintas maneras. Sabemos que si una serie temporal es estacionaria, entonces tiene media y varianza constante. Por eso, la primera forma de estudiar la estacionariedad de una serie es calcular la media y la varianza a través del tiempo y comprobar si son constantes.

En la Figura 34 podemos estudiar la variación de la media y la desviación típica calculadas con una ventana de un día, o lo que es equivalente, 96 observaciones. Se puede observar que ninguno de los dos valores es constante, lo que nos lleva a intuir que la serie temporal no es estacionaria.

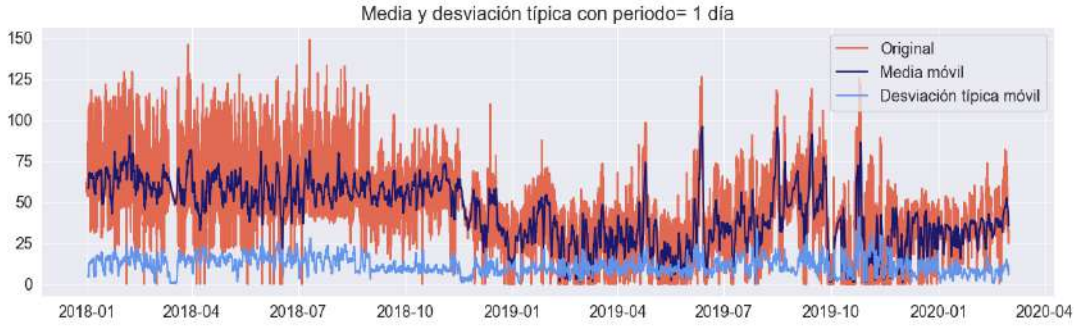

Figura 34: Media móvil y desviación típica

Se ha omitido la varianza en el gráfico de cara a mejorar su legibilidad, pero la conclusión a la que llegamos estudiando la varianza es la misma.

Al repetir este estudio con la serie temporal diferenciada (Figura 35) concluimos que el operador de diferenciación convierte la serie en estacionaria.

Figura 35: Media móvil y desviación típica con la serie diferenciada

La segunda forma de comprobar si se cumple esta propiedad es mediante un test estadístico introducido en capítulos anteriores: Test Aumentado de Dickey-Fuller. Al ejecutar el test sobre la serie obtenemos un p-valor de $5 \cdot 10^{-20}$, considerablemente menor que 0,05. Esto nos lleva a rechazar la hipótesis nula H_0, que si recordamos era *existe una raíz unitaria*.

Este resultado es destacable pues de haber realizado únicamente el test estadístico hubiéramos deducido que la serie temporal es estacionaria al no tener raíces unitarias. Esta serie es por tanto un ejemplo de un proceso estocástico no estacionario sin raíz unitaria.

7.1.2 *Descomposición de la serie temporal*

La descomposición de una serie temporal es un paso fundamental para realizar predicciones precisas. Cada componente se puede modelar y predecir por separado, y luego se pueden combinar para obtener la predicción final de la serie completa. Tal y

como estudiamos en el capítulo correspondiente a los fundamentos matemáticos, las dos descomposiciones usuales para una serie temporal son la aditiva y la multiplicativa.

7.1.2.1 *Estacionalidad*

Dado el contexto, tiene sentido considerar patrones estacionales que se repiten regularmente en un periodo de tiempo. Por lo tanto, es necesario estudiar las distintas componentes estacionales presentes en la serie antes de obtener una descomposición adecuada para la misma.

En este caso particular, tenemos que tener presente que pueden existir varias componentes estacionales. En primer lugar, es probable que se observe una componente estacional diaria, ya que el consumo de aire acondicionado suele aumentar en las horas de mayor calor del día y disminuir durante las horas de la noche. Además, es posible que se observe una estacionalidad semanal, dado que los patrones de consumo pueden variar entre los días de semana y los fines de semana. Asimismo, pueden existir efectos estacionales asociados a cambios en la temperatura y la humedad, lo que puede generar un patrón estacional a lo largo de las distintas estaciones del año.

Existen distintas técnicas para buscar componentes estacionales de manera empírica y no solo basándonos en la intuición. La primera es utilizar un diagrama de caja para estudiar la distribución del consumo por hora, día de la semana o por mes.

En la Figura 36 se pueden ver los diagramas correspondientes. Tras analizarlo, podemos determinar que existe una variación diaria y mensual. En el caso de la componente semanal, las cajas asociadas a cada día de la semana son casi idénticas. Sin embargo, no deberíamos descartar este componente pues la distribución en los bigotes de cada caja es muy distinta.

La segunda técnica para analizar las componentes estacionales es estudiar la autocorrelación y la autocorrelación parcial de la serie temporal dado un periodo k.

En capítulos anteriores se definió la función de autocovarianza para un proceso estocástico $\{Y_t, t \in T\}$ como

$$\gamma_Y(r,s) = Cov(Y_r, Y_s) = \mathbb{E}[(Y_r - \mathbb{E}Y_r)(Y_s - \mathbb{E}Y_s)], \qquad r,s \in T \tag{7.1.1}$$

Si consideramos ahora un periodo fijo pero arbitrario k, y tomamos la serie temporal como una muestra aleatoria $X_1, ..., X_n$ podemos definir esta función como:

$$\gamma_X(k) = \frac{1}{n}\sum_{i=0}^{n-k}[(X_i - \mathbb{E}X)(X_{i+k} - \mathbb{E}X)], \qquad r,s \in T \tag{7.1.2}$$

Teniendo lo anterior en cuenta, podemos definir la función de autocorrelación (ACF) como

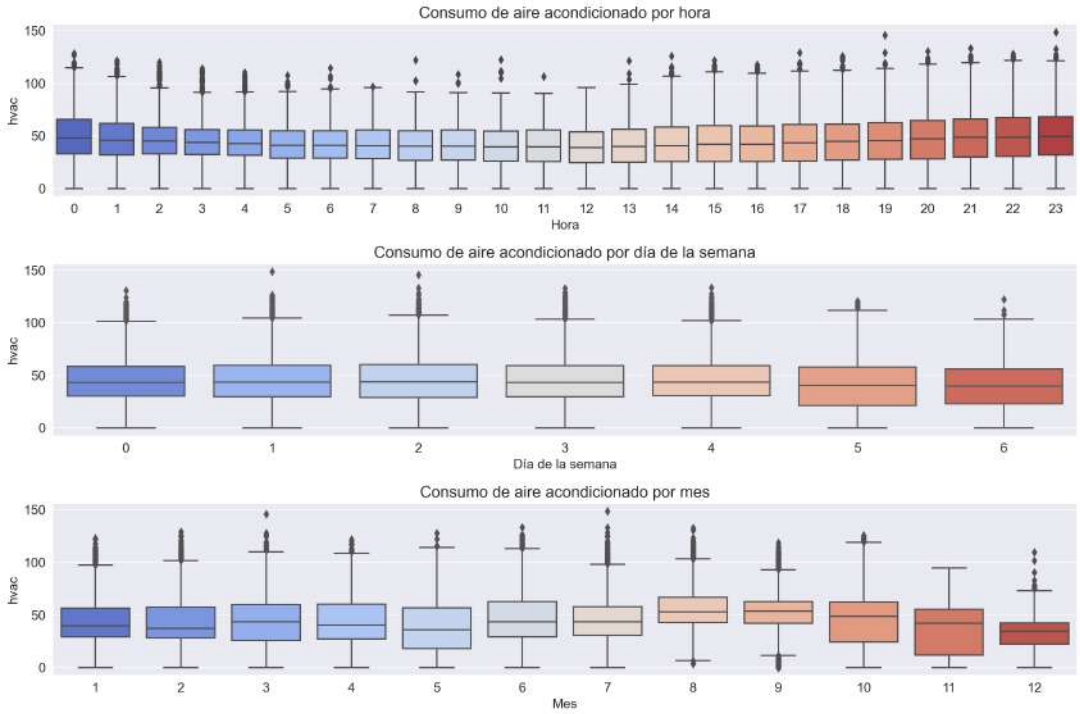

Figura 36: Análisis de componentes estacionales utilizando diagramas de caja

$$r_X(k) = \frac{\gamma_X(k)}{\gamma_X(0)} \qquad k \geq 0 \tag{7.1.3}$$

donde $\gamma_X(0)$ es la varianza de la muestra aleatoria.

De una manera menos formal, esta función mide la correlación entre una observación de la serie en un tiempo determinado y las observaciones en tiempos anteriores. Es decir, muestra cómo de correlacionados están los valores de la serie con los valores de la misma serie en diferentes intervalos de tiempo. Los valores de la ACF varían entre -1 y 1, donde -1 indica una correlación negativa perfecta, 0 indica ausencia de correlación y 1 indica una correlación positiva perfecta.

Esta función es esencial para analizar las componentes estacionales, por ejemplo, si la serie presenta una estacionalidad diaria, al estudiar la gráfica de la función ACF se observará una correlación alta en los intervalos de tiempo correspondientes a un día, es decir, la ACF mostrará picos altos en esos intervalos.

La función de autocorrelación parcial (PACF) de una serie es la que representa autocorrelación entre X_t y X_{t+k} donde se ha eliminado la dependencia lineal de X_t sobre $X_{t+i} \qquad i = 1, ..., k-1$, se denotará $rp_X(k)$. A diferencia de la ACF, que mide la correlación directa entre un valor en un momento de tiempo y los valores en momentos anteriores, la PACF mide la correlación directa entre un valor en un momento de tiempo y los valores en momentos anteriores, pero eliminando la contribución de los valores intermedios.

Sin entrar en demasiado detalle vamos a definirla como:

$$rp_X(k) = \begin{cases} corr(X_{t+1}, X_t) & k = 1 \\ corr(X_{t+k} - \hat{X_{t+k}}, X_t - \hat{X_t}) & k \geq 2 \end{cases}$$

donde $\hat{X_{t+k}}$ y $\hat{X_t}$ representan las combinaciones lineales de $\{X_{t+1}, ..., X_{t+k-1}\}$ que minimizan el error cuadrático medio de X_{t+k} y X_t, respectivamente.

Esta función también resulta útil para identificar patrones estacionales complejos en la serie, pues permite detectar la correlación directa entre los valores en momentos separados por varios intervalos de tiempo.

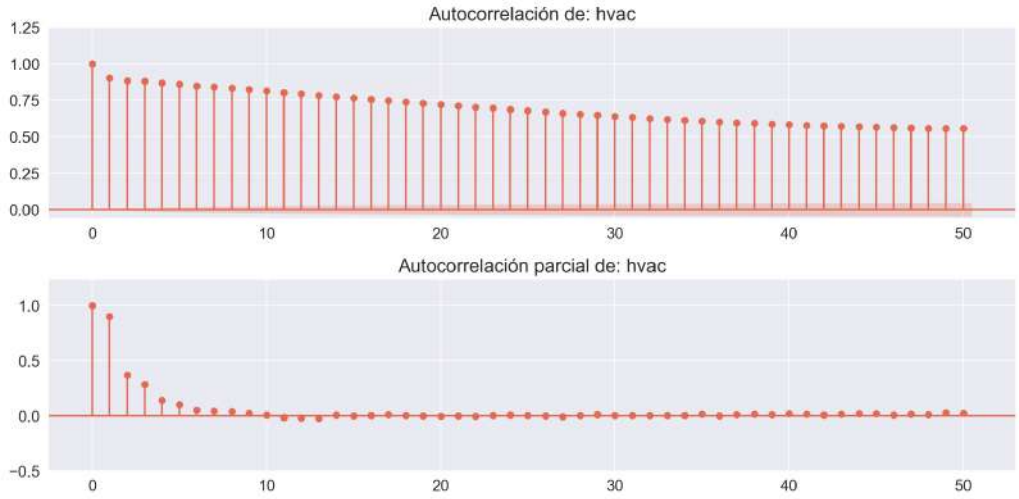

Figura 37: ACF y PACF del consumo de aire acondicionado

La Figura 37 contiene las gráficas correspondientes a las funciones ACF y PACF. De la primera no podemos obtener ninguna conclusión sobre las componentes estacionales, pues existe una autocorrelación demasiado alta en intervalos de tiempo muy altos. Si analizamos la Figura 38 se puede observar un patrón cada 100 intervalos aproximadamente, que coincide con una estacionalidad diaria (96 intervalos), y otro patrón cada 672 intervalos, que corresponde a la estacionalidad semanal.

Tras analizar la gráfica PACF, podemos inferir que el efecto de las observaciones intermedias es considerable, pues al eliminarlo la autocorrelación baja considerablemente de manera muy rápida.

7.1.2.2 *Descomposiciones*

Teniendo en cuenta que existen varias componentes estacionales en la serie, no tiene sentido utilizar una descomposición aditiva o multiplicativa, pues éstas únicamente consideran una componente estacional. Si eligiéramos este tipo de descomposiciones tendríamos información esencial de la serie en el residuo, lo cual no resulta adecuado, pues se perdería esa información al modelar cada componente.

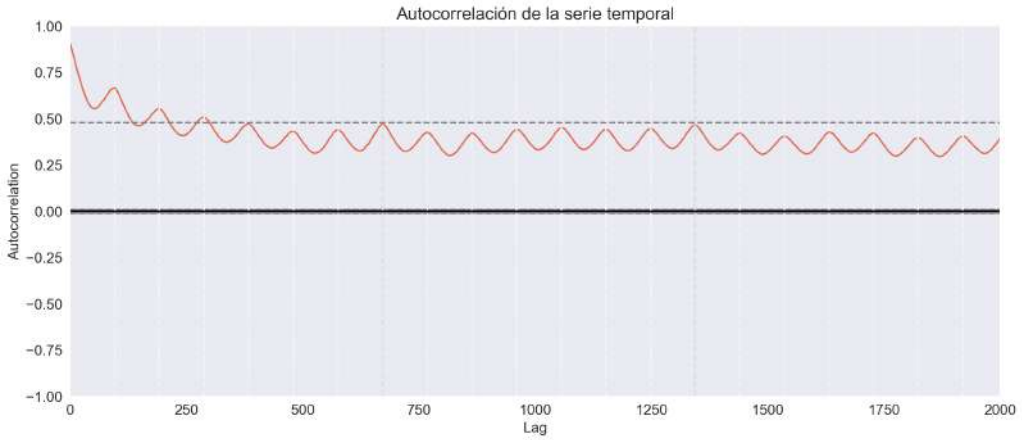

Figura 38: Autocorrelación de la serie con intervalos altos

Por tanto, es necesario estudiar descomposiciones que consideren varias componentes estacionales. Aunque los resultados detallados se encuentran más adelante en la memoria, en la sección 10.2, vamos a estudiar brevemente qué descomposiciones se han considerado.

Descomposición aditiva con múltiples componentes estacionales

Para llevar a cabo esta descomposición se utiliza iterativamente la descomposición aditiva simple. Para obtener cada componente en la descomposición aditiva simple, primero se extrae la tendencia mediante un filtro convolucional. La componente estacional se calcula entonces como la media de los datos sin tendencia para el periodo determinado.

Para poder incluir distintas componentes estacionales procedemos extrayéndolas una a una. Es decir, si consideramos tres componentes (horario, diario y semanal), primero se obtendría la componente horaria, después la diaria y en el último paso, al extraer la semanal, es cuando obtenemos la tendencia y los residuos finales.

Formalmente, esta descomposición se puede escribir como:

$$Y_t = T_t + SH_t + SD_t + SS_t + R_t \tag{7.1.4}$$

Decomposición MSTL

La descomposición MSTL (Descomposición estacional-tendencia múltiple utilizando Loess), se puede expresar como una descomposición aditiva con múltiples componentes estacionales:

$$Y_t = T_t + S_t^1 + ... + S_t^N + R_t \tag{7.1.5}$$

Este método utiliza STL como base, que modela una única componente estacional.

Antes de introducir a alto nivel en qué consiste el método STL, es necesario notar que éste emplea Loess para obtener cada componente.

Loess se utiliza para obtener curvas suavizadas ajustadas a un gráfico de dispersión. Para calcular la curva, se ajusta un polinomio a un conjunto de datos alrededor de un punto específico, lo que requiere definir tanto el grado del polinomio como el tamaño de la ventana utilizada para ajustar la curva. La curva suavizada se utiliza para modelar la tendencia. En el caso de STL, se utiliza el método Loess para obtener una única componente estacional a partir de diversas transformaciones de la serie original. Aunque no se profundizará en estos métodos en este trabajo, es importante mencionarlos.

A continuación vamos a estudiar el algoritmo MSTL paso a paso. En el primer paso, se extrae cada componente estacional de manera individual, comenzando por la componente con un periodo más corto y restando la componente anteriormente extraída antes de continuar con la siguiente. Esto se hace para evitar incluir de forma errónea una estacionalidad pequeña dentro de una componente con un periodo más largo.

El segundo paso trata de refinar las componentes obtenidas, que al fin y al cabo no son más que una estimación. Se vuelve a iterar a través de las componentes, y en cada iteración se añade la componente obtenida en el primer paso a la serie sin componentes estacionales (en el paso anterior se han restado todos). Después, se vuelve a extraer la misma componente utilizando STL. Este paso es fundamental pues extrae la componente de una serie temporal que únicamente presenta esa componente estacional. Esto hace que STL pueda capturar información que no se había conseguido captar en el primer paso.

Este último paso se puede repetir tantas veces como sea necesario. Sin embargo, es importante fijar el número de repeticiones antes de comenzar el proceso.

7.1.2.3 *Análisis de descomposiciones*

Para poder comprobar la bondad de una descomposición, nos centraremos en los residuos obtenidos tras descomponer la serie. Los residuos representan la información no explicada por las componentes consideradas, por lo que nos interesa que dicha información sea lo más parecida posible a ruido blanco, es decir, que no presente correlación. Esto significa que no haya patrones o dependencias temporales significativas en ellos, lo que implica que no queda ninguna información importante sin capturar en la serie.

En general, el análisis de residuos que llevaremos a cabo constará de:

- Las gráficas ACF y PACF de la componente residual
- El gráfico Q-Q, que veremos a continuación
- El histograma de los residuos representado junto con una estimación de la densidad de probabilidad de los datos, también estudiado en detalle más adelante
- Test estadístico de Ljung-Box

En la sección anterior se han presentado en detalle las gráficas ACF y PACF, por lo que comenzaremos directamente por el gráfico Q-Q.

El gráfico Q-Q, también conocido como gráfico cuantil-cuantil, es una herramienta gráfica utilizada para comparar la distribución de una muestra de datos con una distribución teórica o de referencia, como la distribución normal. Los cuantiles de la muestra de datos se representan en el eje vertical, mientras que los cuantiles esperados de la distribución teórica se representan en el eje horizontal. Si los datos siguen la distribución teórica, los puntos en el gráfico se alinearán aproximadamente a lo largo de una línea diagonal.

En nuestro caso, compararemos los cuantiles de los residuos con la distribución normal, pues ya hemos visto que un proceso de ruido blanco está normalmente distribuido, y por tanto, resulta interesante que los residuos se acerquen lo máximo posible a dicha distribución.

Otra herramienta que utilizaremos será el histograma, que es una representación gráfica de la distribución de datos numéricos en forma de barras. El eje horizontal del histograma representa las diferentes categorías o rangos de valores, y el eje vertical muestra la frecuencia con la que se observa cada categoría. En este gráfico se incorporará también la línea KDE (*Kernel Density Estimation*, en inglés), que representa una estimación suave y continua de la densidad de la distribución de los datos.

La línea KDE se obtiene al calcular una función de densidad utilizando un método estadístico llamado **estimación de densidad de núcleo**. Matemáticamente, sería como sigue:

$$\widehat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h\left(x - x_i\right) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{7.1.6}$$

donde K es una función no negativa denominada núcleo, y $h > 0$ es un parámetro de suavizado.

El objetivo de analizar la línea KDE junto al histograma es obtener una visualización más suave de la distribución, lo que puede ayudar a detectar patrones, y características más sutiles que pueden no ser fácilmente visibles en el histograma.

Por último, el test estadístico de Ljung-Box fue introducido en secciones anteriores, y será un indicador de autocorrelación presente en los datos.

Los resultados de los experimentos llevados a cabo se encuentran en el Capítulo 10, y tras experimentar con distintas descomposiciones no hemos encontrado una que resulte en residuos no correlados. Por esta razón, se ha tomado la decisión de realizar predicciones sobre la serie temporal sin descomponer.

7.2 ANÁLISIS MULTIVARIABLE

Es importante llevar a cabo un análisis multivariable para poder comprender cómo se relacionan las variables presentadas en capítulos anteriores. Esto es porque a menudo, las variables están interconectadas y tienen un impacto mutuo que en algunas ocasiones puede llegar a impedir realizar predicciones precisas. Para realizar este análisis, en primer lugar, se deben identificar todas las variables relevantes para la variable objetivo teniendo en cuenta el contexto, y se tiene que asegurar que los datos de estas variables estén disponibles para el período de tiempo que se va a analizar.

En nuestro caso, el inicio de la toma de datos de las variables correspondientes a la temperatura de las zonas termales interiores es el 22 de febrero del 2018. Esta fecha es posterior al inicio del resto de variables, por lo que simplemente la tomaremos como inicio del conjunto de datos. Además, si recordamos la exposición inicial de las variables, tenemos datos de temperatura del aire tomados por dos sensores distintos. Resulta incoherente mantener las dos variables pues son prácticamente idénticas, por lo que tomaremos la media entre las dos como una nueva variable, y descartaremos las anteriores. Por el momento mantendremos el resto de las variables, asumiendo que todas son relevantes para la variable objetivo y más adelante comprobaremos si esta suposición se cumple.

Por otra parte, se debe estudiar si las variables están correlacionadas entre sí. En el caso de que lo estén, la correlación de las variables dificulta la extracción del efecto individual de cada una de ellas, y diremos que estamos ante un problema de **multicolinealidad**. La presencia de este problema puede hacer que sea difícil interpretar los resultados del modelo y hacer predicciones precisas. La multicolinealidad también puede provocar que los coeficientes de los parámetros en el modelo sean inestables y varíen significativamente dependiendo de los datos de entrada utilizados para ajustar el modelo, lo que se traduce como una falta de robustez del modelo.

Para analizar la correlación entre las variables utilizaremos el coeficiente de correlación de Pearson, que mide la correlación lineal entre dos variables y que se calcula como sigue:

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{7.2.1}$$

donde σ_X y σ_Y representan las desviaciones típicas de X e Y, respectivamente.

El problema principal de esta medida es que solo permite estudiar la correlación entre dos variables, y en nuestro caso, al tener un número muy grande de ellas, resulta complicado analizarlo a simple vista. Para solucionarlo, utilizamos la matriz de correlación, que incluye el coeficiente de correlación de Pearson para cada par de variables, y que permite ser visualizada de una manera sencilla.

En la Figura 39 observamos la matriz de correlación de las variables correspondientes a las zonas termales exteriores, ordenadas por bloques (tabla 2). Podemos comprobar

que dentro del primer y último bloque se da una correlación positiva relativamente elevada y de forma independiente a los demás bloques. En el segundo y el tercero sin embargo, vemos una correlación muy alta general, que no se da con las variables del primero y del cuarto.

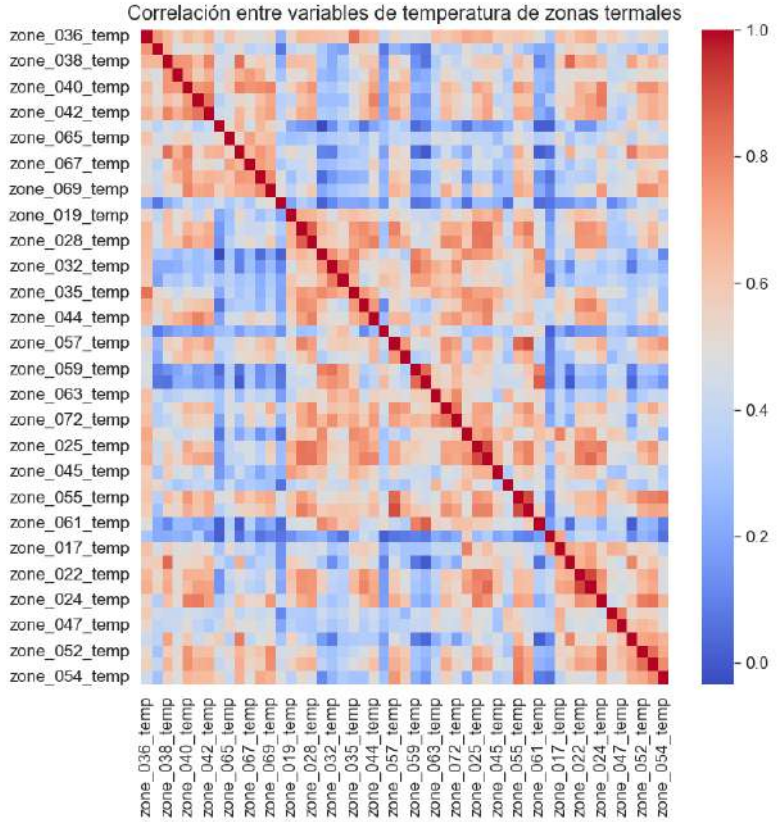

Figura 39: Correlación de las variables de temperatura en zonas termales exteriores

Teniendo en cuenta la alta correlación intra-bloque y que el número de variables que conforman este conjunto es muy elevado, resulta natural buscar una manera de reducir la dimensionalidad sin perder demasiada información. Para ello usamos una técnica denominada **Análisis de Componentes Principales** (PCA, por sus siglas en inglés).

7.2.1 *Análisis de Componentes Principales*

Esta técnica se basa en la necesidad de obtener una representación con una dimensión baja que capture toda la información posible. En este caso, teniendo en cuenta la división de las zonas termales, buscamos una representación con tres dimensiones, que en otras palabras, supone buscar tres variables nuevas que incluyan la información relevante.

Suponemos que tenemos N variables y p observaciones de cada una de ellas. La idea es que cada una de las observaciones se encuentra en un espacio N-dimensional, pero

no todas las dimensiones son igual de relevantes, donde el concepto de relevancia se mide por la variación de cada una de las observaciones a través de las N variables. Cada una de las variables (o dimensiones) obtenidas a través de PCA se denominan componentes principales y son una combinación lineal de las N variables originales.

La primera componente principal Z_1 es la combinación lineal

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \ldots + \phi_{N1} X_N \tag{7.2.2}$$

que tenga la mayor varianza, donde $\sum_{j=1}^{N} \phi_{j1}^2 = 1$. Esta condición es necesaria pues no restringir los valores absolutos podría resultar en una varianza arbitrariamente grande.

Como únicamente estamos interesados en la varianza, vamos a asumir que cada una de las variables se ha centrado para tener media cero. Y entonces, basta con buscar la combinación lineal de las observaciones que sean de la forma:

$$z_{i1} = \phi_{11} x_{i1} + \phi_{21} x_{i2} + \ldots + \phi_{N1} x_{iN} \tag{7.2.3}$$

con la mayor varianza, sujeta a la condición mencionada con anterioridad.

Esto no es más que un problema de optimización, formulado como sigue

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximizar}} \left\{ \frac{1}{p} \sum_{i=1}^{p} \left(\sum_{j=1}^{N} \phi_{j1} x_{ij} \right)^2 \right\} \text{ sujeto a } \sum_{j=1}^{N} \phi_{j1}^2 = 1 \tag{7.2.4}$$

Después de obtener la primera componente principal, podemos pasar a extraer la segunda. La segunda componente principal será la combinación lineal

$$Z_2 = \phi_{12} X_1 + \phi_{22} X_2 + \ldots + \phi_{N2} X_N \tag{7.2.5}$$

Siguiendo este proceso podemos obtener cada una de las componentes hasta llegar a la dimensión requerida. Una vez se han obtenido todas, podemos analizar el porcentaje de la varianza total explicado por cada una de las componentes principales, para determinar cuánta información se ha perdido con respecto a las variables originales.

Consideramos la varianza del conjunto original como la suma de las varianzas de cada una de las variables:

$$\sum_{j=1}^{N} \text{Var}\left(X_j\right) = \sum_{j=1}^{N} \frac{1}{p} \sum_{i=1}^{p} x_{ij}^2 \tag{7.2.6}$$

Y la varianza explicada por la i-ésima componente principal viene dada por:

$$\frac{1}{p}\sum_{i=1}^{p} z_{im}^2 = \frac{1}{p}\sum_{i=1}^{p}\left(\sum_{j=1}^{N}\phi_{jm}x_{ij}\right)^2 \tag{7.2.7}$$

Por lo que el porcentaje de varianza explicada por cada una de las componentes no es más que el resultado de dividir la varianza de la componente i-ésima entre la varianza total, y multiplicarlo por 100.

Es necesario notar que el algoritmo de PCA está afectado por la magnitud de cada variable, por lo que será necesario estandarizar las variables mediante la operación siguiente:

$$Z = \frac{X - \mu}{s} \tag{7.2.8}$$

donde μ es la media y s la desviación típica de X.

Volviendo a nuestro caso particular, obtenemos tres componentes, cuyos porcentajes de varianza explicada son 0.51, 0.11 y 0.06, respectivamente. Es destacable la bajada repentina en la varianza explicada, pero resulta natural con el proceso a seguir para obtener las componentes.

También podemos aplicar esta técnica al conjunto de variables correspondiente a la temperatura de las zonas termales interiores, pues suponen un conjunto con una dimensionalidad muy alto y variables altamente correladas. Tras realizar el proceso, esta vez con cuatro componentes principales, obtenemos como porcentajes de varianza explicada 0.45, 0.11, 0.10 y 0.07.

Finalmente, podemos estudiar la correlación entre todas las variables, representada en la Figura 40. En el gráfico de la izquierda se ve la matriz de correlación completa, y en el de la derecha se ha filtrado para considerar únicamente los valores mayores que 0,6 en valor absoluto. Podemos observar que existe una correlación elevada entre las variables de uso de energía en el ala sur, entre los dos conjuntos de variables que hemos obtenido mediante PCA en el apartado anterior y entre las variables de humedad relativa y temperatura de rocío.

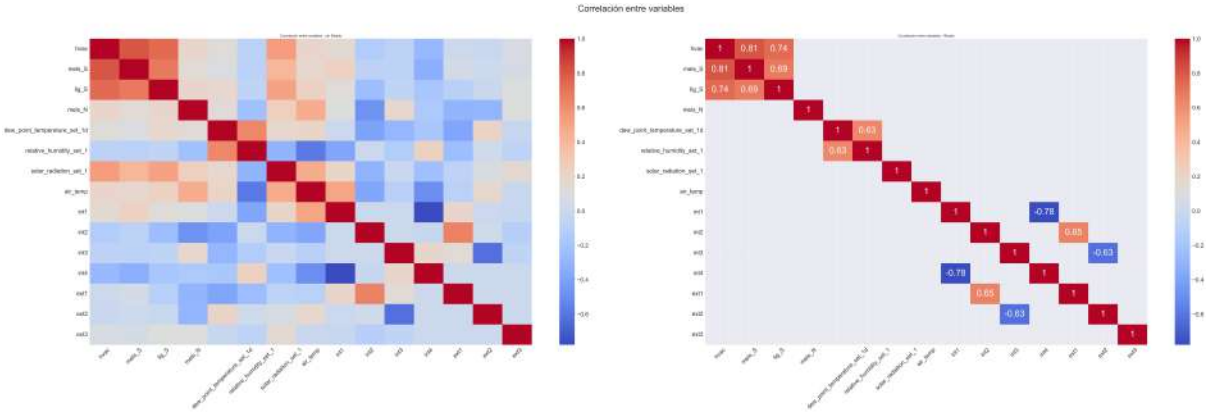

Figura 40: Correlación de las variables consideradas para el estudio

Para evitar la multicolinearidad se han llevado a cabo tres soluciones diferentes. La primera es tomar como variable el consumo de energía total en el ala sur, es decir, considerar la suma de consumo misceláneo y consumo en electricidad, y descartar éstas últimas. En segundo lugar, vamos a descartar la variable de temperatura de rocío, pues aunque pueda aportar información, se considera que solucionar el problema de correlación es más importante. Por último, se ha repetido el proceso de obtención de PCA sobre todas las variables de temperatura en zonas (interior y exterior), esta vez obteniendo seis componentes principales.

Al repetir el análisis de la matriz de correlación tras realizar estos cambios (Figura 41) comprobamos que la única correlación presente se da entre las variables de consumo energético en el ala norte y sur, lo que resulta natural. Por el momento, este será el conjunto de variables consideradas para realizar predicciones.

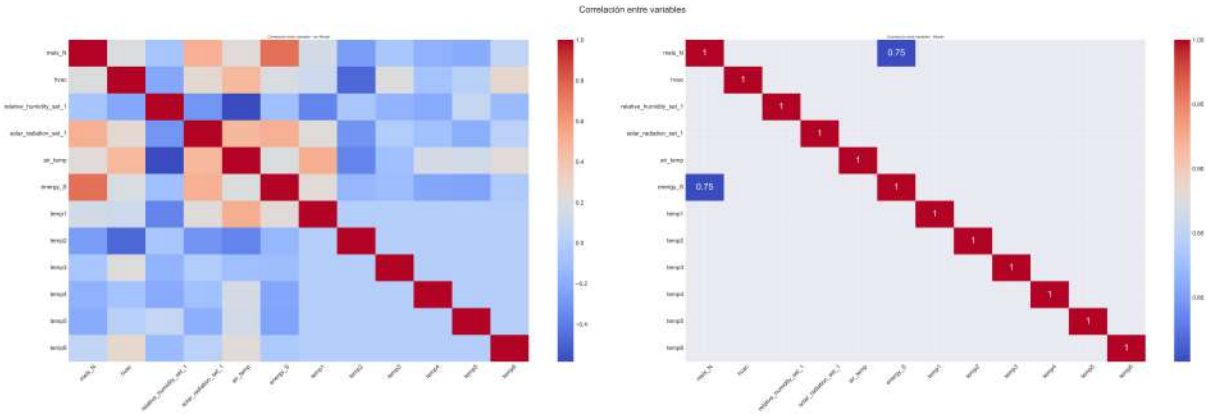

Figura 41: Correlación de las variables consideradas para el estudio tras modificaciones

En este capítulo, se han analizado las propiedades de la serie temporal tanto en su forma univariable como multivariable. Se ha llevado a cabo una exploración exhaustiva de los datos para entender su comportamiento y detectar patrones y tendencias en la serie. Asimismo, se han identificado y tratado diferentes problemas como la presencia de valores atípicos o la multicolinealidad entre variables. Con todo ello, se dispone de una base sólida de conocimiento para abordar el siguiente capítulo, que se centrará en la obtención de un modelo que permita realizar predicciones fiables de la variable objetivo.

8

MODELOS DE PREDICCIÓN

En este capítulo, presentaremos los distintos modelos de predicción utilizados en nuestro estudio. Para ello, dividiremos los modelos en dos categorías: modelos de ML y redes neuronales LSTM. Estudiaremos la base teórica de cada algoritmo, y comentaremos alguno de los problemas que han surgido, junto con las soluciones planteadas.

En primer lugar estudiaremos los modelos de Machine Learning, comenzando por los clásicos ARIMA y SARIMAX en su versión más simple y avanzando hasta incorporar la estacionalidad múltiple descubierta en el capítulo anterior. Posteriormente, experimentaremos con Prophet, un modelo ya introducido en capítulos anteriores. Por último dentro de esta categoría, nos adentraremos en modelos más complejos como Unobserved Components o TBATS.

Más adelante, nos centraremos en las redes neuronales LSTM, que tal como se estudió en el capítulo de fundamentos, utilizan una arquitectura especializada para procesar y predecir series temporales. Expondremos algunos de los resultados más interesantes obtenidos con este tipo de modelos, tanto univariables como multivariables y con diferentes ventanas de predicción.

8.1 MODELOS DE MACHINE LEARNING

Tal y como veníamos comentando, en esta primera sección vamos a exponer los modelos de ML que hemos considerado en el estudio. Se expondrán de manera general, pues los resultados concretos de la ejecución de los mismos se encuentran en el capítulo correspondiente de la parte experimental de este trabajo.

8.1.1 *ARIMA y SARIMAX*

Los modelos ARIMA y SARIMAX forman parte de los modelos autorregresivos de media móvil que fueron introducidos en capítulos anteriores.

ARIMA

Comenzaremos recordando la fórmula matemática de un proceso ARIMA, que es como sigue:

$$\begin{aligned} Y_t &= c + \alpha_1 Y'_{t-1} + \cdots + \alpha_p Y'_{t-p} + \phi_1 \epsilon_{t-1} + \cdots + \phi_q \epsilon_{t-q} + \epsilon_t \\ &= c + \sum_{n=1}^{p} \alpha_n X_{t-n} + \sum_{n=1}^{q} \phi_n \epsilon_{t-n} + \epsilon_t \end{aligned}$$

donde Y_t es la serie temporal que se quiere predecir, ϵ_t es la serie de residuos, p es el grado del polinomio autorregresivo, y q es el grado del polinomio de media móvil. Opcionalmente, se puede añadir un término $\sum_{n=1}^{r} \beta_n X_{n_t}$, por cada variable exógena X_t que se desee incluir.

Cabe destacar que los modelos ARIMA necesitan que la serie temporal sea estacionaria, y como hemos visto en apartados anteriores, no lo es, por lo que se realizará un paso de diferenciación, que permite que la serie cumpla este requisito, y cuyo grado viene representado por un parámetro d.

Los parámetros p, d y q se pueden ajustar manualmente, observando los gráficos de la función ACF y PACF, y determinando el grado correcto de cada uno de los polinomios. Sin embargo, resulta más preciso realizar una búsqueda por fuerza bruta, analizando todas las combinaciones posibles entre ellos, estableciendo un límite previo. Para realizar la búsqueda es necesario determinar una forma de comparar los modelos que nos permita elegir el más correcto. La forma más habitual para este cometido es utilizar el **Criterio de información de Akaike** (AIC, por sus siglas en inglés).

El criterio AIC proporciona una manera de comparar diferentes modelos y seleccionar el que mejor explica los datos originales. La diferencia principal con otras métricas ya introducidas como MAE o MAPE, entre otras, en que tiene en cuenta la complejidad de cada modelo. Las métricas usuales evalúan la precisión del modelo basándose en la diferencia entre el valor predicho y el valor real, pero sin tener en cuenta el número de parámetros del modelo. Un modelo con una gran cantidad de parámetros puede comportarse bien en un conjunto concreto de datos, pero no ajustarse bien a nuevos datos, generando un problema de *overfitting* o alta varianza.

La fórmula principal de AIC es la siguiente:

$$AIC = 2K - 2ln(L) \tag{8.1.1}$$

donde L es la función de máxima verosimilitud, y K es el número de parámetros del modelo.

El primer término es una penalización para los modelos con un gran número de parámetros, y el segundo mide la falta de ajuste del modelo. Esto nos lleva a deducir que un valor más bajo de AIC representa, en teoría, un modelo mejor.

Es fundamental asegurar que no se da un problema de sobreajuste en el modelo, y aunque el criterio AIC previene la aparición del mismo, existen otras técnicas para detectarlo y corregirlo.

La manera más sencilla de detectar un sobreajuste es comparar las medidas de rendimiento en el conjunto de entrenamiento y en el conjunto de testeo. En el caso de que el modelo presente sobreajuste, las métricas en el conjunto de entrenamiento serán considerablemente superiores.

En nuestro caso, tras ajustar con el criterio AIC los parámetros del modelo, no se llegó a predecir correctamente la variable objetivo. Inicialmente se pensaba que era debido a un problema de sobreajuste, pues dentro del conjunto de entrenamiento las predicciones eran correctas. No obstante, tras analizar el comportamiento del modelo y de la serie, se llegó a la conclusión de que un modelo ARIMA no era adecuado para predecir la variable objetivo.

La razón principal es que la serie temporal presenta varias componentes estacionales (horaria, diaria y semanal), y un modelo ARIMA simple no es capaz de procesarlas de manera correcta. Es por este motivo por el que originalmente se deseaba descomponer la serie, que ya vimos que fue imposible. Por tanto, se procedió a experimentar con SARIMAX, que sí que es capaz de modelar al menos una componente estacional.

SARIMAX

Ya hemos mencionado en múltiples ocasiones que SARIMAX modela series con componentes estacionales. Para hacerlo, se procede incluyendo dos términos que se encargan de explicar la estacionalidad, y que corresponden a los polinomios autorregresivos y de media móvil, en los que en vez de tomar $Y_{t-1}, ..., Y_{t-p}$, se toman $Y_{t-s}, ..., Y_{t-sp}$, donde s es el periodo de estacionalidad. La fórmula matemática quedaría como sigue:

$$Y_t = c + \sum_{n=1}^{p} \alpha_n Y_{t-n} + \sum_{n=1}^{q} \phi_n \epsilon_{t-n} + \sum_{n=1}^{P} \theta_n Y_{t-sn} + \sum_{n=1}^{Q} \eta_n \epsilon_{t-sn} + \epsilon_t \tag{8.1.2}$$

Sin embargo, es fácil darse cuenta de la limitación principal de utilizar este modelo en la serie temporal que representa el uso de energía, y es que no incluye múltiples estacionalidades. Para hacer frente a este problema hay dos soluciones: la primera, elegir un periodo de estacionalidad y perder el resto, y la segunda, incorporar las componentes estacionales como variable exógena.

Para poder incorporar todos los periodos de estacionalidad como variables exógenas utilizaremos Series de Fourier. Básicamente, buscamos modelar la estacionalidad por

medio de una función periódica, y al incorporarlo como variable exógena se consigue que el modelo tenga en cuenta la periodicidad.

Es importante que al calcular los términos de Fourier no se añada información de la variable objetivo. Si ocurriera esto, diríamos que se ha dado una filtración de datos, y el modelo obtenido no sería válido. Es por ello que solamente se usa una función de tiempo, es decir, para un determinado instante t, calculamos:

$$F(t) = [\sum_{i=1}^{M} sin(\frac{2\pi t}{T}), \sum_{i=1}^{M} cos(\frac{2\pi t}{T})] \tag{8.1.3}$$

donde M es el grado de la serie y T es el periodo de estacionalidad que deseamos incluir.

Es natural preguntarse si sería posible incluir la estacionalidad adicional mediante la incorporación de variables como el día de la semana, o la hora del día, de manera que la información respecto del periodo de estacionalidad esté implícita. Sin embargo, esta solución es incapaz de modelar patrones estacionales complejos. Es por este motivo por el que se utilizan términos de Fourier, que descomponen estos patrones en componentes sinusoidales con diferentes frecuencias y amplitudes, lo que resulta en una mejor capacidad para modelar estacionalidades no lineales o irregulares en los datos.

8.1.2 *Prophet*

El siguiente algoritmo de predicción es Prophet, que ya se introdujo anteriormente, y cuya fórmula matemática viene dada por:

$$Y_t = g_t + s_t + h_t + \epsilon_t \tag{8.1.4}$$

donde g_t representa la tendencia, s_t la estacionalidad, h_t el efecto de los días festivos y ϵ_t es la componente residual.

Ya se estudió que este modelo emplea series de Fourier para representar la estacionalidad, y si hay más de una componente, las puede modelar adecuadamente sin requerir el procedimiento descrito previamente.

No obstante, tras experimentar con este modelo, llegamos a la conclusión de que nuestros datos presentaban un problema que iba a dificultar el modelado con cualquiera de los algoritmos propuestos: una elevada varianza. Los modelos de predicción suelen tener dificultades para capturar la variabilidad de una serie con alta varianza, lo que puede llevar a errores significativos en la predicción. Además, cuando una serie temporal tiene una varianza alta, significa que hay una gran cantidad de fluctuaciones aleatorias que no están relacionadas con la tendencia general de la serie. Esto hace que sea difícil distinguir entre patrones aleatorios y verdaderos cambios en la serie.

Para reducir la varianza, normalmente se trata de reducir la amplitud de los valores extremos mediante transformaciones matemáticas de la serie. Una de las más usuales es la transformación logarítmica, que simplemente aplica esta función sobre la serie. Otra muy común es la transformación Box-Cox, cuya fórmula matemática es:

$$y_t^* = \begin{cases} \frac{(y_t^\lambda - 1)}{\lambda} & \text{si } \lambda \neq 0 \\ \ln(y_t) & \text{si } \lambda = 0 \end{cases} \tag{8.1.5}$$

donde Y_t es el valor de la serie original y λ es el parámetro de la transformación. La transformación de Box-Cox tiene como objetivo ajustar la distribución de la variable original para que se aproxime lo máximo posible a una distribución normal, lo que puede reducir la varianza. El valor de λ se determina empíricamente encontrando el valor que maximiza el logaritmo de la máxima verosimilitud de los datos transformados, también denominado log-verosimilitud.

Estas dos transformaciones nos ayudarán a la hora de reducir la varianza, aunque es importante evaluar su efecto en los datos antes de aplicarlas, pues no siempre serán efectivas.

8.1.3 *Unobserved Components*

A continuación vamos a introducir un modelo llamado Modelo de Serie Temporal Estructural, o *Unobserved Components*, en inglés. Este modelo se podría considerar como un modelo de regresión múltiple con coeficientes variables en el tiempo. Matemáticamente, es como sigue:

$$Y_t = \mu_t + s_t + c_t + \sum_{i=1}^{p} \phi_t Y_{t-i} + \sum_{j=1}^{m} \beta_j X_{jt} + \epsilon_t \tag{8.1.6}$$

donde μ_t representa la componente de tendencia, s_t la componente estacional, c_t la componente cíclica y ϵ_t el residuo.

Además, el polinomio $\sum_{i=1}^{p} \phi_t Y_{t-i}$ permite incluir términos autorregresivos, que representan la información de las observaciones pasadas, y el polinomio $\sum_{j=1}^{m} \beta_j X_{jt}$ representa el efecto de las variables exógenas que se quieran incluir.

A continuación vamos a exponer cómo se modelan las componentes principales.

Tendencia

A la hora de modelar la tendencia, se dan diversas alternativas, a continuación vemos dos de ellas:

- Modelar la tendencia como ruido blanco, esto es:

$$\mu_t = \mu_{t-1} + \eta_t \quad \text{donde} \quad \eta_t \sim N(0, \sigma_\eta^2) \tag{8.1.7}$$

- Utilizar una tendencia localmente lineal, para la que se asume que la tendencia varía linealmente a lo largo del tiempo pero puede cambiar de dirección en algunos puntos. Viene bien cuando la tendencia no está completamente definida, y matemáticamente viene definida por:

$$\mu_t = \mu_{t-1} + \beta_{t-1} + \eta_t \quad \text{donde} \quad \eta_t \sim N(0, \sigma_\eta^2) \qquad \textbf{Nivel}$$
$$\beta_t = \beta_{t-1} + \xi_t \quad \text{donde} \quad \xi_t \sim N(0, \sigma_{\xi_t}^2) \qquad \textbf{Pendiente}$$

Existen otras alternativas que siguen la segunda estructura, como la tendencia localmente determinista, que asume cambios específicos en ciertos puntos de tiempo en lugar de cambios arbitrarios en la dirección, o una tendencia suavizada que no permite cambios abruptos. Es evidente que existen diversas opciones para modelar la tendencia, por lo que es fundamental elegir la opción más adecuada para los datos en cada caso.

Diremos que los elementos η_t y ξ_t son elementos estocásticos, es decir, que presentan cierto grado de variabilidad, y por tanto se consideran como variables aleatorias. Para facilitar el modelado, se asume que siguen una distribución normal de media cero, y con varianza $\sigma_{\xi_t}^2$, que será un parámetro a estimar por el método de la máxima verosimilitud.

Estacionalidad

Este modelo considera una estacionalidad general, determinada por dos parámetros: el número de estaciones m y la varianza de un elemento estocástico añadido σ_ω^2, que formalmente sería:

$$s_t = -\sum_{i=1}^{m-1} s_{t-i} + \omega_t \quad \text{donde} \quad \omega_t \sim N(0, \sigma_\omega^2) \tag{8.1.8}$$

También permite incluir componentes estacionales con una frecuencia determinada (y no excluye la estacionalidad que acabamos de mencionar). Esta estacionalidad de nuevo se basa en funciones sinusoidales, y matemáticamente se modela como:

$$s_t = -\sum_{i=1}^{h} s_{j,t} \tag{8.1.9}$$

De nuevo s es el periodo de la estacionalidad, h es el número de armónicos que se desea obtener y $s_{j,t}$ viene dado por:

$$s_{j,t} = s_{j,t-1} cos(\frac{2\pi j}{m}) + s_{j,t-1}^* sin(\frac{2\pi j}{m}) + \omega_{j,t} \tag{8.1.10}$$
$$s_{j,t}^* = -s_{j,t-1} sin(\frac{2\pi j}{m}) + s_{j,t-1}^* cos(\frac{2\pi j}{m}) + \omega_{j,t}^* \tag{8.1.11}$$
$$\tag{8.1.12}$$

donde $\omega_{j,t}, \omega^*_{j,t} \sim N(0, \sigma^2_\omega)$.

El parámetro correspondiente a la varianza del elemento estocástico se estima también por el método de máxima verosimilitud.

Ciclo

Esta componente, que no aparece en los modelos aditivo y multiplicativo, representa efectos cíclicos con un periodo mucho mayor que el de las componentes estacionales. Formalmente, viene dado por:

$$c_{t+1} = \rho_c(\tilde{c}_t \cos \lambda_c t + \tilde{c}^*_t \sin \lambda_c) + \tilde{\omega}_t \tag{8.1.13}$$
$$c^*_{t+1} = \rho_c(-\tilde{c}_t \sin \lambda_c t + \tilde{c}^*_t \cos \lambda_c) + \tilde{\omega}^*_t \tag{8.1.14}$$

donde $\omega_t, \tilde{\omega}^*_t \sim N(0, \sigma^2_{\tilde{\omega}})$.

El número de parámetros en este caso, depende de la configuración escogida: sin entrar en detalle, si el ciclo es estocástico o amortiguado se añadirá un parámetro extra, $\sigma^2_{\tilde{\omega}}$ y p_c, respectivamente. De nuevo, estos parámetros se estimarán con la máxima verosimilitud.

8.1.4 *TBATS*

El último algoritmo de Machine Learning que vamos a introducir en este trabajo se denomina TBATS, donde las iniciales representan características del modelo

- T: Estacionalidad trigonométrica
- B: Transformación Box-Cox
- A: Modelado de los errores con $ARIMA$
- T: Componentes de tendencia
- S: Componentes estacionales

Por lo tanto, vemos que este modelo se puede utlizar para modelar series temporales con componentes estacionales complejas, que es justo lo que necesitamos en nuestro caso, tal y como hemos ido estudiando.

TBATS prueba diferentes combinaciones de características en en el modelo para determinar cuál es la mejor para ajustarse a los datos, comparándolas con la medida AIC. En particular, el modelo considera un modelo de suavizado exponencial simple, al que se le incorporan componentes con distintas características. Se probarán:

- La transformación Box-Cox
- Componentes de tendencia amortiguada y de tendencia, que introduciremos más adelante
- Un modelo $ARIMA$ para los errores
- Modelo estacional y no estacional
- Combinaciones de armónicos para las componentes estacionales

Comenzaremos estudiando brevemente en qué consiste un **modelo con suavizado exponencial**. Se trata de calcular una media ponderada de los valores pasados, dando mayor importancia a los valores más recientes. A medida que se avanza en el tiempo, la influencia de los valores pasados disminuye exponencialmente, lo que da lugar a un suavizado de la serie temporal. Matemáticamente, sería:

$$\hat{Y}_t = \alpha Y_{t-1} + (1-\alpha)\hat{Y}_{t-1} \tag{8.1.15}$$

donde α es el parámetro de suavizado, y $\hat{Y}_{t-1}$ es la predicción realizada en el instante anterior [14]. Es necesario establecer un valor l_0 que represente el valor predicho en el instante 0, y que será un parámetro a estimar.

Otra forma de estudiar el suavizado exponencial simple, y la que nos resulta más interesante de cara a TBATS, es la descompuesta, que es como sigue:

$$\begin{aligned} \hat{y}_{t+1} &= \ell_t \\ \ell_t &= \alpha y_t + (1-\alpha)\ell_{t-1} \end{aligned} \tag{8.1.16}$$

En este contexto, TBATS va incorporando características que convierten el modelo que acabamos de introducir en un modelo capaz de modelar series temporales complejas. Aunque vayamos a explicarlas de manera secuencial, en realidad se comprueban distintas combinaciones entre ellas de manera parecida a una búsqueda por fuerza bruta.

En primer lugar, como ya hemos mencionado, TBATS prueba la opción de incorporar una **transformación Box-Cox previa**.

Después, se incorporan **componentes de tendencia**, que no estaban presentes en el modelo simple. Para éstas, existen dos alternativas: una tendencia lineal y una amortiguada.

El modelo con tendencia lineal sería:

$$\begin{aligned} \hat{y}_{t+h} &= \ell_t + hb_t \\ \ell_t &= \alpha y_t + (1-\alpha)\left(\ell_{t-1} + b_{t-1}\right) \\ b_t &= \beta^*\left(\ell_t - \ell_{t-1}\right) + (1-\beta^*)\, b_{t-1}, \end{aligned} \tag{8.1.17}$$

donde además de los parámetros anteriores, se incorpora b_t, que representa la estimación de la tendencia en el instante t y β^*, el parámetro de suavizado.

Esta fórmula es más general que la anterior, pues se admiten predicciones de h pasos, es decir, predecimos con h instantes de anterioridad. Si ponemos $h = 1$, obtenemos el caso anterior.

La segunda alternativa para la tendencia fue introducida en [10], y se denomina tendencia suavizada, pues se añade un parámetro que hace que a largo plazo, la tendencia se convierta en constante. De manera formal sería:

$$\begin{aligned} \hat{y}_{t+h} &= \ell_t + \left(\phi + \phi^2 + \cdots + \phi^h\right) b_t \\ \ell_t &= \alpha y_t + (1-\alpha)\left(\ell_{t-1} + \phi b_{t-1}\right) \\ b_t &= \beta^* \left(\ell_t - \ell_{t-1}\right) + (1-\beta^*)\,\phi b_{t-1}. \end{aligned} \tag{8.1.18}$$

Se puede observar que se ha añadido el parámetro ϕ, y que si $\phi = 1$, obtenemos el modelo anterior.

La siguiente característica que se prueba en TBATS es el **modelado de los errores** mediante $ARIMA$, donde se añade una componente residual r_t, que se modela con $ARIMA$. Los parámetros p, q y d forman parte del conjunto general de parámetros a estimar. Formalmente, sería:

$$\begin{aligned} y_{t+1} &= \ell_t + \phi b_t + r_{t+1} \\ \ell_t &= \ell_{t-1} + \phi b_{t-1} + \alpha r_t \\ b_t &= (1-\phi) b + \phi b_{t-1} + \beta r_t \\ r_t &= \sum_{i=1}^{p} \varphi_i r_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t \end{aligned} \tag{8.1.19}$$

Se puede observar que se ha tomado $h = 1$, y que la última fila de la expresión representa un modelo $ARIMA$ usual. Hay otros cambios con respecto a la formulación anterior, no entraremos en detalle en ellos, pues la intuición detrás de cada componente sigue siendo la misma. Cabe destacar la constante b, que representa la tendencia a largo plazo, mientras que b_t representa la tendencia a corto plazo.

Por último, estudiamos la inclusión de T componentes estacionales, que se pueden modelar o bien de una manera lineal, dando lugar a (se han omitido el resto de componentes para evitar redundancias):

$$\begin{aligned} y_{t+1} &= \ell_t + \phi b_t + \sum_{i=0}^{T} s_{t-m_i}^{(i)} + r_{t+1} \\ s_t^{(i)} &= s_{t-m_i}^{(i)} + \gamma_i r_t \end{aligned} \tag{8.1.20}$$

O bien se pueden modelar con funciones trigonométricas, y entonces cada $s_{t-m_i}^{(i)}$, denominado a continuación s_t por simplicidad, vendrá dado por:

$$s_t = \sum_{i=1}^{h_i} s_{j,t} \tag{8.1.21}$$

$$s_{j,t} = s_{j,t-1} cos(\frac{2\pi j}{m_i}) + s_{j,t-1}^* sin(\frac{2\pi j}{m_i}) + \gamma_1 r_t \tag{8.1.22}$$

$$s^*_{j,t} = -s_{j,t-1} sin(\frac{2\pi j}{m_i}) + s^*_{j,t-1} cos(\frac{2\pi j}{m_i}) + \gamma_2 r_t \tag{8.1.23}$$

$$\tag{8.1.24}$$

donde m_i es el periodo de la componente $s^{(i)}_{t-m_i}$, y h_i es el número de armónicos que incorpora la serie de Fourier.

Una vez se ha comprendido correctamente qué pretende la inclusión de cada una de las componentes en el modelo TBATS, es sencillo entender el procedimiento de ajuste del modelo. Se genera un conjunto de modelos donde cada uno tiene una configuración distinta, y se escoge el mejor según el criterio AIC.

8.2 LSTM

En esta sección nos adentraremos en el estudio de las redes neuronales LSTM (*Long Short-Term Memory*), una estructura que ha demostrado ser altamente efectiva en la resolución de problemas relacionados con series temporales. En primer lugar, describiremos su arquitectura básica, detallando cada uno de sus componentes y su funcionamiento. Además, se explicarán las matemáticas subyacentes que permiten a esta estructura retener información relevante de una secuencia temporal durante un período de tiempo prolongado.

Las redes neuronales LSTM forman parte de un conjunto de redes neuronales denominado redes recurrentes (RNNs), una clase de redes que se utilizan para procesar secuencias de datos, como texto o series temporales. Sin embargo, el problema de explosión del gradiente puede dificultar su entrenamiento, especialmente en secuencias largas. Este problema ocurre durante cuando el valor del gradiente se vuelve extremadamente grande. Esto puede llevar a que los pesos de la red neuronal se actualicen de manera muy brusca, lo que a su vez puede provocar que el modelo no converja o que tenga un desempeño pobre.

Para abordar este problema, se han desarrollado las redes neuronales LSTM, que son una variante de las RNNs capaces de recordar información a largo plazo y prevenir la desaparición o explosión del gradiente.

8.2.1 *Arquitectura LSTM*

En secciones anteriores veíamos que una red neuronal está compuesta por capas que a su vez incorporan unidades o neuronas. Las redes LSTM disponen de bloques de memoria que sustituyen a las neuronas.

Un bloque de memoria tiene componentes que lo convierten en una estructura más inteligente que las neuronas simples con una función de activación presentes en las redes neuronales usuales. Los bloques de memoria se comunican entre sí a través de

puertas (*gates*) para controlar qué información se retiene y qué se olvida en cada paso temporal.

Las componentes de un bloque de memoria son:

- Un estado de celda que almacena información a largo plazo, que se actualiza y modifica a través de las puertas, y que será notado por c_t.
- Un estado de celda oculto, que es una representación comprimida del estado de celda, será notado por h_t.
- Una serie de puertas, que se abren o se cierran en función de las entradas, lo que permite controlar la cantidad de información que fluye hacia el estado de celda y hacia las salidas de la capa y de esta manera, la red puede aprender a recordar información importante y a ignorar la información innecesaria en una serie temporal. Existen tres tipos de puerta en un bloque:
  - **Puerta de olvido:** decide qué información almacenada en el bloque se pierde.
  - **Puerta de entrada:** decide qué información de entrada se utiliza para actualizar el estado.
  - **Puerta de salida:** decide la salida, basándose en la entrada y en el estado del bloque.

En este tipo de red neuronal, los pesos están presentes en las puertas de cada bloque de memoria, y son lo que permite que se aprendan los patrones entre las entradas y las salidas de la red. En la Figura 42 se puede comprobar el detalle del bloque de memoria.

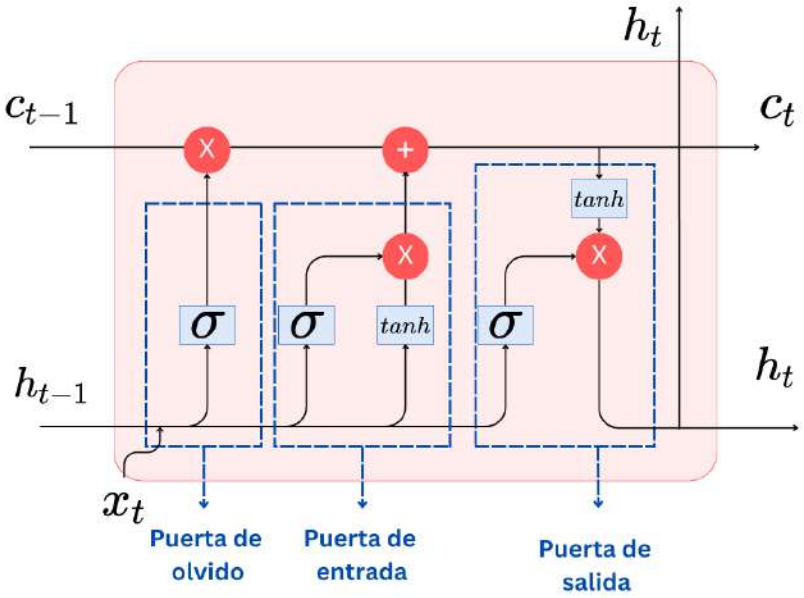

Figura 42: Estructura interna de un bloque de memoria

A continuación vamos a explicar dónde se encuentran los pesos y los sesgos en este tipo de arquitecturas.

Comenzamos por la **puerta de olvido**, donde se combina la entrada x_t y el estado oculto de celda anterior h_{t-1}, produciendo como salida un valor f_t, mediante la siguiente expresión:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{8.2.1}$$

donde W_f representan los pesos asociados con la entrada y con el estado oculto anterior, que se han concatenado, y b_f representa el sesgo. Cabe destacar el uso de la función sigmoide, que lo convierte en un valor entre 0 y 1. Más adelante, se multiplicará por el estado de celda anterior, y es donde se establecerá qué información del estado anterior es descartada.

Continuamos introduciendo en detalle la **puerta de entrada**. Se puede dividir en dos partes, la primera se encarga de valorar la importancia de la información que aporta la entrada, y la segunda se encarga de transformar la información.

Para la primera, se vuelve a realizar la misma operación, esta vez con pesos distintos:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{8.2.2}$$

Y para la segunda, simplemente hay que utilizar la función *tanh* para que los valores se encuentren en el intervalo $[-1, 1]$:

$$n_t = \tanh(W_n[h_{t-1}, x_t] + b_n) \tag{8.2.3}$$

Una vez se ha obtenido la salida de la puerta de olvido y de la puerta de entrada, se puede proceder a realizar la actualización del estado de celda, que es como sigue:

$$c_t = f_t * c_{t-1} + i_t * n_t \tag{8.2.4}$$

Por último, se tiene que calcular la salida de la celda, por medio de la puerta de salida, de nuevo mediante la expresión:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{8.2.5}$$

Para calcular el siguiente estado oculto de la celda, se vuelve a utilizar *tanh* para transformar el estado anterior, c_t:

$$h_t = o_t * tanh(c_t) \tag{8.2.6}$$

Aunque nos hemos fijado en el detalle de una sola unidad o celda, en realidad una red neuronal LSTM está compuesta por capas donde cada capa tiene m unidades. A nivel práctico, en una capa únicamente hay un bloque de memoria, por lo que resulta natural preguntarse cómo puede darse una situación en la que haya m unidades. Para responder a esta cuestión, es necesario romper con la asociación entre neurona y bloque de memoria, y considerar el propio bloque como una capa, donde la dimensión del estado oculto será m. Esto quiere decir que, en principio, cuantas más unidades

tenga una capa LSTM, más complejos (dimensiones más altas) serán los patrones que podrá recordar.

8.2.2 *Entrenamiento de redes LSTM*

Acabamos de ver un ejemplo de cómo se obtendrían c_t y h_t en una celda, y sabemos que este proceso forma parte del *forward propagation.*

Recordamos que en las redes neuronales tenemos un conjunto de entrenamiento dividido en lotes o *batches*, a través del cual se realizan iteraciones, o *epochs*. Una vez se ha terminado de procesar todas las entradas de un lote, se actualizan los pesos de cada capa. Además, una capa únicamente está formada por un bloque de memoria. La estructura al tener p capas viene dada por p bloques de memoria interconectados, donde el estado oculto de la capa i sirve como entrada para la capa $i + 1$.

En realidad, en LSTM no se procesan las observaciones de manera independiente, sino que se procesan en conjuntos secuenciales, por lo que en cada capa obtenemos un conjunto de estados ocultos, que sirven como entrada para la siguiente capa.

El *backpropagation* se produce una vez se han obtenido las salidas correspondientes a las observaciones de un lote, y el procedimiento es idéntico al de las redes neuronales usuales: se calcula el gradiente de la función de coste con respecto a cada una de las matrices de pesos, desde el último bloque de memoria hasta el primero, y se actualizan siguiendo un algoritmo de optimización, como el descenso del gradiente. En este caso, los cálculos para obtener los gradientes son mucho más complejos debido a la arquitectura de cada bloque de memoria, por este motivo no entraremos en detalle en ellos.

Podemos dar por finalizada esta parte del trabajo, que supone un bloque fundamental para comprender los experimentos que se han llevado a cabo, y que se revisarán de manera exhaustiva más adelante.

Parte IV

EXPERIMENTACIÓN

Exposición de los experimentos llevados a cabo para definir el algoritmo de imputación de datos. Estudio de técnicas de predicción mediante algoritmos de Machine Learning univariable y multivariable y redes neuronales LSTM.

9

ALGORITMO DE IMPUTACIÓN DE DATOS

En este capítulo, se presentarán los experimentos llevados a cabo para definir de manera adecuada el algoritmo de imputación de datos presentado en capítulos anteriores. Como se ha discutido anteriormente, la imputación de datos es una tarea crítica en el procesamiento de datos faltantes, y es un error muy habitual cuando se tratan datos obtenidos por medio de sensores, pues son muy sensibles a fallos. Por lo tanto, la implementación de un algoritmo de imputación que permita solucionar este problema es fundamental para el futuro desarrollo de modelos de predicción.

9.1 PRESENTACIÓN DE PROBLEMÁTICAS

En esta sección haremos un recorrido a través de las incógnitas principales que surgieron al implementar el algoritmo de imputación, y presentaremos los experimentos concretos en cada caso que nos llevaron a solucionarlas.

Recordamos brevemente las bases del algoritmo planteado, descrito en detalle en el Algoritmo 1. Cada hueco inferior a una hora se imputará mediante interpolación lineal, y para los superiores a una hora se generarán los datos utilizando los valores anteriores y posteriores al hueco mediante un algoritmo de Machine Learning que los transforma en datos multidimensionales con forma de matriz.

La primera incógnita que surge por lo tanto, es qué algoritmo de Machine Learning se debe utilizar para imputar los datos.

La restricción principal del planteamiento básico es que para rellenar el hueco completo, es necesario que la longitud de los datos anteriores y posteriores sea mayor o igual a la longitud del hueco. Esta restricción es fundamental, pues no siempre se verifica, y para que el algoritmo sea lo más escalable posible, es necesario solventarla.

Para ello, vamos a presentar las situaciones en las que esta restricción no se cumpliría.

- En primer lugar, si el primer hueco está demasiado cerca del inicio de la serie, no dispondríamos de datos anteriores suficientes.
- De la misma manera, si el último hueco está demasiado cerca del final de la serie, no dispondríamos de datos posteriores.

- Si entre dos huecos no hay distancia suficiente, podríamos no tener datos posteriores para el primer hueco o datos anteriores para el segundo hueco (o las dos situaciones a la vez).

9.1.1 *Primer problema: elección de algoritmo ML*

La primera incógnita surgió a la hora de decidir qué algoritmo usar en el caso base, esto es, cuando haya datos suficientes para entrenar un modelo con los datos anteriores y otro con los posteriores.

En [23] se proponía utilizar modelos de tipo *Random Forest* o *Support Vector Regression*, por lo que decidimos compararlos. Para ello, se tomaron segmentos sin fallos de datos de la primera variable de consumo energético mels_S (consumo misceláneo en el ala Sur), y se imputaron valores nulos de manera artificial.

En primer lugar, se imputó un fallo de dos días de longitud, esto es, un total de 192 observaciones.

Figura 43: Hueco generado artificialmente

En la Figura 44 se puede observar que las predicciones de ambos modelos sobre un hueco artificial pequeño (Figura 43) resultan muy parecidas, y aunque no llegan a los valores extremos de la serie, sí que llegan a captar la periodicidad de la misma, factor esencial a la hora de hacer predicciones futuras.

Al repetir el experimento sobre otro hueco artificial de dos días de longitud, obtenemos las predicciones de la Figura 45. Podemos observar que, de nuevo, no llega a captar los valores extremos, pero SVR se acerca un poco más. Además, se puede comprobar también que cuando el patrón periódico se para, el modelo no es capaz de adaptarse correctamente.

Se repitió el experimento, esta vez generando un hueco más grande, de 10 días de longitud, es decir 960 observaciones (Figura 46). Los resultados obtenidos se pueden estudiar en la Figura 47, donde claramente el modelo SVR tiene una mejor eficiencia en cuanto a los valores predichos.

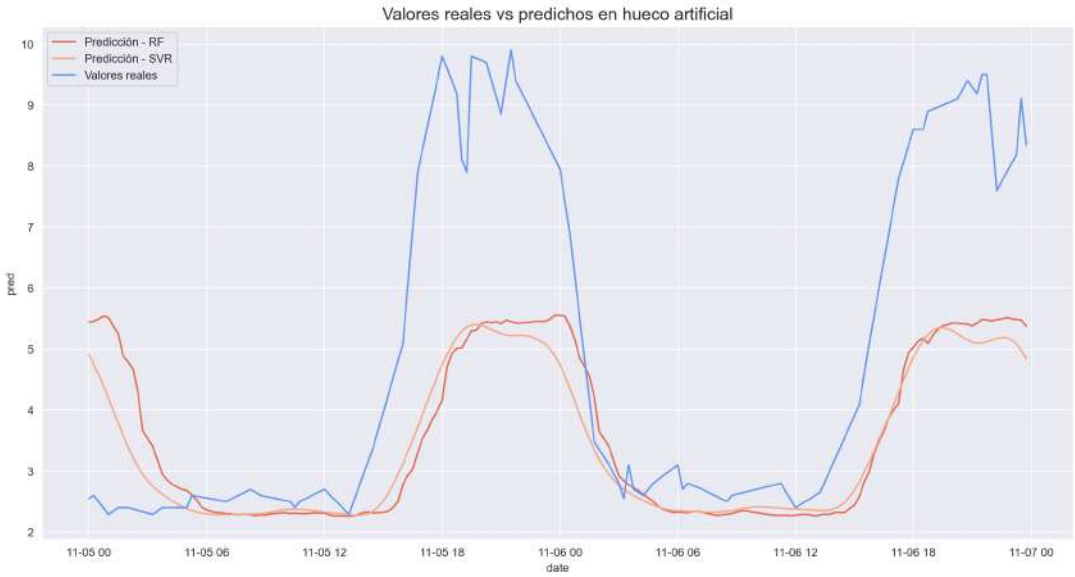

Figura 44: Datos obtenidos mediante el algoritmo de imputación - SVR vs RF

Además, cabe destacar que el tiempo de entrenamiento crece a medida que aumenta el tamaño del hueco, debido a que aumenta también el número de datos utilizados para entrenar el modelo.

Aunque el tiempo de entrenamiento no tiene por qué ser sistemáticamente más alto en uno u otro, al realizar los experimentos anteriores comprobamos que el algoritmo SVR es mucho más rápido que Random Forest. Hay algunas razones para ello. La principal es que SVR utiliza un subconjunto de los datos de entrenamiento para entrenar el modelo, mientras que RF utiliza todos los datos de entrenamiento. Además, la complejidad del modelo en SVR puede ser menor que la de RF, ya que SVR intenta encontrar un hiperplano que maximice la distancia entre los puntos de datos, mientras que RF puede tener muchos árboles de decisión, cada uno de los cuales puede ser bastante complejo.

Otro factor que puede afectar el tiempo de ejecución es la implementación específica del algoritmo. En general, las implementaciones de SVR son más eficientes que las de RF porque el cálculo de las distancias entre los puntos de datos es más simple que la construcción de múltiples árboles de decisión.

Por todos estos motivos, decidimos escoger *Support Vector Regression* para imputar los datos.

9.1.2 *Segundo problema: cantidad de datos*

Una vez se decidió qué algoritmo de ML utilizar, y teniendo en cuenta que el número mínimo de datos anteriores y posteriores que se tienen que utilizar es la longitud del hueco, debemos decidir cuál será el límite superior, es decir, suponiendo que tenemos datos suficientes, cuántos cogemos de más.

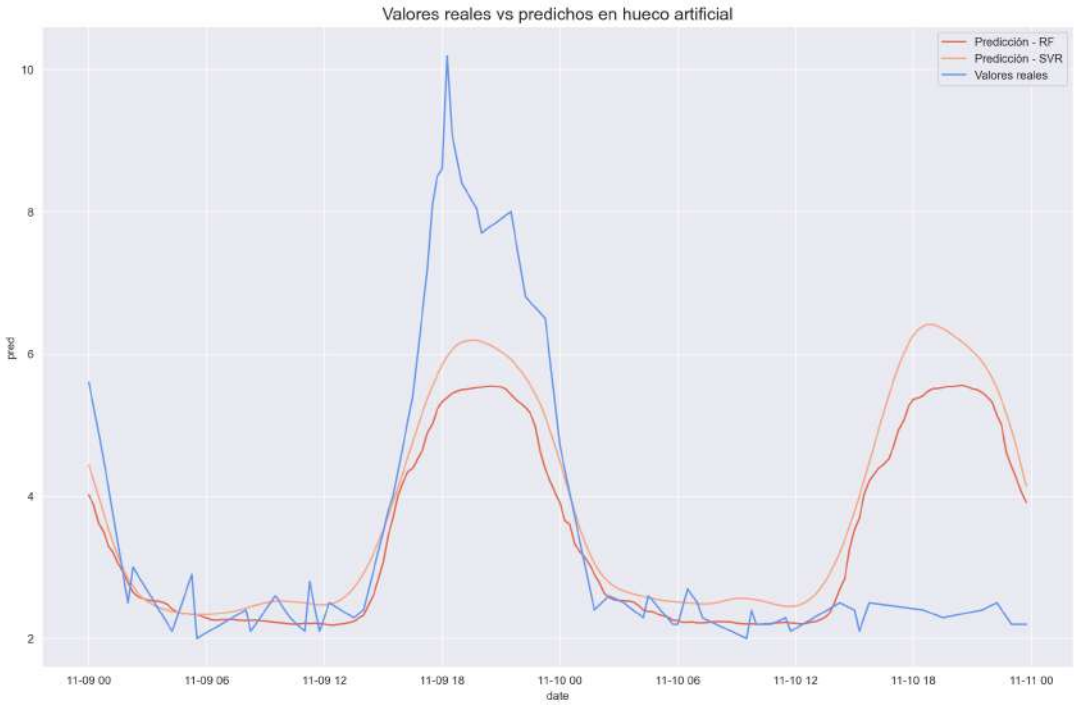

Figura 45: Segunda prueba de predicción

En general, se podría decir que a mayor cantidad de datos se tiene acceso a una mayor cantidad de información, lo que puede mejorar la calidad de las predicciones realizadas. Sin embargo, esto no siempre es el caso, ya que la cantidad de datos puede no ser un factor determinante de la precisión del modelo.

Esto es debido a que al incrementar el número de datos de entrenamiento, existe la posibilidad de incluir observaciones ruidosas que pueden influir en el modelo, haciendo que el modelo se ajuste demasiado a estos datos específicos y tenga un rendimiento deficiente en nuevos datos de prueba. Podemos decir, por tanto, que al aumentar la cantidad de datos se puede presentar un problema de *overfitting* o sobreajuste.

Es importante tener en cuenta que, como comentábamos antes, a medida que aumenta la cantidad de datos, el tiempo de procesamiento requerido para entrenar y ajustar el modelo también puede aumentar, lo que puede ser prohibitivo en términos de recursos computacionales.

Por tanto, el número máximo de datos de entrenamiento supuso la segunda incógnita de implementación del algoritmo de imputación.

El experimento llevado a cabo para solucionar este problema consistió en generar de manera independiente 20 huecos de longitudes 1 a 10 días en una ubicación aleatoria de una serie temporal sin fallos (variables de clima exterior).

Para cada hueco de longitud m días, se imputaron los datos utilizando n días de datos anteriores y posteriores, donde:

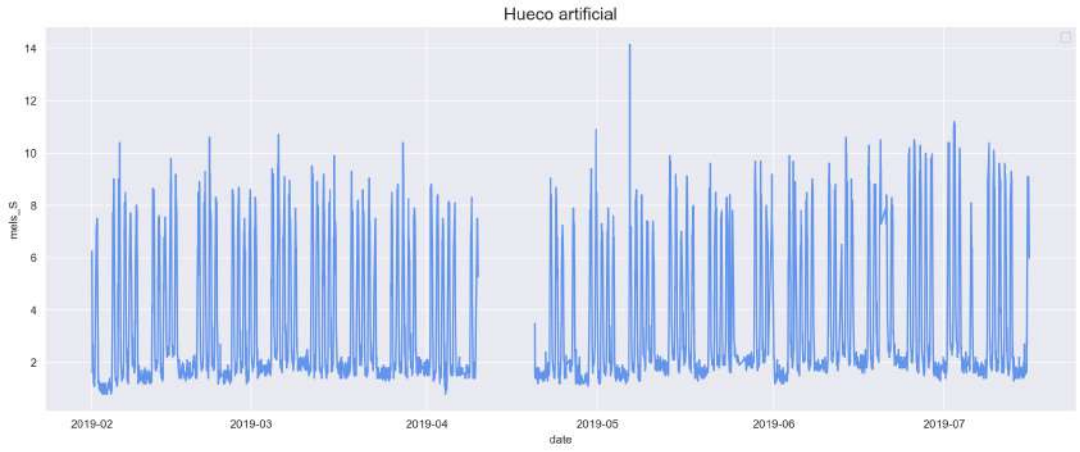

Figura 46: Hueco generado artificialmente

Figura 47: Datos obtenidos mediante el algoritmo de imputación - SVR vs RF

- $n = m$, es decir, misma cantidad de datos
- $n = 2m$, se toman dos días de información por cada día sin datos
- $n = 7m$, una semana de información por cada día sin datos
- $n = 14m$, dos semanas de información por cada día sin datos
- $n = 21m$, tres semanas de información por cada día sin datos

Las opciones consideradas están basadas en las componentes estacionales presentes en la serie (día, semana), que nos llevan a pensar que pueden ser las opciones más adecuadas.

Tras imputar los datos, se compararon los resultados obtenidos, resumidos en la Figura 48. Podemos observar que los modelos con más información (desde una semana hasta tres semanas) tienen un tiempo de ejecución mucho mayor que el resto cuando la longitud del hueco incrementa.

Además, con respecto al error cuadrático medio (MSE), aunque los resultados son muy parecidos, el modelo que utiliza dos días de información por cada día de fallo suele ser el que tiene un MSE menor (Figuras 48 y 49) .

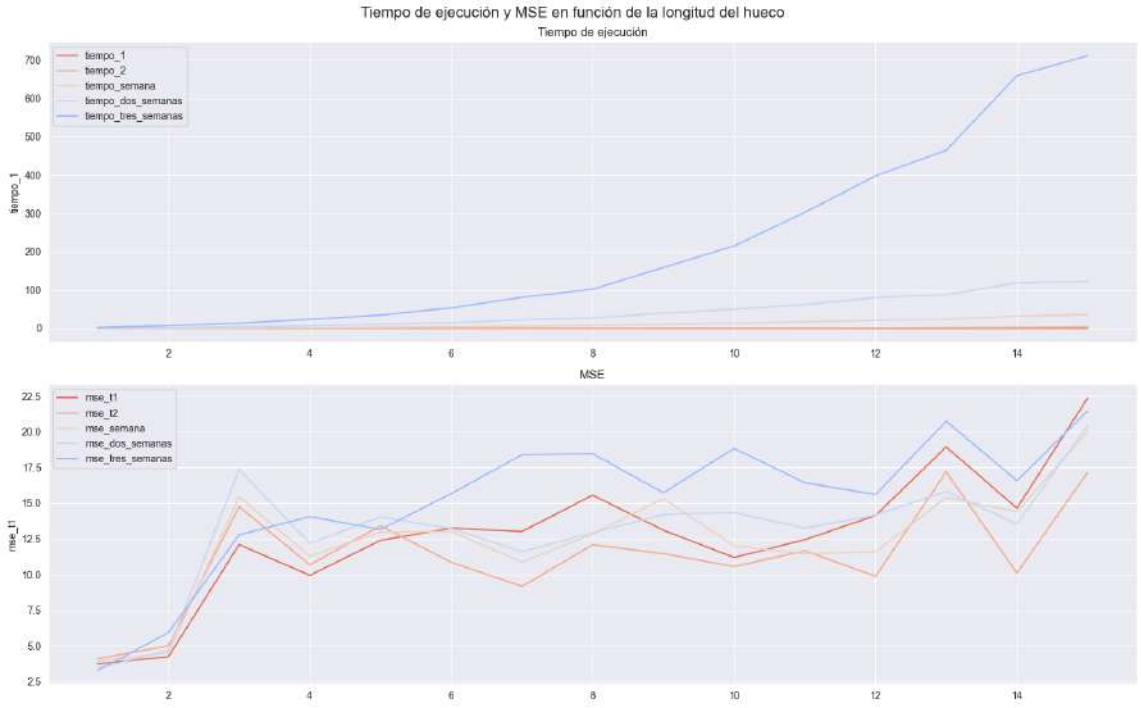

Figura 48: Resumen del experimento base

Por los resultados obtenidos se decidió que, en la medida de lo posible, se utilizarán dos días de información previa y posterior por cada día de datos faltantes. Si no hay suficientes datos disponibles para tomar dos días de información, se utilizarán los datos disponibles, siempre y cuando la longitud de los mismos sea mayor que la longitud del espacio vacío que se pretende llenar, es decir, la cantidad de datos estará siempre entre m y $2m$.

9.1.3 *Tercer problema: fusión de huecos*

El tercer problema surge cuando se incumple el límite inferior de cantidad de datos anteriores y posteriores disponibles. Se va a separar esta casuística en dos distintas, que conformarán el tercer y cuarto problema, respectivamente.

La primera de ellas se da cuando no hay datos suficientes entre huecos. Esta situación se da cuando estamos en un hueco H_i, de longitud T_i, y la distancia entre H_i y el siguiente, H_{i+1} es menor que T_i.

Se podría considerar también la situación en la que la distancia es menor que la longitud del segundo hueco, por lo que no habría suficientes datos anteriores. Sin embargo, como se van a ir imputando los datos de manera secuencial, podemos asumir que esta situación no se va a dar, y que a partir del segundo hueco, siempre hay datos anteriores suficientes disponibles. De hecho, en todo el conjunto de datos considerado, este problema no ha estado presente.

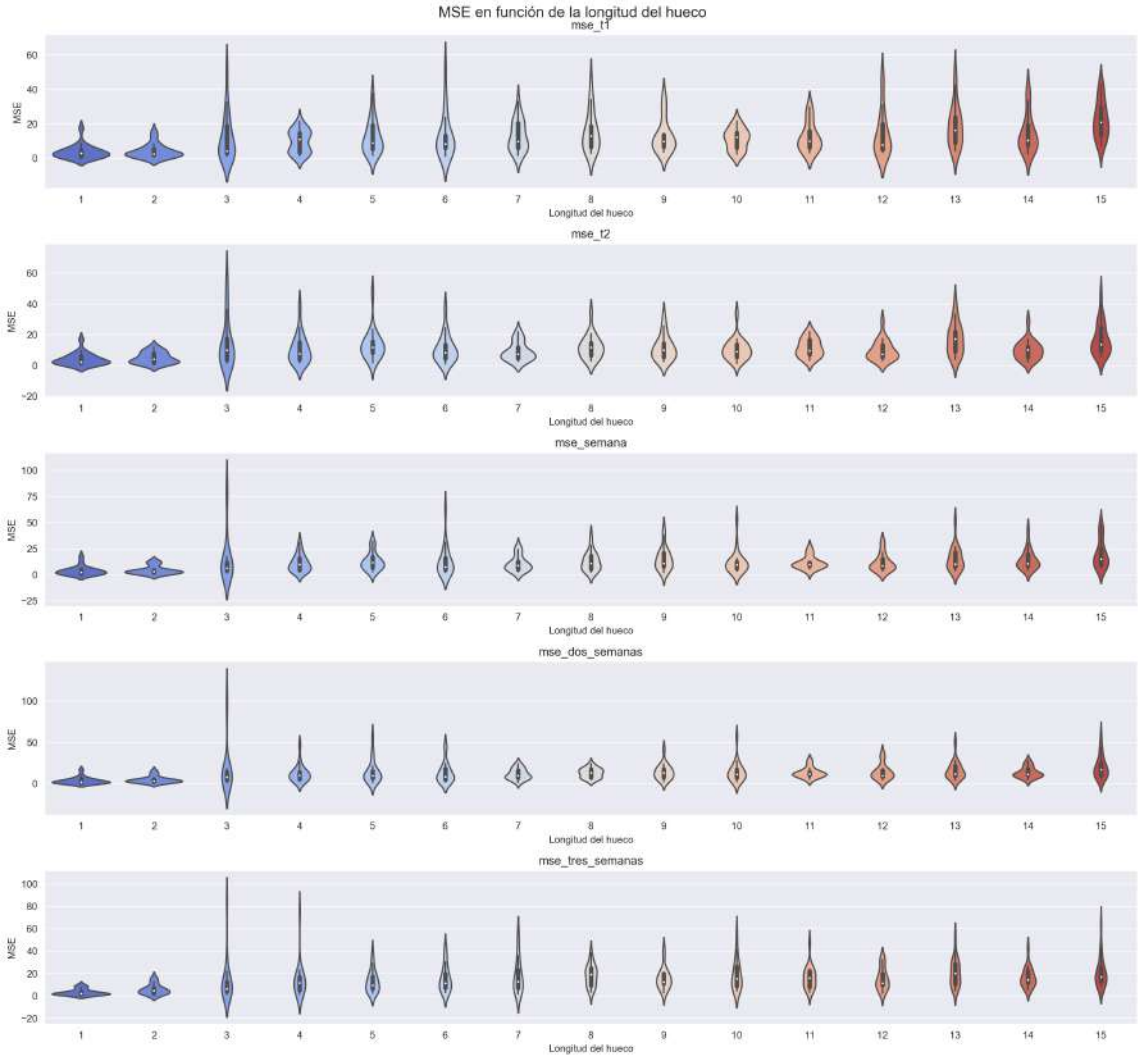

Figura 49: MSE en detalle del experimento base

Volviendo a la problemática de datos insuficientes entre huecos, vamos a solucionarlo fusionando los huecos, es decir, en el ejemplo anterior pasaríamos a considerar un único hueco T_j de longitud $T_i + T_{i+1} + distancia$. Es evidente que esta solución plantea dos problemas nuevos:

- El primero, es que al fusionar huecos, si queremos aplicar el algoritmo base descrito anteriormente, necesitaremos muchos más datos anteriores y posteriores, y puede dar lugar a bucles de fusiones muy difíciles de cortar, pues cada vez se necesitan más datos para dejar de fusionar.
- El segundo, es que si usamos el algoritmo SVR, estamos perdiendo la información intermedia, pues se considera únicamente la anterior y posterior.

Para descubrir cómo se podía solucionar este problema, se introdujeron una serie de huecos artificiales en la serie temporal descrita anteriormente.

En primer lugar se generaron dos, cinco y diez huecos sujetos a ser fusionados, es decir, que la distancia entre ellos era siempre inferior a la longitud del hueco anterior, o la longitud acumulada al fusionar los huecos anteriores.

Se decidió comparar la imputación al utilizar el algoritmo SVR con dos días de información y con una semana de información, ambas opciones estudiadas en detalle en el apartado anterior, y también al utilizar Prophet.

La justificación del uso de este último algoritmo es que al ser un modelo específico para series temporales, tiene una dependencia temporal que resulta muy útil en este caso particular, pues permite que se utilicen todos los datos no nulos de la serie, incluyendo los datos presentes entre los huecos fusionados. Además, el tiempo de entrenamiento al utilizar toda la serie es razonable, por lo que aparentemente podría ser la opción adecuada en casos en los que se tienen que imputar muchos datos muy próximos.

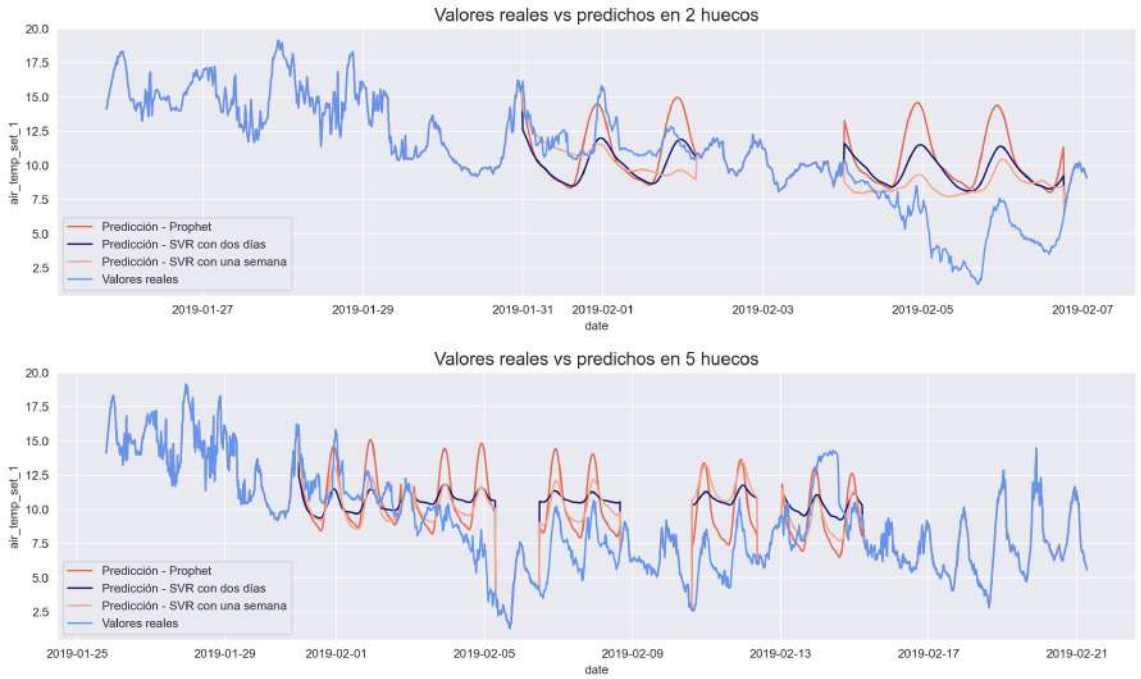

Figura 50: Imputación en huecos fusionados

En la Figura 50 se puede observar como en general el algoritmo SVR tiene un rendimiento mejor, aunque es necesario destacar la alta variabilidad de los resultados en función de la ubicación y longitud del hueco.

Para obtener conclusiones más generales, se llevó a cabo un experimento generando huecos susceptibles de ser fusionados, de longitud aleatoria siguiendo una distribución normal de media 250 y desviación típica 50, es decir, de media 2.6 días.

Para cada número de huecos a fusionar se repitió el experimento 20 veces, y los resultados medios obtenidos se pueden observar en la Figura 51, donde se ve que a partir de 3 huecos fusionados, generalmente es mejor utilizar el algoritmo Prophet. Por lo tanto, se procederá de esa manera en la implementación final.

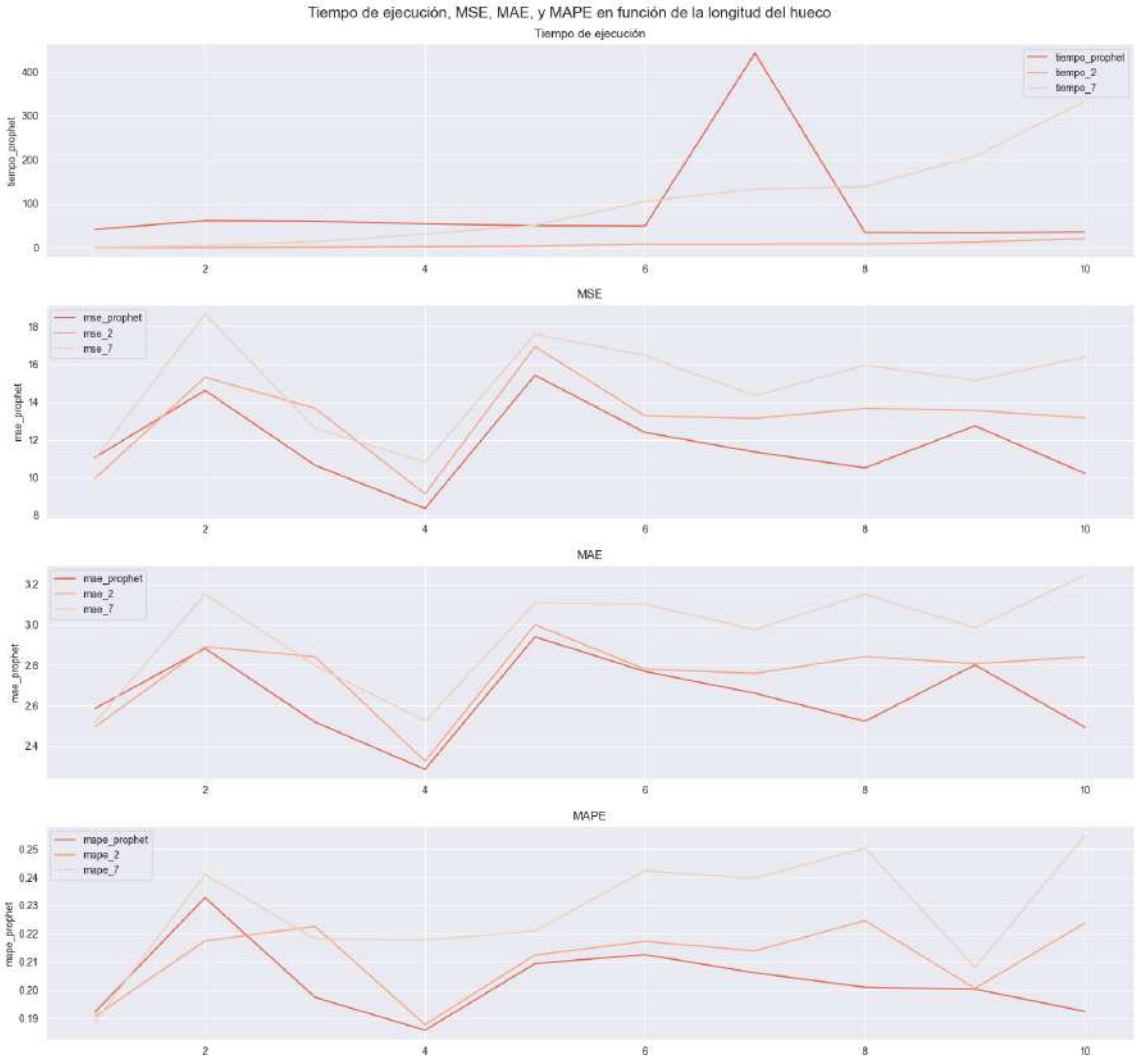

Figura 51: Resumen del experimento para el tercer problema

9.1.4 *Cuarto problema: huecos en situaciones extremas*

La última situación que todavía no hemos abordado es aquella en la que para el primero o el último hueco no hay datos suficientes (anteriores y posteriores, respectivamente).

Surge entonces una incógnita: ¿es suficiente con utilizar únicamente los datos disponibles o resultaría mejor utilizar un modelo como Prophet, que además se puede entrenar en toda la serie?

Para resolver la duda, se generaron huecos en los que no había datos posteriores suficientes (Figura 52), y compararon los resultados de imputar los datos nulos con las tres alternativas anteriores: el algoritmo SVR con dos días de información y con una semana de información, y Prophet.

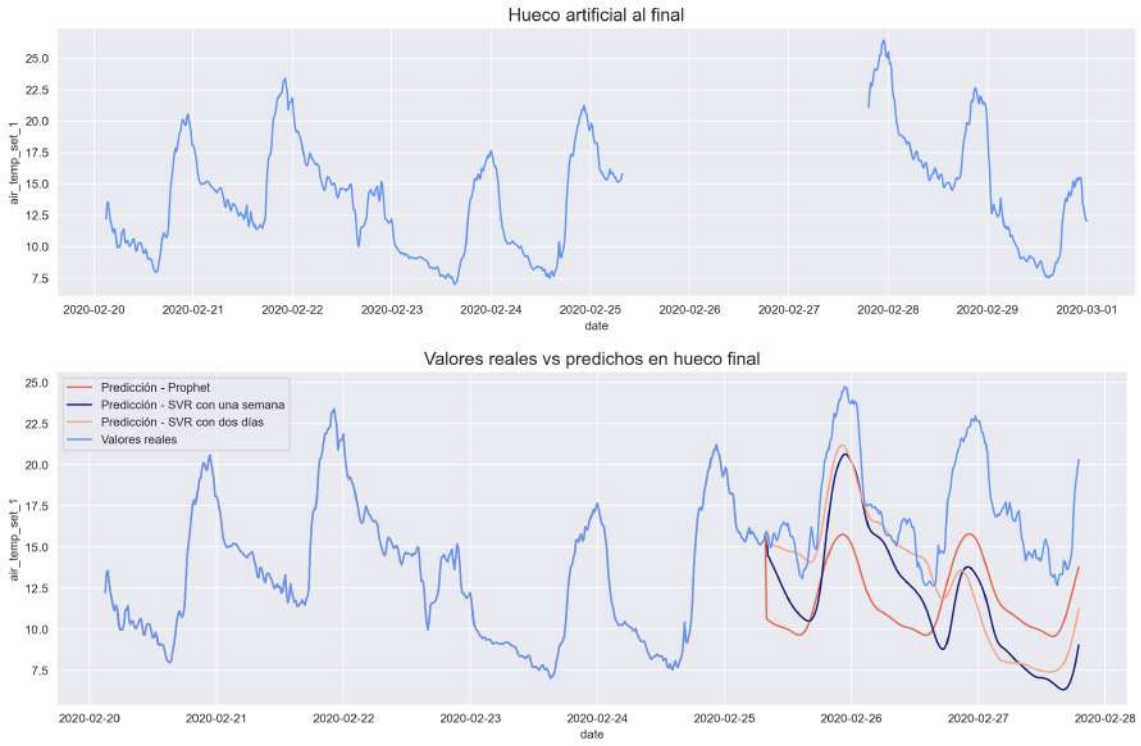

Figura 52: Hueco artificial al final

Los resultados concretos de una iteración se pueden estudiar en la Figura 52, donde no se puede destacar ninguna de las tres alternativas como superior al resto en precisión.

Para cada longitud se repitió el experimento 20 veces, obteniendo los resultados expuestos en la Figura 53, que demuestra la igualdad de los algoritmos planteados en cuanto a las métricas utilizadas. Sin embargo, vemos que si bien el tiempo al utilizar SVR es proporcional al número de datos que se utilicen para entrenarlo, la diferencia entre el tiempo máximo de SVR y Prophet es bastante grande.

Por este motivo, vamos a mantener el uso de SVR con dos días de información en los huecos que cumplan esta condición.

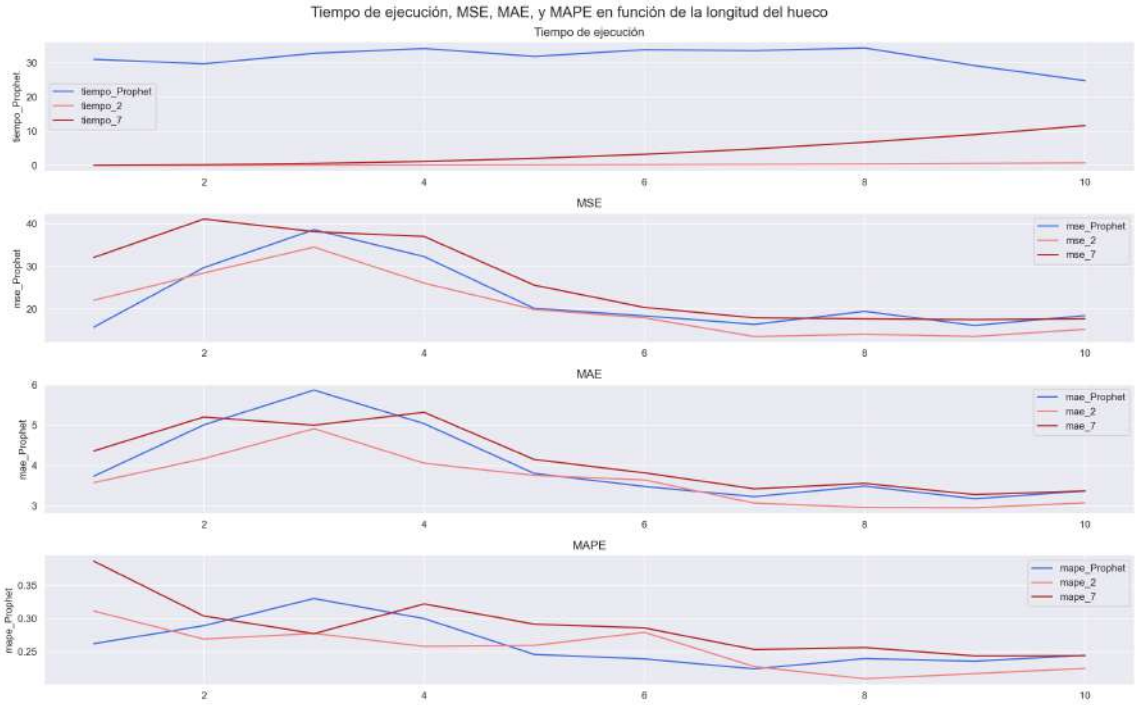

Figura 53: Resumen del experimento para el cuarto problema

9.2 ANÁLISIS DE RESULTADOS

En esta sección, se estudiarán los resultados obtenidos al aplicar el algoritmo sobre las series temporales consideradas.

No estudiaremos los datos correspondientes al clima exterior, ya que el algoritmo solo ha imputado un total de 35 datos, lo que representa una proporción insignificante del conjunto completo de datos. En su lugar, nos centraremos en los datos de uso de energía y clima interior.

Los resultados obtenidos en los datos de uso de energía (Figura 54) son bastante correctos a primera vista, lo que resulta lógico teniendo en cuenta que varios de los experimentos se realizaron sobre fragmentos de estas variables.

Sobre los datos de clima interior, es necesario realizar una distinción entre los datos de temperatura interior y exterior por zonas y los datos de temperatura umbral de calefacción y refrigeración. Para los datos de temperatura interior y exterior, los resultados también son consistentes, se puede comprobar en la Figuras 55 y 56.

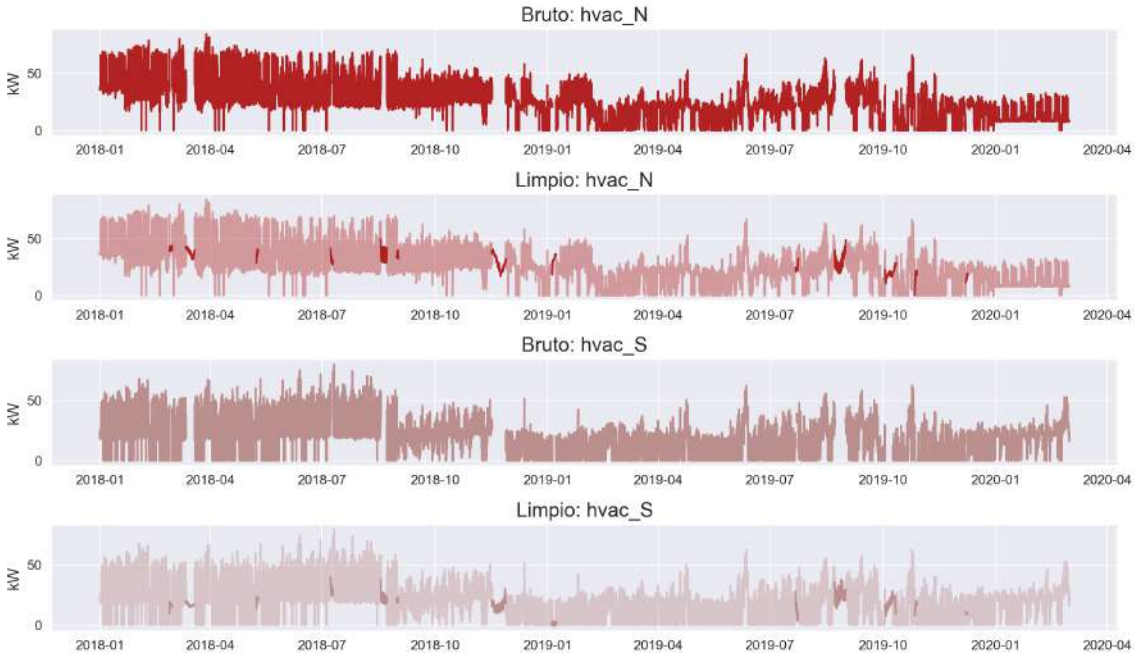

Figura 54: Ejemplo de imputación sobre datos de uso de energía

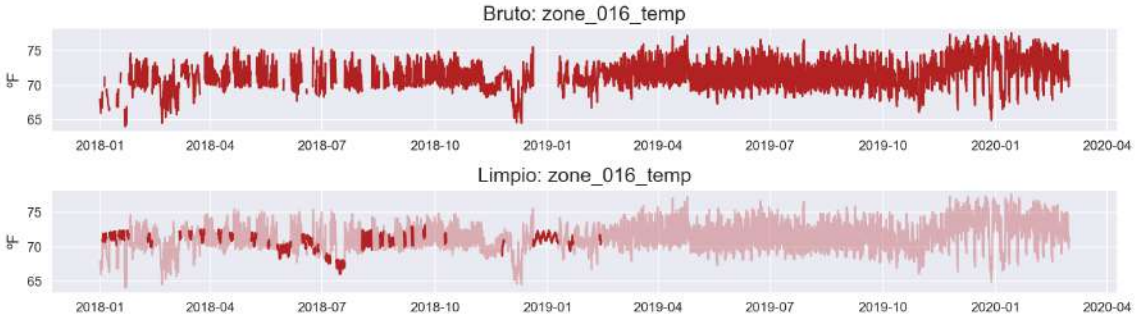

Figura 55: Ejemplo de imputación sobre datos de temperatura exterior

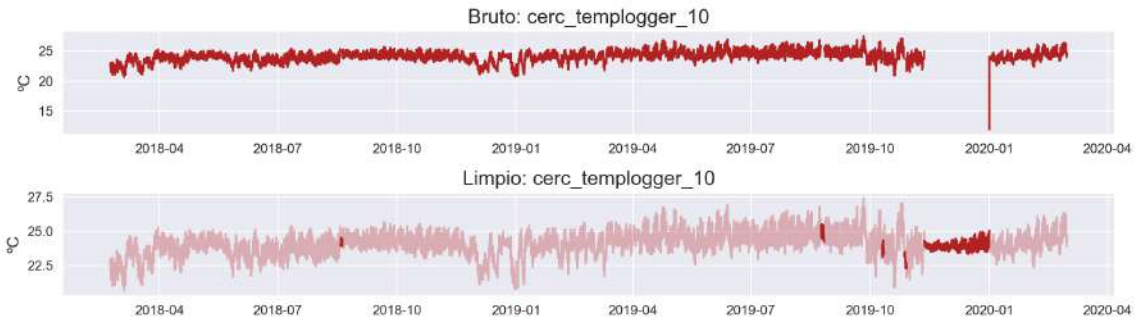

Figura 56: Ejemplo de imputación sobre datos de temperatura interior

Sin embargo, al considerar los datos de temperatura umbral, el resultado del algoritmo no es correcto. Esto es porque las variables de este conjunto de datos toman

valores discretos (Figura 57) y el algoritmo no es capaz de captar este tipo de distribución. Además, consideramos que este tipo de información puede añadir más ruido a las predicciones, pues no se puede extraer un patrón claro que contribuya al consumo de energía. Aunque se podría implementar una imputación adaptada a estos datos, al no ser éste el objetivo de este trabajo, se ha tomado la decisión de descartar los conjuntos de información sobre la temperatura umbral de calefacción y refrigeración.

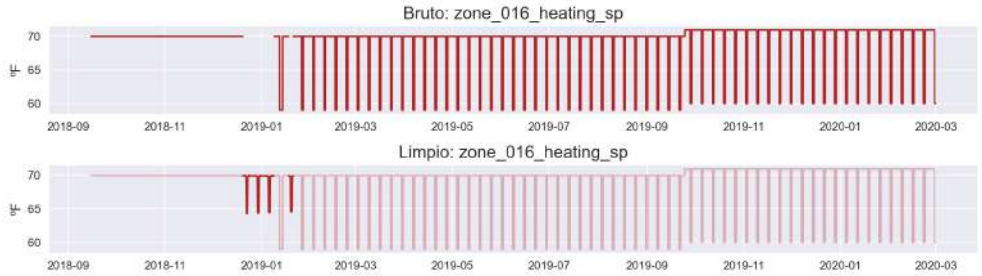

Figura 57: Ejemplo de imputación sobre datos de temperatura umbral de calefacción

10

DESCOMPOSICIÓN DE LA SERIE TEMPORAL

En este capítulo, se presentarán los resultados de las distintas descomposiciones de series temporales introducidas en capítulos anteriores.

Para ello, examinaremos detalladamente la estructura de cada una de las componentes, analizando sus características y patrones específicos. Además, se prestará especial atención al análisis de los residuos, que representan la variabilidad no explicada por la tendencia y las componentes estacionales. Se explorarán los patrones residuales y se evaluará la presencia de autocorrelación o cualquier otro tipo de dependencia temporal.

10.1 DESCOMPOSICIONES USUALES

En primer lugar, estudiaremos el resultado de analizar las descomposiciones aditiva y multiplicativa. La limitación principal de ambas es que no admiten múltiples componentes estacionales, lo cual supone un problema particularmente en nuestros casos, que ya sabemos que presentan una estacionalidad un poco más compleja.

Para ejemplificar el resultado obtenido escogeremos como periodo de componente estacional 96, es decir, un día completo. En realidad, se han analizado todas las componentes estacionales presentes, pero para evitar redundancias en esta sección únicamente se procederá con la frecuencia escogida.

El resultado obtenido de realizar la descomposición aditiva se encuentra expuesto en la Figura 58. Se puede ver que a simple vista parece que los residuos tienen cierta información que hacen que su forma no sea la de ruido blanco. Además, aunque la componente estacional no se vea de manera clara, si se analiza en un intervalo de tiempo más corto, se puede comprobar la forma sinusoidal característica de estas componentes.

El resultado del análisis de residuos se puede observar en la Figura 59. Se puede comprobar en el gráfico ACF que existe autocorrelación, y además, tras realizar el test de Ljung-Box obtenemos un p-valor equivalente a 0.0, lo que confirma la teoría de presencia de correlación.

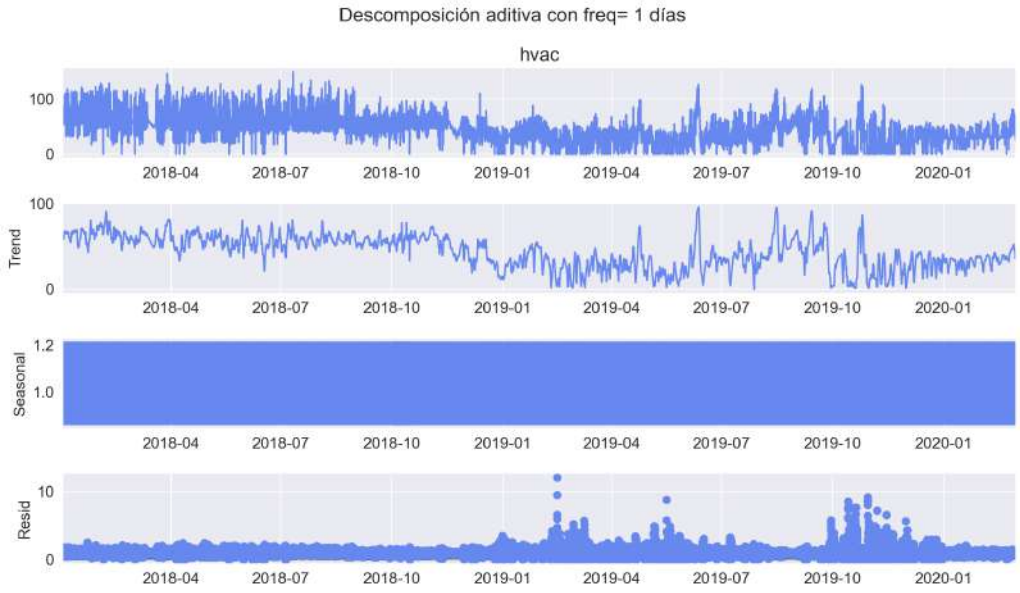

Figura 58: Descomposición aditiva

Se ha decidido, para reducir la extensión de esta sección, no incluir el detalle de la descomposición multiplicativa. En ella, además de observar presencia de autocorrelación, se ha observado que la distribución es muy distinta de la distribución normal, por lo que de nuevo descartamos la teoría de que los residuos sean un proceso estocástico de ruido blanco.

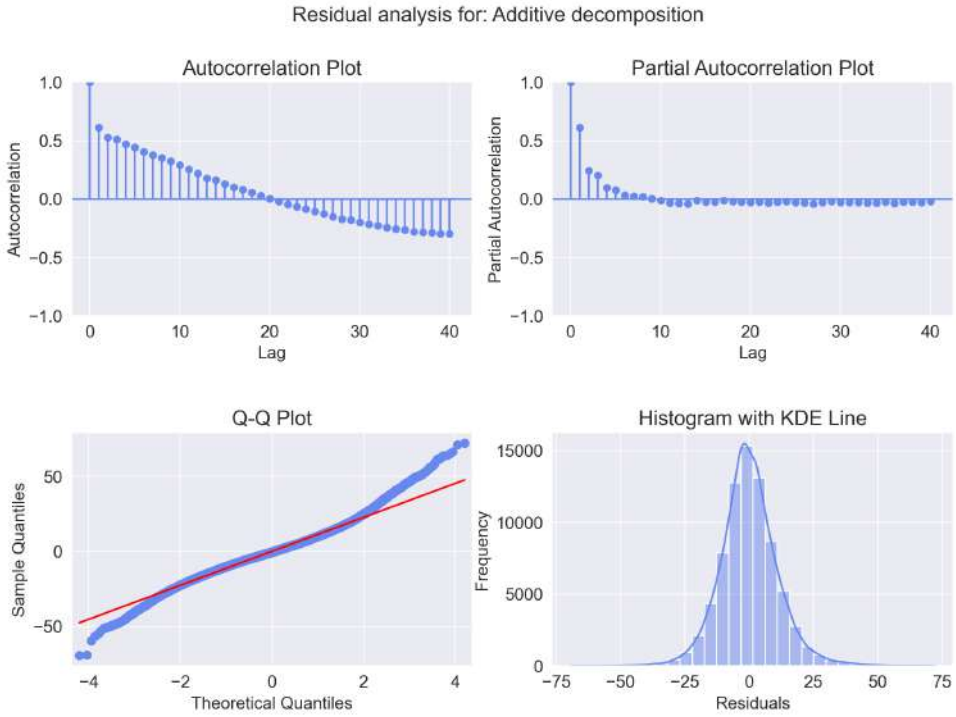

Figura 59: Estudio de residuos de la descomposición aditiva con frecuencia = 96

10.2 DESCOMPOSICIÓN ADITIVA AVANZADA

A continuación, presentamos el resultado de extraer todas las componentes estacionales, con periodos 4, 96, y 672, que representan una hora, día y semana, respectivamente. Para extraerlas, se obtendrá una descomposición aditiva de manera iterativa siguiendo el procedimiento descrito en la sección 7.1.2, y la componente residual será el residuo obtenido tras extraer la última componente estacional.

El detalle de esta descomposición se encuentra en la Figura 60, donde se ven con más claridad las componentes estacionales. En este caso la forma de la componente residual (última fila) se parece más al ruido blanco, sin embargo, tras realizar el análisis de residuos volvemos a concluir que existe autocorrelación presente en ellos, y por lo tanto, no podemos utilizarla para predecir los datos, pues estaríamos perdiendo información de la serie original.

10.3 DESCOMPOSICIÓN MSTL

Por último, vamos a estudiar los resultados de aplicar la descomposición MSTL (Descomposición estacional-tendencia múltiple utilizando Loess) introducida en capítulos anteriores.

Podemos estudiar el detalle de la misma en la Figura 62, donde destaca la falta de sinusoidalidad de las componentes estacionales. Esto es porque a diferencia de otros

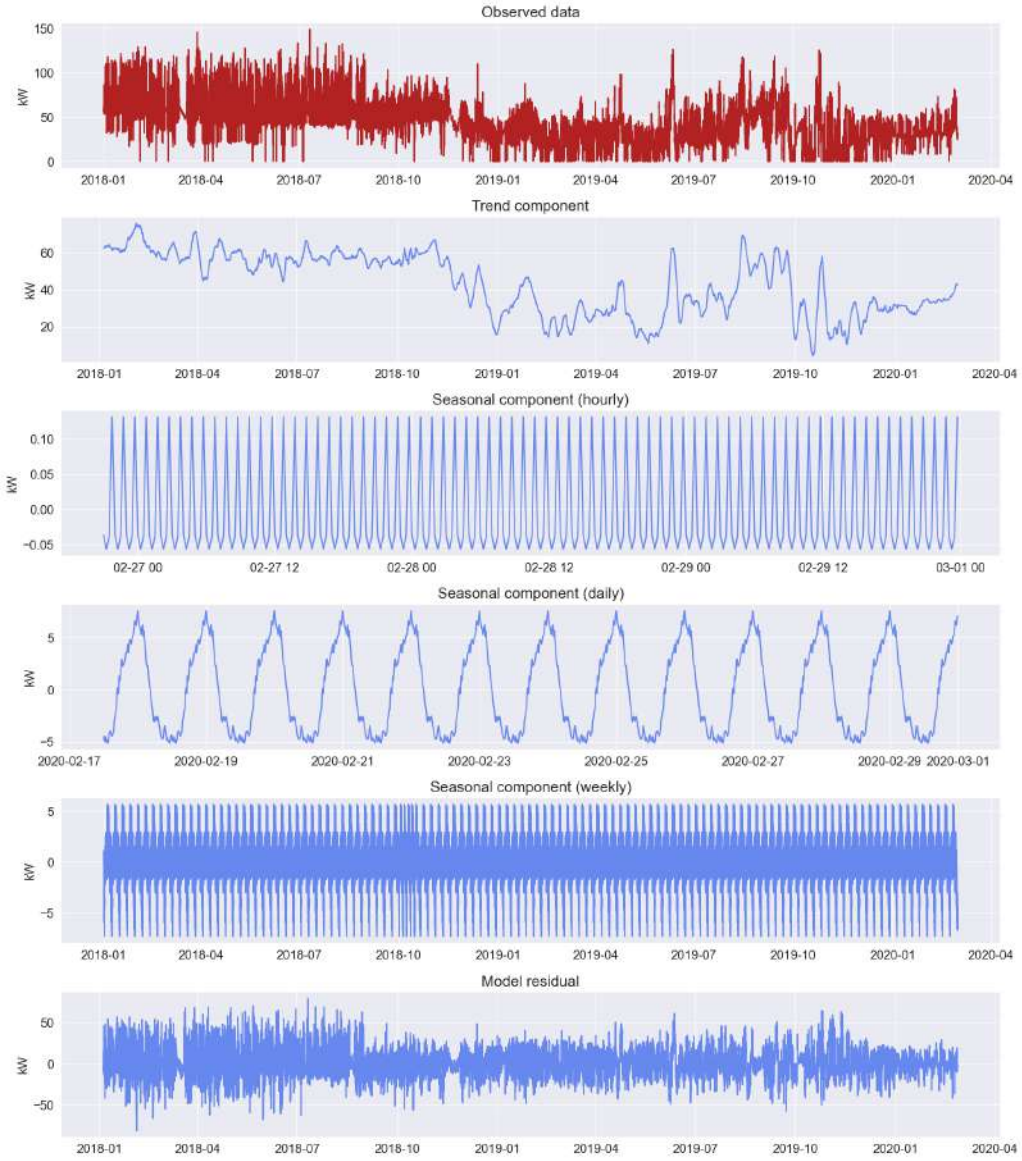

Figura 60: Estudio en detalle de la descomposición avanzada

métodos ya vistos donde se modelan las componentes estacionales mediante series de Fourier, MSTL utiliza técnicas la técnica Loess para capturar las variaciones estacionales.

El uso de Loess permite modelar componentes estacionales con patrones más complejos, pues se ajusta localmente a los datos y por tanto es capaz de capturar variaciones estacionales irregulares, como cambios de amplitud o frecuencia a lo largo del tiempo.

Para finalizar esta sección, podemos observar el análisis residual de esta descomposición (Figura 63). Es destacable el patrón observable en el gráfico PACF, que, junto con un p-value nulo en el test de Ljung-Box, nos permite afirmar la presencia de correlación periódica en los residuos.

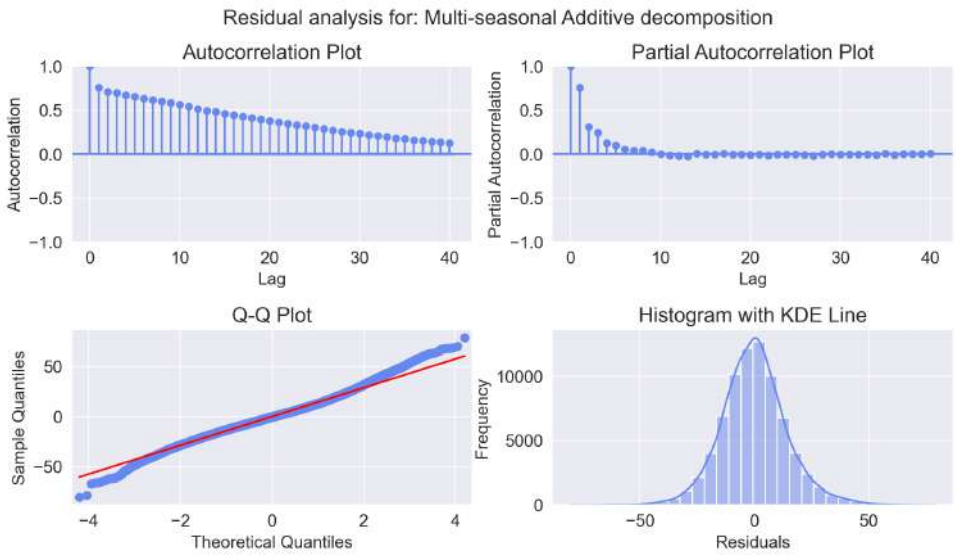

Figura 61: Estudio de los residuos de la descomposición avanzada

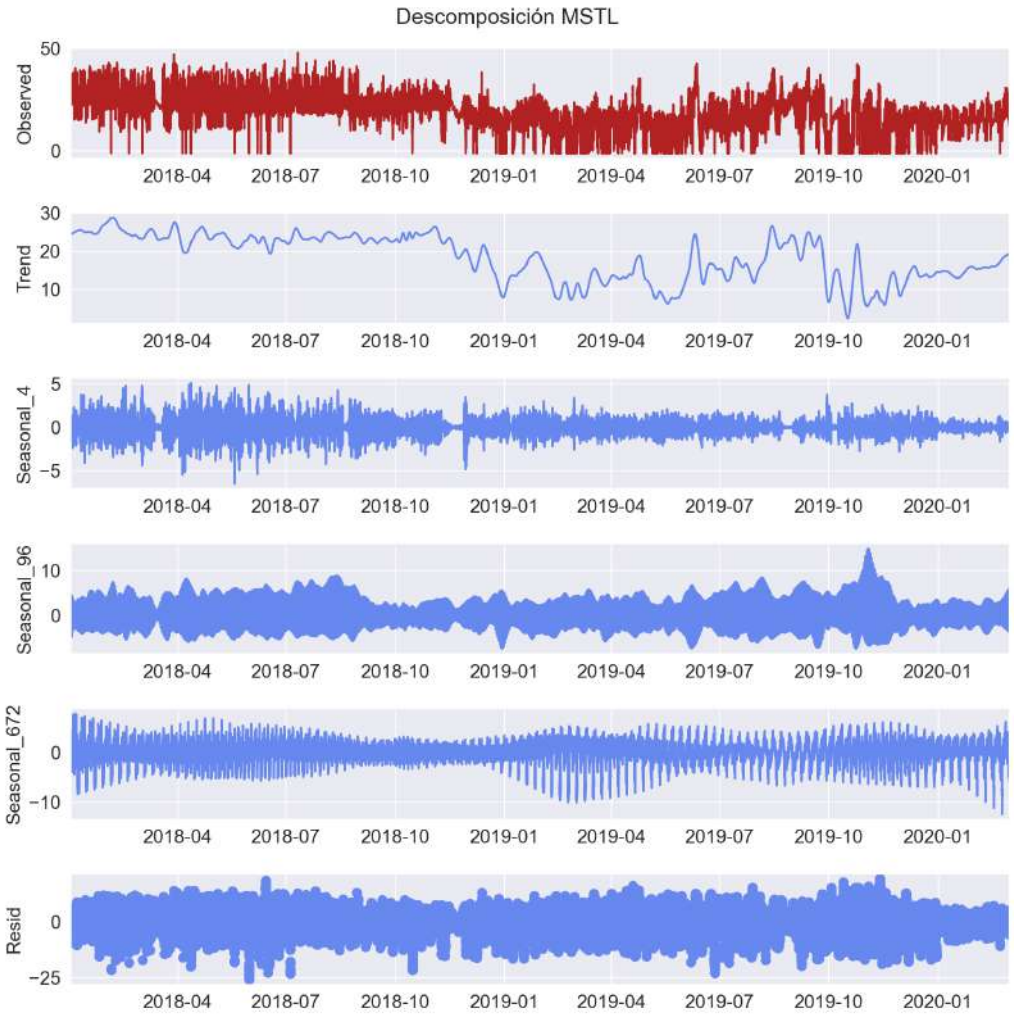

Figura 62: Estudio en detalle de la descomposición MSTL

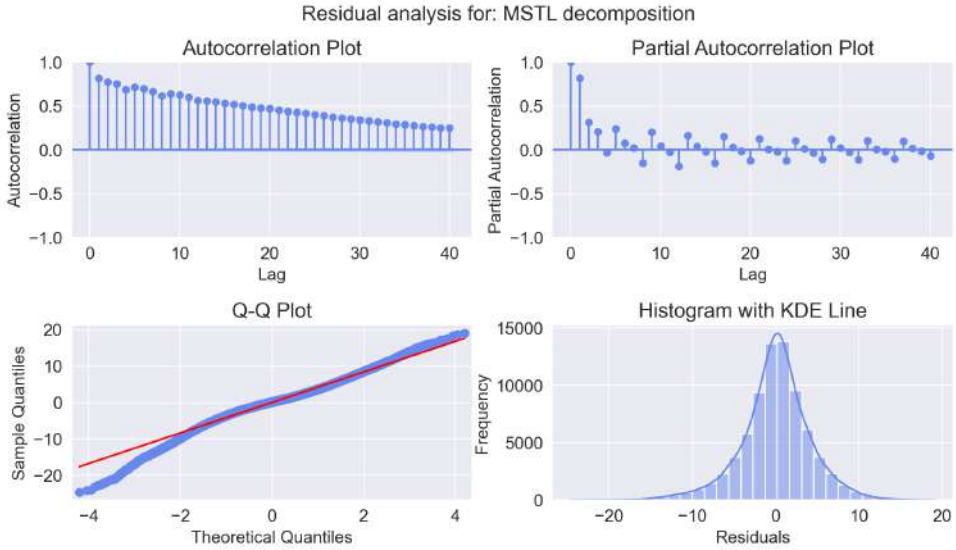

Figura 63: Estudio de los residuos en la descomposición MSTL

11

ALGORITMOS DE PREDICCIÓN

En este capítulo, presentaremos los resultados obtenidos al aplicar los algoritmos principales de predicción que hemos estudiado.

Comenzaremos por los algoritmos de Machine Learning de series temporales, como ARIMA, SARIMAX, Prophet, Unobserved Components y TBATS (explicados en el capítulo 8). Analizaremos el rendimiento de estos algoritmos en términos de métricas de evaluación, como el error cuadrático medio y el error porcentual absoluto medio. También exploraremos el efecto de ajustar los hiperparámetros de cada algoritmo y realizaremos comparaciones entre ellos. Por último mencionar que para los algoritmos en los que se pueda, probaremos a introducir variables exógenas en el entrenamiento, lo que nos permitirá incluir el efecto del resto de variables en las predicciones.

Posteriormente, nos adentraremos en el análisis de los resultados obtenidos mediante el uso de redes neuronales LSTM (*Long Short-Term Memory*, en inglés). Ya hemos estudiado que estas redes son especialmente adecuadas para el tratamiento de series temporales debido a su capacidad para capturar dependencias a largo plazo y su habilidad para modelar patrones secuenciales en los datos. Evaluaremos el desempeño de las redes LSTM en términos de precisión de predicción, analizando la influencia de diferentes arquitecturas de red, como el número de capas y unidades ocultas.

11.1 ALGORITMOS DE MACHINE LEARNING

Para analizar los resultados de los algoritmos de Machine Learning estudiados, vamos a incorporar un concepto que no habíamos visto hasta la fecha: la ventana móvil. La ventana móvil, también conocida como ventana deslizante, es una técnica que consiste en considerar en cada paso un subconjunto más pequeño de la serie, de longitud fija, denominado ventana. La ventana, que inicialmente consta de las primeras m observaciones, se desplaza a lo largo de la serie temporal con un paso determinado, lo que permite realizar predicciones en diferentes intervalos de tiempo.

La idea es que en la práctica, en el contexto en el que nos encontramos no es habitual realizar una predicción a varios meses vista, sino que se busca obtener una estimación del consumo de energía del mismo día, con algunas horas de antelación. En estos

casos además, a la hora de hacer predicciones, podemos contar con todos los datos recogidos hasta el momento de realizar una predicción, por lo que el utilizar una ventana móvil tiene sentido.

La elección del tamaño de la ventana y el paso de desplazamiento es crucial en el proceso de predicción de series temporales. Una ventana más pequeña puede capturar cambios rápidos y patrones locales, mientras que una ventana más grande puede capturar tendencias a largo plazo en la serie. Es importante encontrar un equilibrio entre la capacidad del modelo para capturar patrones detallados y la capacidad de generalización a largo plazo. En nuestro caso, vamos a utilizar siempre una ventana de 284 observaciones, es decir, utilizaremos la información de los últimos 4 días, para predecir las siguientes horas de información. También es necesario determinar la longitud de las predicciones, es decir, con cuánta antelación se quiere realizar predicciones. Para ello, compararemos los resultados al predecir 32, 64 y 96 observaciones, que representan 8h, 16h y 1 día, respectivamente.

Además, en los algoritmos que lo permitan, incluiremos información adicional al entrenamiento en forma de variables exógenas. Es necesario notar el paso previo de procesamiento que asegura que las variables introducidas como exógenas no están correladas entre sí, que ya se expuso en detalle en capítulos anteriores.

Consideraremos como conjunto de entrenamiento el 80 % de los datos, y como conjunto de prueba el 20 % restante.

11.1.1 *ARIMA y SARIMAX*

En primer lugar estudiaremos la alternativa más simple para modelar una serie temporal, y es mediante un proceso ARMA, sin estacionalidad. Los parámetros necesarios para este modelo (p, q, d) se han escogido mediante una búsqueda exhaustiva de todas las combinaciones entre 0 y 3 para cada parámetro, y comparando posteriormente la medida AIC y BIC.

Además, las variables se han escalado para poder garantizar la media nula. También resulta conveniente para poder comparar los modelos en los que se incorporan el resto de variables después de hacer PCA, que estarán escaladas.

La combinación escogida es $(p, d, q) = (2, 1, 2)$, y los resultados obtenidos al utilizar una ventana móvil en el conjunto se pueden estudiar en la Figura 64. Se observa como en el caso de no utilizar ventana móvil, los resultados no son válidos, pues no capturan ningún patrón relevante de la serie. Cuando realizamos predicciones en series con gran cantidad de datos sin utilizar técnicas como la ventana móvil, es común observar que las predicciones resultantes son una línea recta, que representa la convergencia del modelo hacia un valor de equilibrio a largo plazo, sin reflejar adecuadamente las fluctuaciones y las características temporales de los datos.

Estas predicciones planas y estáticas pueden resultar poco útiles, especialmente cuando se trata de capturar cambios estacionales o tendencias a corto plazo, como es nues-

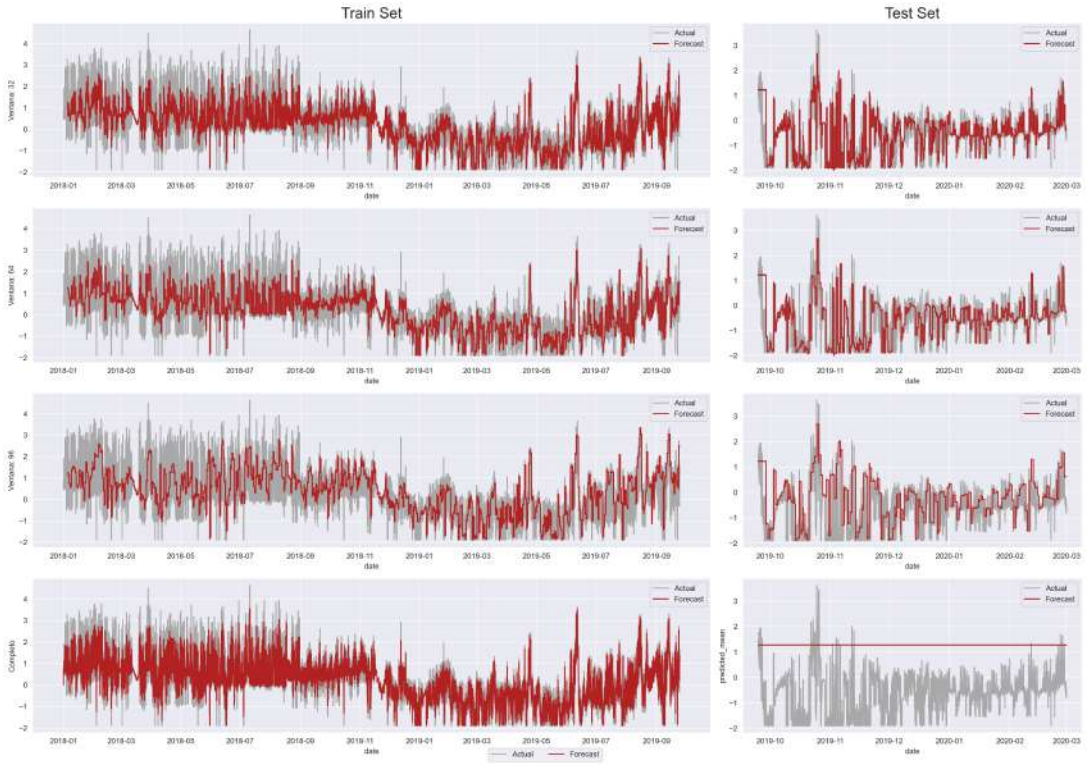

Figura 64: Resultados obtenidos al hacer predicciones con un modelo ARIMA(2,1,2) y ventana móvil de 32, 64 y 96 observaciones, y sin el uso de ventana móvil.

tro caso. Además, a medida que se incrementa la antelación con la que se predicen los datos, el modelo reduce el detalle con el que predice la información.

Podemos proceder a realizar la misma prueba, pero esta vez con un proceso SARIMAX, que se ha ajustado mediante una búsqueda por fuerza bruta utilizando el criterio AIC. En este caso, para la parte del proceso ARMA se ha obtenido un orden de $(3,1,1)$, y para la parte estacional hemos obtenido $(2,0,1,4)$. Tal y como vimos anteriormente, la limitación principal de este modelo es su incapacidad para modelar múltiples componentes estacionales, y en este caso vamos a proceder asumiendo únicamente una estacionalidad de periodo 4, es decir, horaria. Más adelante introduciremos el resto de componentes mediante series de Fourier.

Podemos observar (Figura 65) como de nuevo, en el caso de que no se use la ventana móvil, las predicciones representan la situación de equilibrio a largo plazo, y por tanto no son útiles para predecir con relativamente poca antelación.

A continuación repetimos el experimento con el mismo modelo SARIMAX, pero esta vez introduciendo series de Fourier (Figura 66) para incluir las componentes estacionales que no habíamos incluido en el modelo anterior. Las tres componentes estacionales que habíamos encontrado son las que tienen un periodo de hora, día y semana. Sin embargo, el entrenamiento del modelo se ralentizaba demasiado al incluir las dos componentes, y el impacto en la precisión de las predicciones no era proporcional, por lo que se decidió incluir únicamente una serie de Fourier que represente la esta-

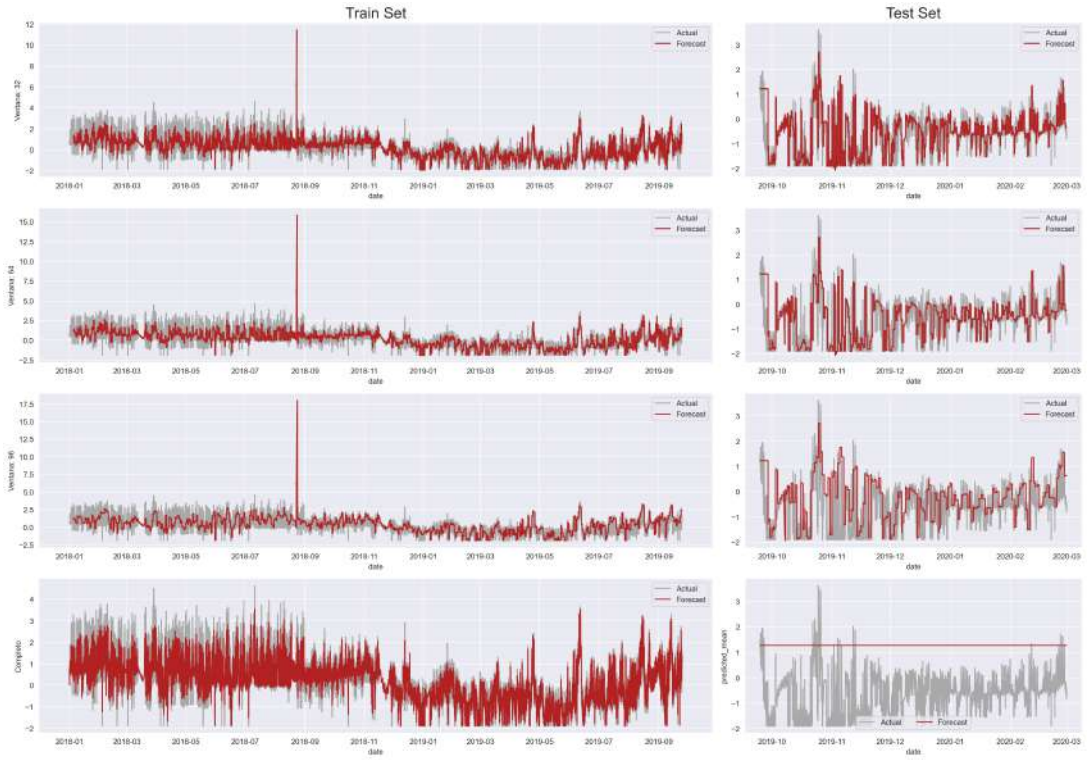

Figura 65: Resultados obtenidos al hacer predicciones con un modelo SARIMAX(3,1,1)x(2,0,1,4) y ventana móvil de 32, 64 y 96 observaciones, y sin el uso de ventana móvil.

cionalidad diaria. Para hacerlo, transformamos cada fila del índice de la serie original (las fechas) en dos variables distintas, que no son más que el seno y el coseno de la serie de Fourier.

Los resultados que obtenemos vuelven a ser muy parecidos a los resultados ya analizados, en este caso se puede observar que en el modelo que no utiliza ventana móvil, las predicciones convergen de una manera un poco más lenta. Esto tiene sentido pues al añadir las variables exógenas estamos incrementando la complejidad del modelo.

Por último, vamos a probar a introducir la información presente en el resto de variables, y analizaremos el impacto de cada una de ellas. Cabe notar que estamos suponiendo que el resto de variables están disponibles para el periodo que queremos predecir.

Cabe notar que este último experimento es con diferencia el que más tiempo tardó en completarse, por lo que debería ser un aspecto a considerar a la hora de escoger el algoritmo más adecuado. Si analizamos los resultados (Figura 67) podemos comprobar que no se pierde tanta precisión al ampliar el número de predicciones como en los modelos anteriores. Más adelante compararemos las métricas obtenidas en cada uno de ellos.

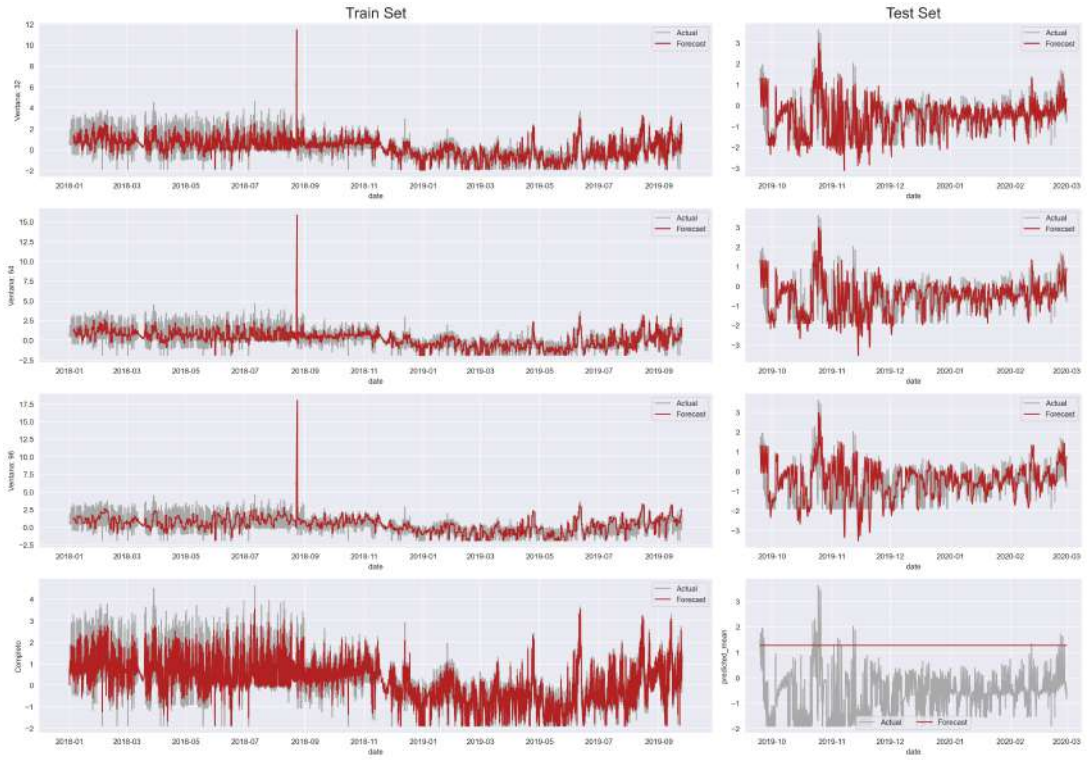

Figura 66: Resultados obtenidos al hacer predicciones con un modelo SARIMAX(3,1,1)x(2,0,1,4) con series de Fourier y ventana móvil de 32, 64 y 96 observaciones, y sin el uso de ventana móvil.

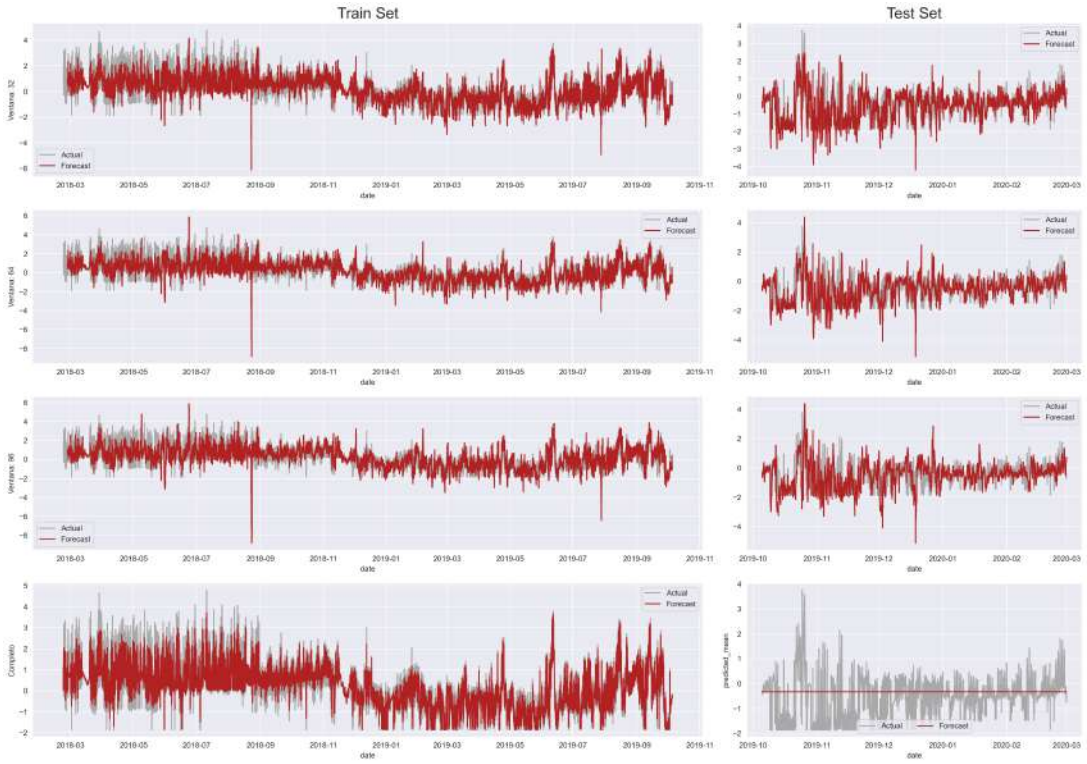

Figura 67: Resultados obtenidos al hacer predicciones con un modelo SARIMAX(3,1,1)x(2,0,1,4) multivariable y ventana móvil de 32, 64 y 96 observaciones, y sin el uso de ventana móvil.

11.1.2 *Prophet*

El siguiente algoritmo para el cual vamos a analizar los resultados es Prophet. Tal y como vimos anteriormente, es un modelo adecuado para poder representar la estacionalidad presente en la serie, aunque es necesario solventar el problema de la varianza.

Para ello, vamos a comparar el rendimiento del algoritmo con los datos originales y con los datos transformados mediante Box-Cox y la transformación logarítmica. Aunque Prophet no incorpora una gran cantidad de parámetros, se pueden personalizar ciertos aspectos del modelo, y para ello se probaron distintas combinaciones de parámetros, y se escogió la que proporcionaba un error porcentual absoluto medio (MAPE) menor.

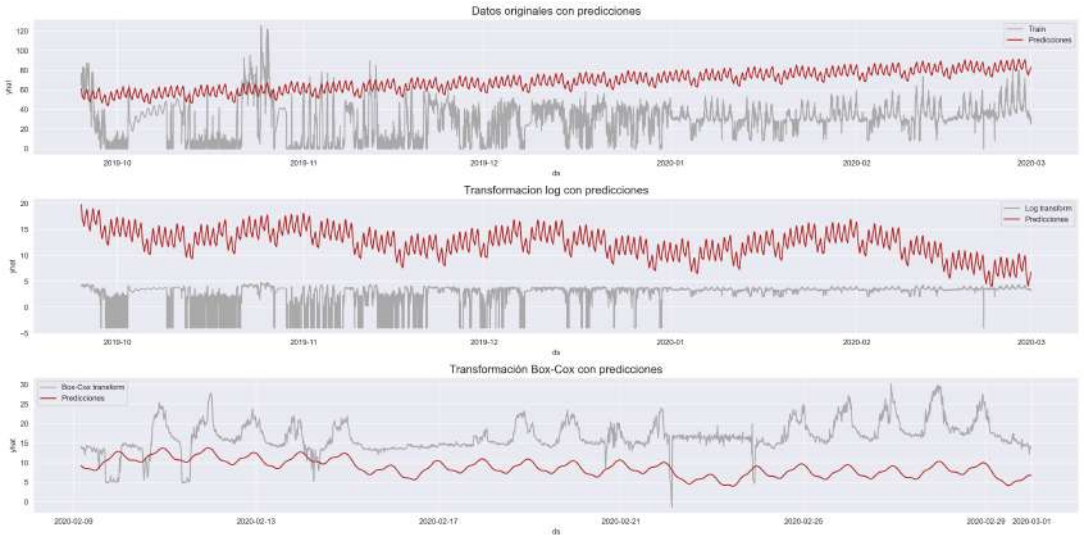

Figura 68: Resultados obtenidos al hacer predicciones con un modelo Prophet sobre la serie original, y la serie transformada mediante el uso de Box-Cox y el logaritmo.

En la Figura 68 podemos comprobar que realmente, ninguna de las tres alternativas resulta adecuada. Esto puede ser por diversos motivos, como por ejemplo que al entrenar sobre todos los datos de entrenamiento estemos incluyendo demasiado ruido, y que sea mejor utilizar la alternativa de ventana móvil.

Vamos a comprobar la precisión de las predicciones al utilizar los datos escalados originales y distintas longitudes de ventana móvil (Figura 69). Se pueden realizar dos observaciones: la primera, que las predicciones son menos exactas en general, y la segunda, que hay menos diferencia en la calidad de las predicciones entre las distintas longitudes de ventana móvil. Esto nos lleva a pensar que Prophet no es sensible a la cantidad de información que tiene que predecir mientras no se esté prediciendo a largo plazo.

En la Figura 70 podemos estudiar las predicciones al incluir el resto de variables. Destaca en este caso negativamente la influencia de las variables exógenas, que cla-

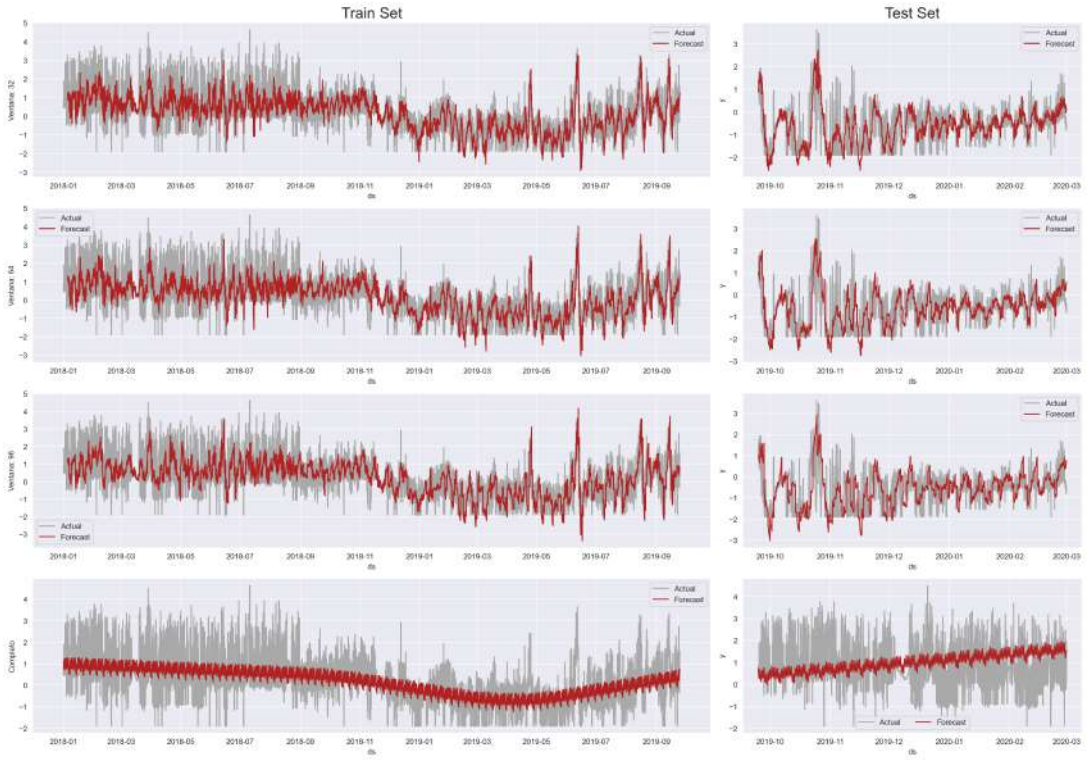

Figura 69: Resultados obtenidos al hacer predicciones con un modelo Prophet con ventana móvil de 32, 64 y 96 observaciones, y sin el uso de ventana móvil.

ramente empeoran la calidad de las predicciones. Por este motivo, descartamos el incluir el resto de variables al utilizar este algoritmo.

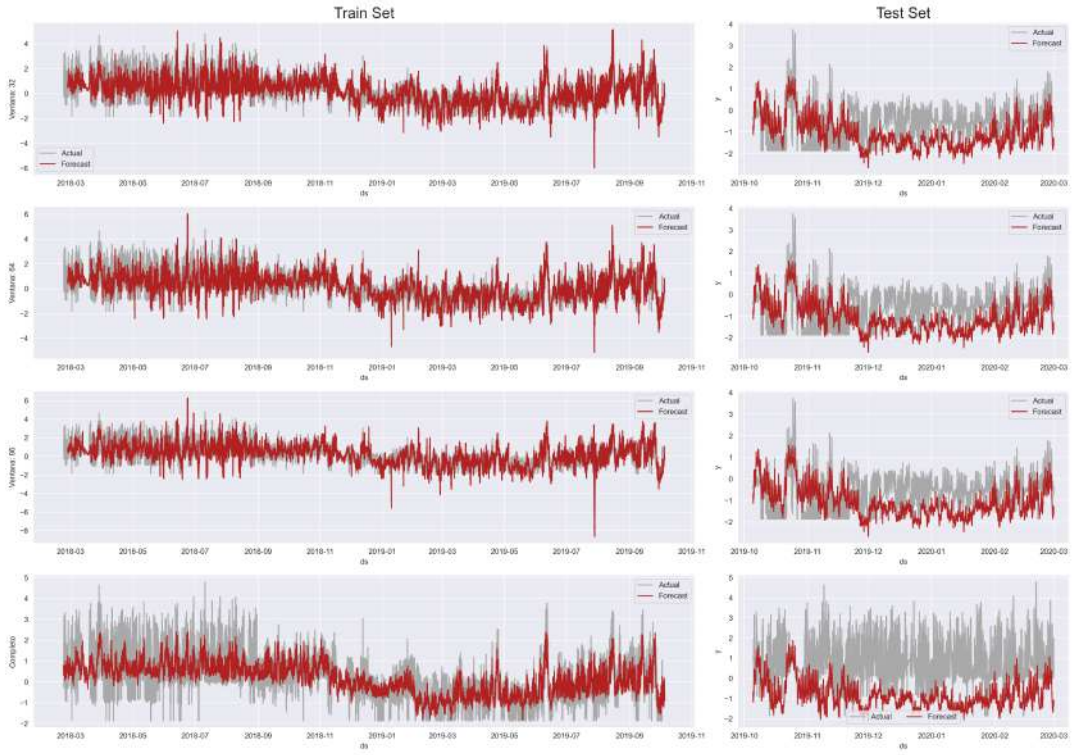

Figura 70: Resultados obtenidos al hacer predicciones con un modelo Prophet multivariable con ventana móvil de 32, 64 y 96 observaciones, y sin el uso de ventana móvil.

11.1.3 *Unobserved Components*

El siguiente algoritmo que vamos a considerar es Unobserved Components. Vimos que este modelo realizaba una descomposición con las componentes usuales, pero cuyo modelado era más complejo que en las descomposiciones aditiva y multiplicativa.

Para este algoritmo se repitió el mismo experimento, pero duplicando el tamaño de la ventana de entrada: en lugar de considerar 4 días de información, se utilizó una ventana de 8 días para cada predicción. Esta modificación se basa en los resultados obtenidos al realizar el experimento con la ventana de tamaño anterior. En ese caso, las predicciones no fueron satisfactorias en ninguno de los tres escenarios evaluados, siendo ligeramente más precisas cuando no se utilizó una ventana móvil. A raíz de estos resultados, se infiere que el algoritmo puede beneficiarse de una mayor cantidad de información para lograr predicciones más precisas, lo cual es coherente con su naturaleza compleja.

Al observar los resultados (Figura 71), podemos comprobar que se trata de un modelo que captura correctamente las fluctuaciones de la serie, pero que presenta valores en ocasiones demasiado extremos. Sería interesante comprobar si estos valores extremos se corresponden con una subida o bajada demasiado abrupta en los datos, pues en ese caso estas predicciones sí que serían útiles para nuestro caso de uso.

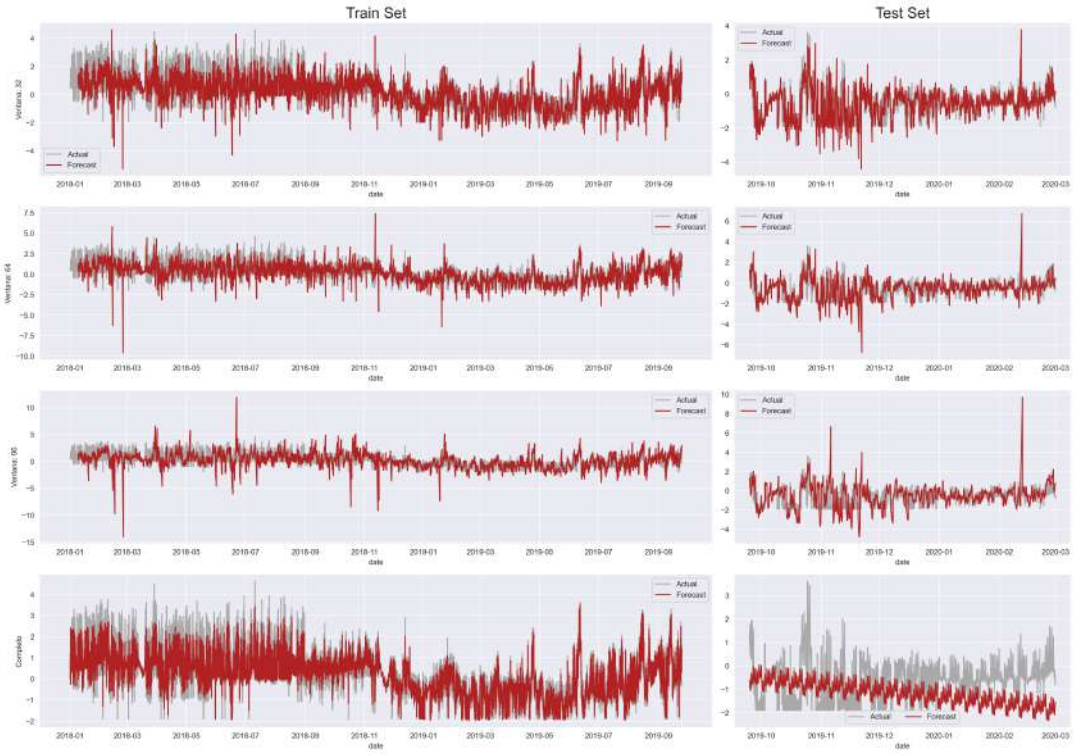

Figura 71: Resultados obtenidos al hacer predicciones con un modelo Unobserved Components con ventana móvil de 32, 64 y 96 observaciones, y sin el uso de ventana móvil.

De la misma manera, repetimos el procedimiento anterior, incluyendo el resto de variables y manteniendo la ventana móvil duplicada (Figura 72). Podemos comprobar la diferencia en las precisiones a largo plazo, que aunque siguen sin ser adecuadas, se ajustan mejor que en el modelo univariable, y presentan mejores resultados en las métricas consideradas. Además, se puede observar que persiste el problema de los valores extremos al predecir con 12h y un día de antelación, pero que en el caso de predecir con 8h de antelación este problema se ha solventado.

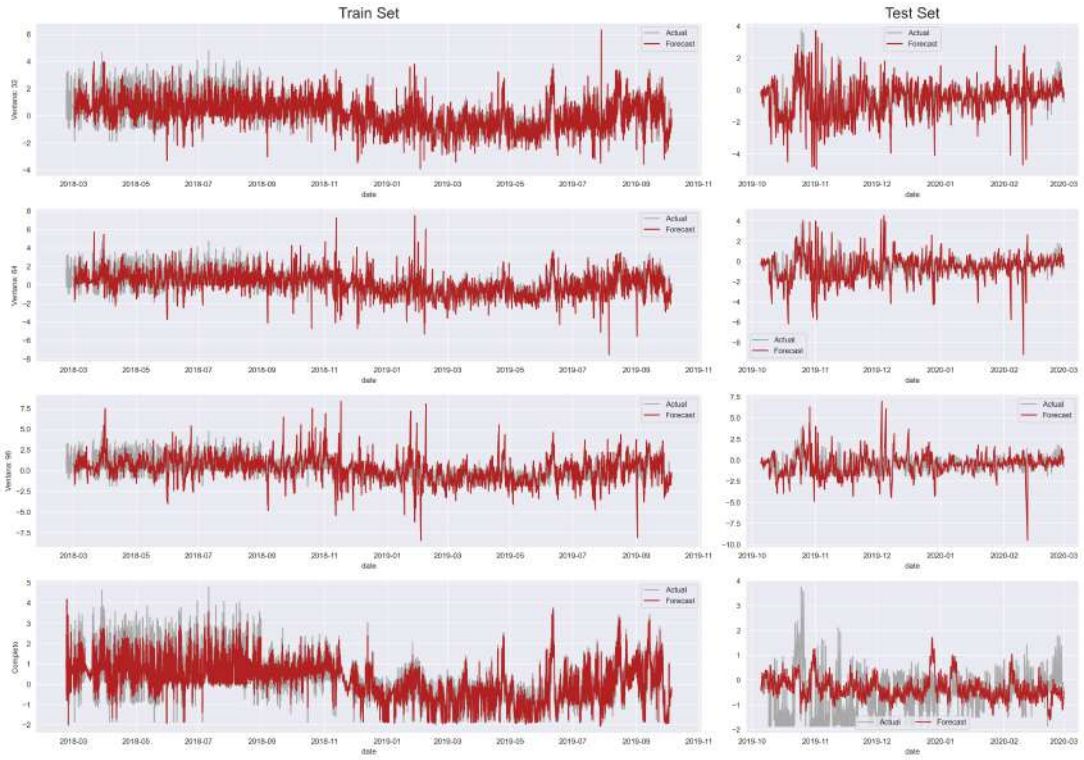

Figura 72: Resultados obtenidos al hacer predicciones con un modelo Unobserved Components multivariable con ventana móvil de 32, 64 y 96 observaciones, y sin el uso de ventana móvil.

11.1.4 *TBATS*

El último algoritmo de los que hemos estudiado es TBATS. Antes de realizar el experimento completo, se intentó hacer una predicción única de dos días con la ventana de predicción usual de cuatro días de información, y los resultados no fueron satisfactorios. Por este motivo, se repitió la prueba con dos tamaños de ventana superiores (Figura 73), observando una mejora considerable al utilizar tamaños más grandes. Además, es necesario destacar que por el funcionamiento del ajuste del algoritmo, es uno de los que más tiempo tarda, y es un factor a tener en cuenta a la hora de escoger el tamaño de ventana. Es por estas razones por las que se escogió como tamaño de ventana el tamaño duplicado, de ocho días de información.

Con el fin de llevar a cabo el experimento con este modelo, se ha optado por reducir la cantidad de predicciones realizadas en el conjunto de entrenamiento (Figura 74). Sin embargo, podemos concluir que este enfoque no ha resultado adecuado para nuestro caso concreto. Tanto el tiempo requerido para reentrenar el modelo como la calidad de las predicciones obtenidas nos indican que esta elección no es la más apropiada para nuestro conjunto de datos y objetivo de predicción. Además, este algoritmo no pemite la inclusión de variables exógenas para mejorar las predicciones. Por lo tanto, es necesario explorar alternativas más adecuadas que nos permitan obtener

Figura 73: Comparación de tamaño de ventana para hacer predicciones con dos días de antelación

resultados más satisfactorios y eficientes en términos de tiempo de entrenamiento y calidad de predicción.

Dado que los modelos anteriores no han alcanzado los resultados deseados, en la siguiente sección exploraremos una alternativa prometedora: las redes neuronales LSTM. Analizaremos el rendimiento de las redes LSTM realizando predicciones con 8 horas de antelación, y al final del capítulo, compararemos las métricas obtenidas por cada algoritmo para evaluar su desempeño y determinar la mejor opción para nuestro caso específico.

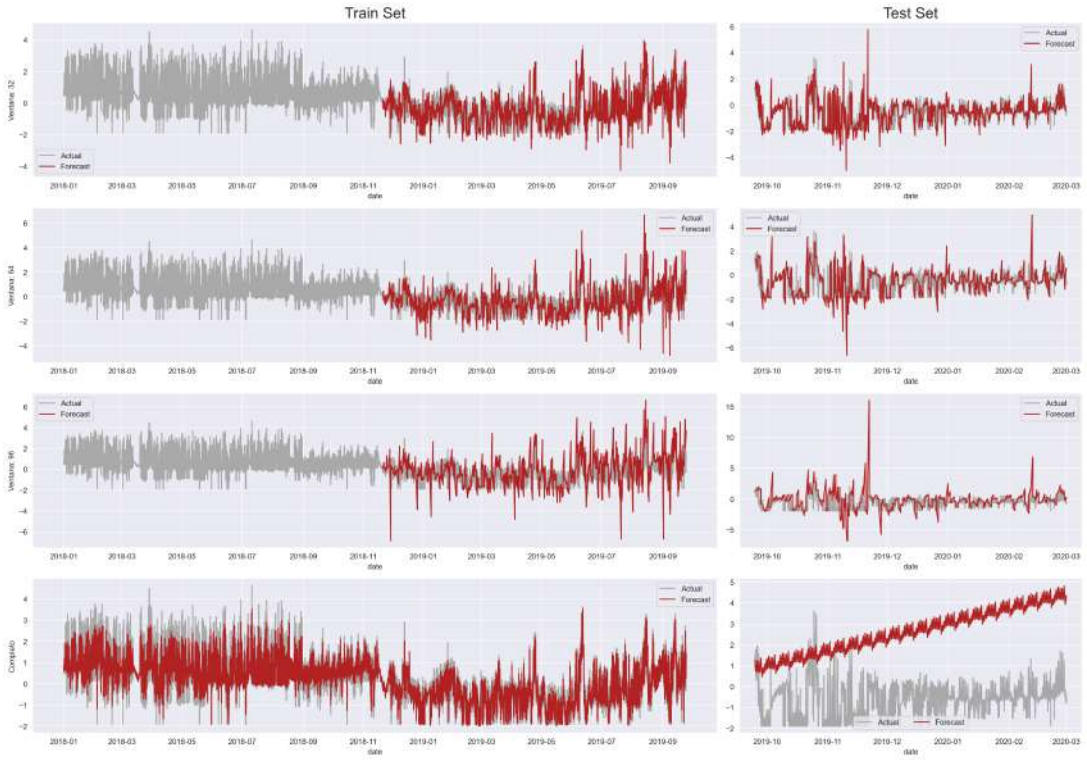

Figura 74: Resultados obtenidos al hacer predicciones con un modelo TBATS con ventana móvil de 32, 64 y 96 observaciones, y sin el uso de ventana móvil.

11.2 LSTM

En esta sección analizaremos los resultados de los experimentos con redes neuronales LSTM (*Long Short-Term Memory*). Nos centraremos exclusivamente en el tamaño de ventana más pequeño, que realiza 32 predicciones. El objetivo es captar los patrones inmediatos de la serie, como subidas o bajadas repentinas, y hemos observado que esto se logra de manera más efectiva con ventanas de tamaño reducido. Además, se ha considerado que 8h es suficiente tiempo para tomar decisiones sobre la administración del aire acondicionado en el edificio en función de los resultados obtenidos.

Figura 75: RMSE según tiempo de predicción

En la Figura 75 podemos comprobar el crecimiento del RMSE (*Root Mean Squared Error*) dentro de una misma predicción con uno de los modelos que veremos a continuación. Podemos concluir, por tanto, que cuanto más largas sean las predicciones, más imprecisas serán.

Dividiremos nuestro conjunto de datos en conjuntos de entrenamiento, validación y prueba. El conjunto de entrenamiento contendrá el 70 % de las observaciones, mientras que el conjunto de validación se compondrá del 20 % y el conjunto de prueba del 10 % restante.

Es importante recordar que el conjunto de entrenamiento se utiliza para actualizar los pesos del modelo, mientras que el conjunto de validación se utiliza para evaluar la mejora en el rendimiento a medida que se actualizan los parámetros. Además, a lo largo de todos los experimentos, hemos mantenido un número fijo de lotes, siendo este valor de 20.

Para cada configuración se han analizado tanto los resultados cuantitativos, a través de métricas de evaluación como por ejemplo el error cuadrático medio (MSE), o el error porcentual absoluto medio (MAPE), como los resultados cualitativos, mediante la comparación visual de las predicciones generadas por el modelo con los valores reales del conjunto de prueba.

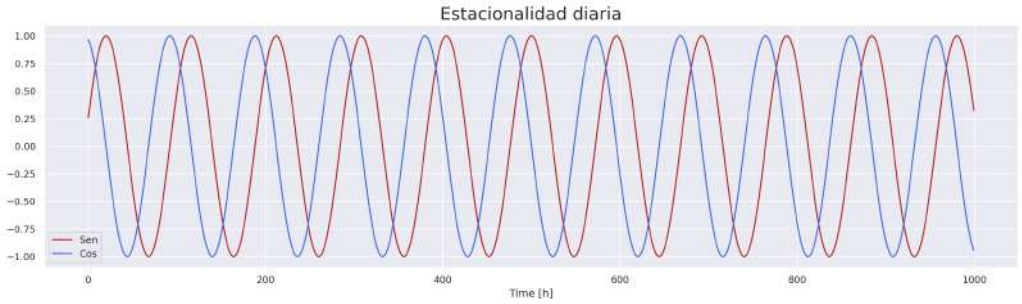

Figura 76: Modelado de la estacionalidad diaria a partir de Series de Fourier

Además, se han añadido como variables exógenas los términos seno y coseno de la serie de Fourier correspondiente a la estacionalidad horaria, diaria y semana, en la Figura 76 podemos comprobar el modelado de la estacionalidad diaria.

11.2.1 *Configuración trivial*

La primera configuración de red que vamos a analizar, que no es realmente una red neuronal como tal, es la que únicamente predice la siguiente observación, cuyo valor será igual al último valor que ha recibido como entrada. Este modelo es el modelo más trivial que podemos considerar, y se va a mantener como referencia para evaluar la bondad de las configuraciones siguientes.

Es necesario notar que aunque aparentemente este modelo pueda parecer idóneo (Figura 77) , su utilidad está muy limitada, pues únicamente puede predecir la siguiente

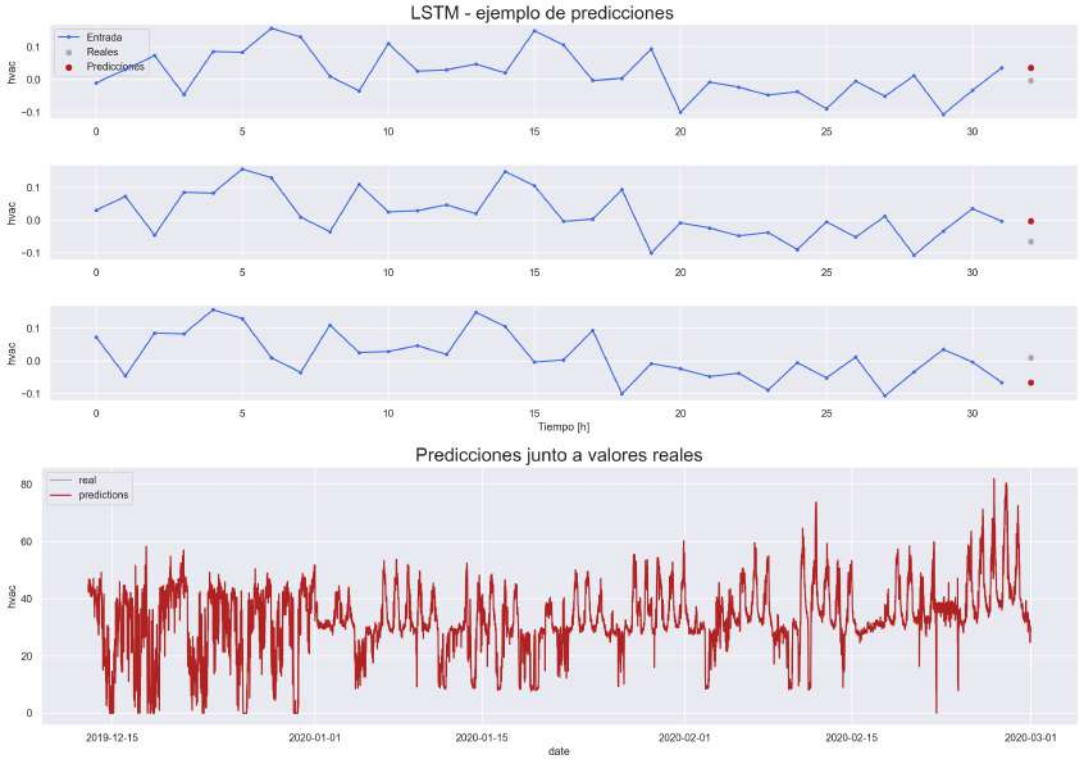

Figura 77: Predicciones obtenidas según el modelo trivial

observación. Tal y como veíamos en los modelos anteriores, en nuestro caso de uso resulta más interesante predecir con cierta antelación, por lo que esta configuración no resultaría rentable.

11.2.2 *Experimentos LSTM*

Hemos decidido utilizar como entrada 192 observaciones, lo que equivale a dos días de información. Esta elección se basa en los resultados obtenidos al probar con el tamaño de ventana anterior de 384 observaciones.

En esos experimentos previos, observamos que al utilizar una ventana más grande, se producía un rendimiento inferior, específicamente un aumento consistente de aproximadamente 0.1 puntos en el Error Cuadrático Medio (MSE). Por lo tanto, para mejorar la precisión y la calidad de las predicciones, hemos decidido reducir el tamaño de la ventana de entrada a 192 observaciones.

Arquitectura simple

La primera arquitectura consiste de una sola capa LSTM con 120 unidades, al que se le ha añadido una capa de Dropout, cuyo objetivo es regularizar la red al desactivar aleatoriamente algunas de sus unidades durante el entrenamiento, reduciendo el sobreajuste.

Al definir la capa LSTM se han utilizado los parámetros de inicialización y regularización del kernel. La inicialización de kernel se refiere al proceso de establecer los valores iniciales de los pesos del bloque y definir correctamente este parámetro ayuda a evitar problemas como el estancamiento del entrenamiento.

Por otro lado, la regularización de kernel se utiliza para controlar la complejidad del modelo y prevenir el sobreajuste. Al aplicar una regularización al kernel, se penaliza el crecimiento excesivo de los pesos, lo que puede mejorar la generalización del modelo y reducir el riesgo de *overfitting*.

Después de experimentar con distintos valores para estos parámetros en esta arquitectura, se llegó a la conclusión de que la opción más correcta era inicializar los pesos mediante una distribución normal centrada en 0 y con desviación típica igual al número de entradas. Para la regularización, se utilizará una combinación de los métodos L1 y L2, ya introducidos anteriormente.

Arquitectura 1 Arquitectura del modelo LSTM simple

```
Model: "LSTM_32_modelo1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_47 (LSTM)              (None, 120)               60480

 dropout_31 (Dropout)        (None, 120)               0

 dense_27 (Dense)            (None, 32)                3872

 reshape_27 (Reshape)        (None, 32, 1)             0

=================================================================
Total params: 64,352
...
_________________________________________________________________
```

Las predicciones obtenidas (Figura 78) muestran un rendimiento aceptable en términos generales. Sin embargo, se observa una tendencia de las predicciones a ser relativamente planas, lo que indica una limitada capacidad para capturar variaciones y picos dentro de la serie temporal. Esta falta de capacidad para detectar patrones más complejos sugiere que los datos subyacentes presentan una mayor complejidad, y la arquitectura actual no es capaz de capturar completamente estas características.

El gráfico de la función de pérdida (*loss*) en los conjuntos de entrenamiento y validación (Figura 79) proporciona una visualización de la evolución del rendimiento del modelo a lo largo de la fase de entrenamiento. La función de pérdida representa la diferencia entre las predicciones del modelo y los valores reales de los datos, y se espera que a medida que se vayan ajustando los parámetros, se vea reducida.

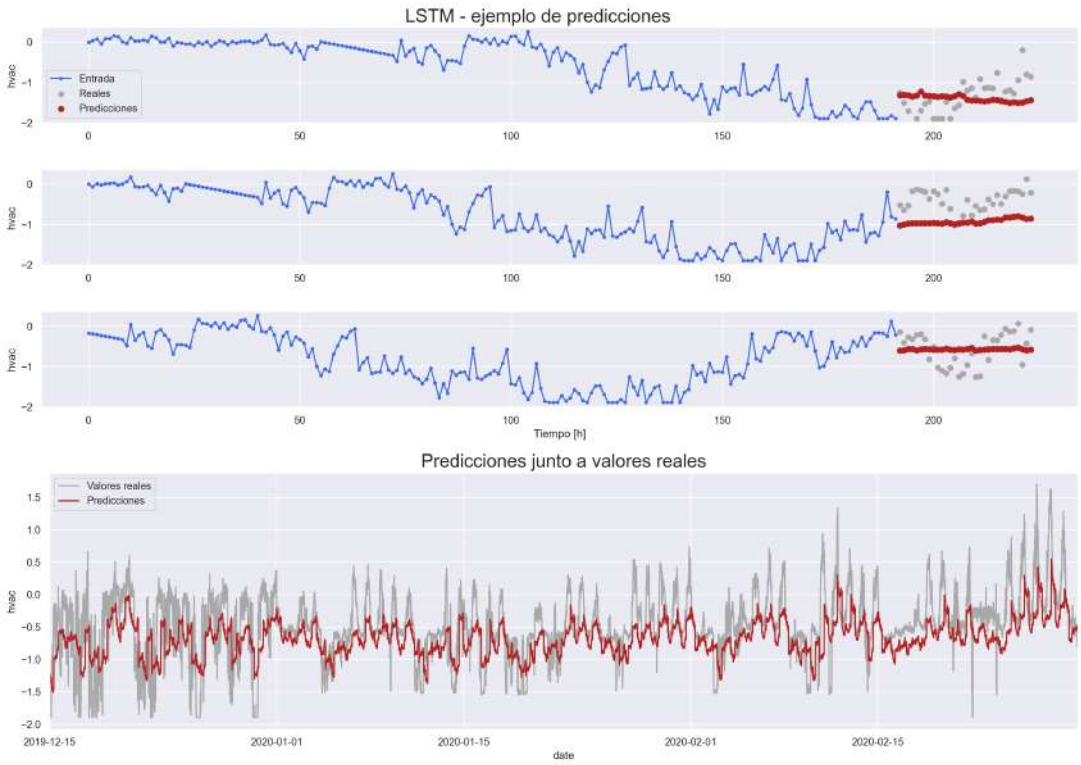

Figura 78: Predicciones obtenidas con el modelo LSTM_32_modelo1

En el gráfico, el eje horizontal representa las iteraciones o épocas del entrenamiento, mientras que el eje vertical representa el valor de la función de pérdida. El seguimiento de la función de pérdida en ambos conjuntos nos permite evaluar la capacidad del modelo para aprender de los datos, e idealmente, esperamos ver una disminución continua de la función de pérdida en el conjunto de entrenamiento y una convergencia similar en el conjunto de validación. De no ser así, podríamos determinar que hay un problema de *overfitting*.

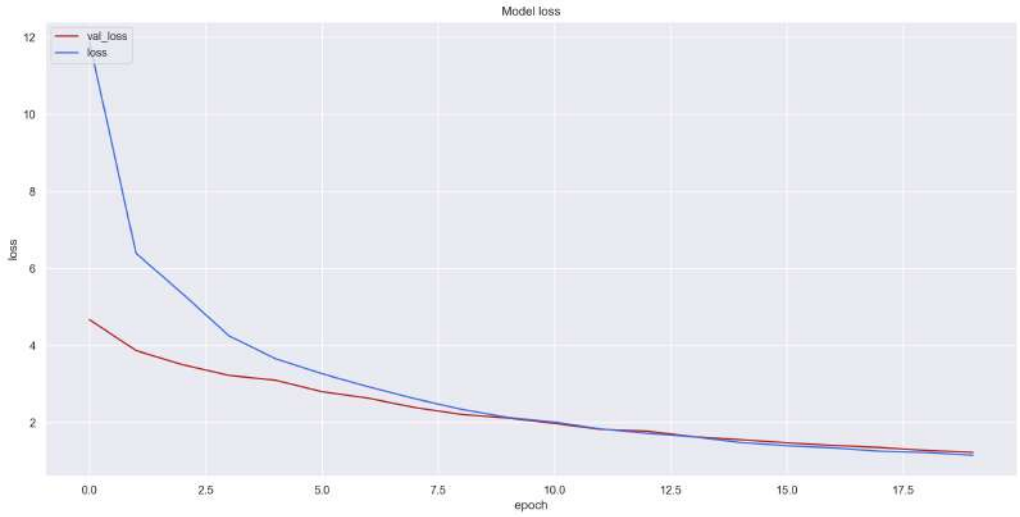

Figura 79: Función de pérdida de LSTM_32_modelo1

Arquitectura compleja

La segunda arquitectura con la que se experimentó es más compleja, pues divide la red en dos bloques: el primero trabaja de manera idéntica a la red que acabamos de estudiar, y el segundo recibe como entrada la salida del primero, y repite el proceso, generando la salida definitiva.

El resumen de la red se puede estudiar a continuación:

Si observamos las predicciones obtenidas (Figura 80) podemos concluir que aunque esta arquitectura no sea capaz de predecir la forma sinusoidal que caracteriza a los datos, sí que es capaz de predecir correctamente la magnitud de las subidas y bajadas. Esto es especialmente relevante en el contexto de las predicciones de consumo energético, donde el enfoque principal se centra en estimar con precisión los niveles de consumo en lugar de replicar la forma exacta de la serie temporal.

Además, si observamos la función de pérdida (Figura 81) podemos comprobar que en este caso no es necesario añadir una capa de Dropout, pues con los regularizadores del kernel es suficiente.

Es interesante notar una diferencia con los algoritmos de Machine Learning que hemos analizado, y es que durante la obtención de predicciones en el conjunto de prueba, no se reentrena la red.

Si nos centramos en el contexto en el que se quiere implementar este modelo, resultaría práctico reentrenar la red con los datos que tengamos disponibles hasta el momento de hacer la predicción. Por eso mismo, vamos a repetir la obtención de predicciones siguiendo este método (Figura 82).

Arquitectura 2 Arquitectura del modelo LSTM complejo

```
 Model: "LSTM_32_modelo_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_188 (LSTM)             (None, 150)               93600

 dense_112 (Dense)           (None, 80)                12080

 activation_27 (Activation)  (None, 80)                0

 repeat_vector_20 (RepeatVec  (None, 80, 80)           0
 tor)

 lstm_189 (LSTM)             (None, 150)               138600

 dense_113 (Dense)           (None, 32)                4832

 activation_28 (Activation)  (None, 32)                0

 reshape_91 (Reshape)        (None, 32, 1)             0

=================================================================
Total params: 249,112
Trainable params: 249,112
Non-trainable params: 0
_________________________________________________________________
```

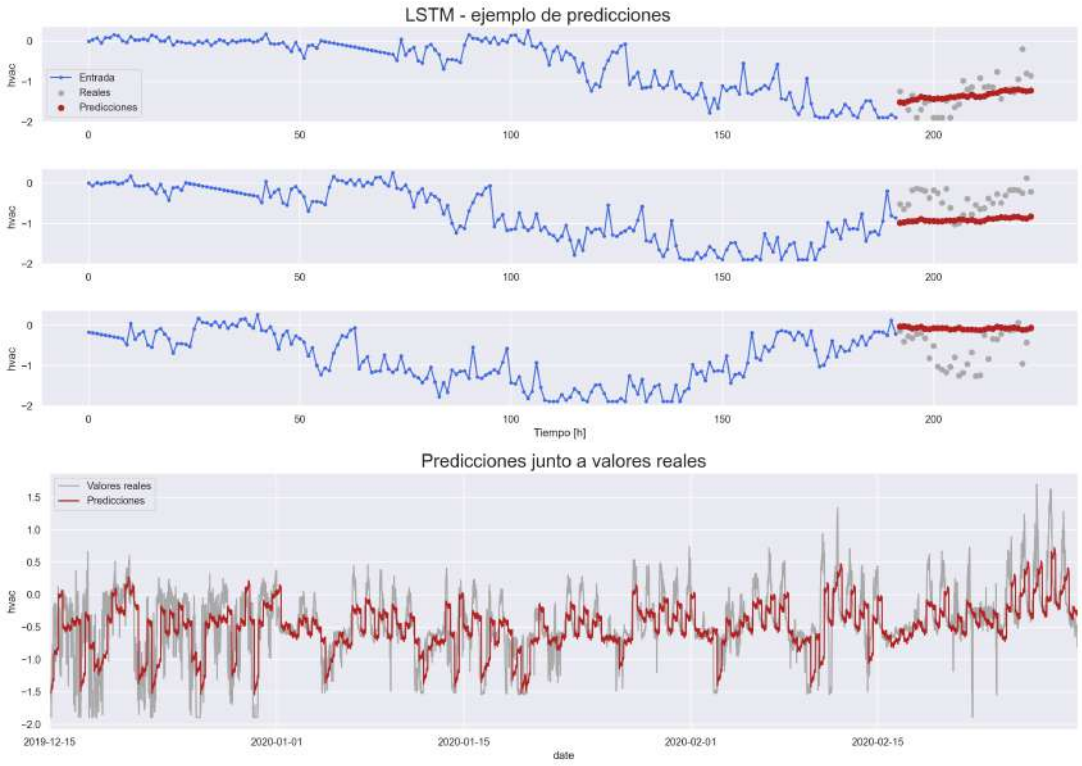

Figura 80: Predicciones obtenidas con el modelo LSTM_32_modelo2

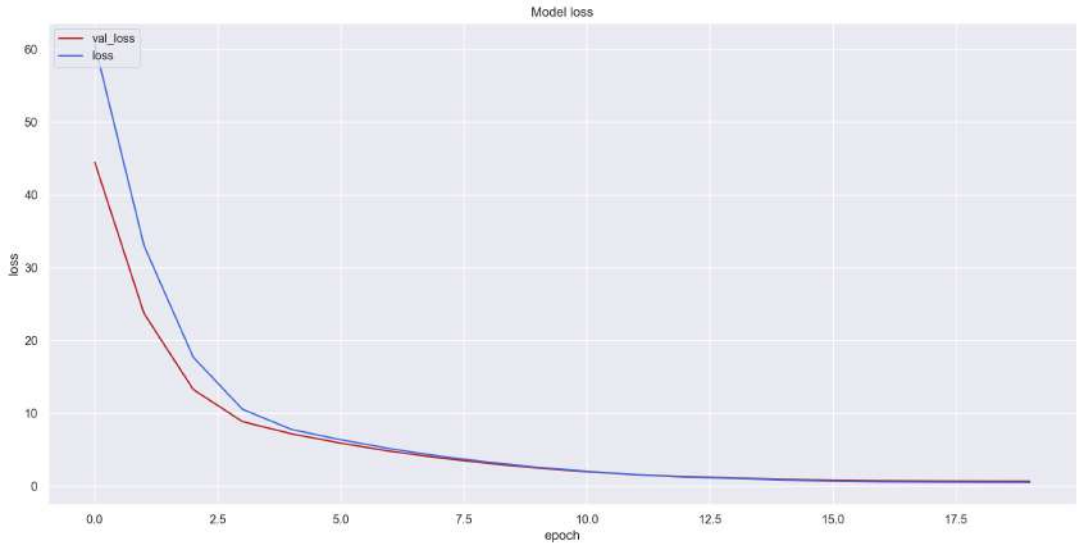

Figura 81: Función de pérdida del modelo LSTM_32_modelo2

Figura 82: Predicciones obtenidas con el modelo LSTM_32_modelo2 reentrenando la red

Modelo multivariable

Por último, vamos a implementar un modelo multivariable, con la estructura de variables que ya hemos presentado anteriormente.

Al enfrentarse al conjunto de variables surge una duda lógica, y es si todas las variables son relevantes para el problema, o si por el contrario incluirlas añade ruido al entrenamiento y perjudica la calidad de las predicciones.

Para responder a la pregunta vamos a considerar, en primer lugar, una red que reciba como entrada todas las variables. Nos hemos basado en la arquitectura compleja que acabamos de estudiar, reduciendo el número de unidades de las capas densas intermedias de 80 a 60, pues tras realizar pruebas con valores más pequeños se descubrió que al utilizar este valor se obtenían mejores resultados.

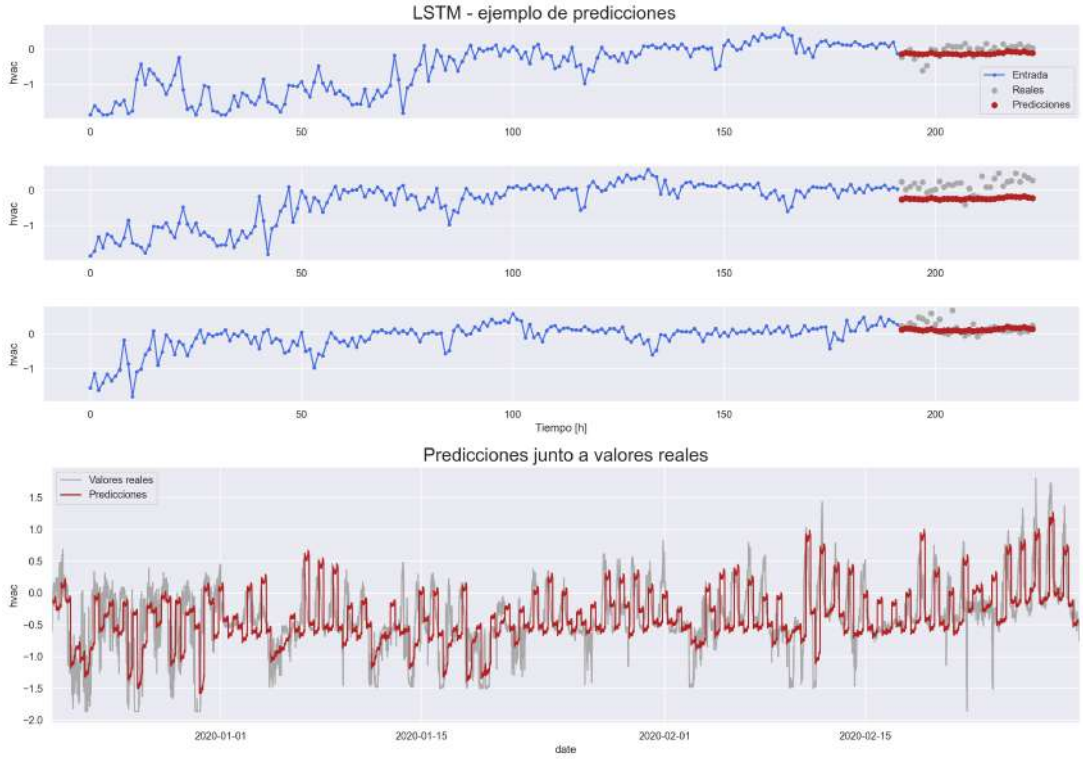

Figura 83: Predicciones obtenidas con el modelo multivariable

En este caso, podemos comprobar que la calidad de las predicciones ha aumentado considerablemente, especialmente al analizar el patrón de subidas y bajadas, que en este caso sí que se capta prácticamente al completo (Figura 83).

En cuanto a la función pérdida, representada en la Figura 84, podemos comprobar que no se produce overfitting, y que la curva tiene la forma esperada. Para entrenar este modelo se ha utilizado un *learning rate* de 0,001.

Es necesario analizar cuántas variables sería preciso añadir al modelo, teniendo en cuenta que a medida que aumenta la dimensionalidad de los datos de entrada, tam-

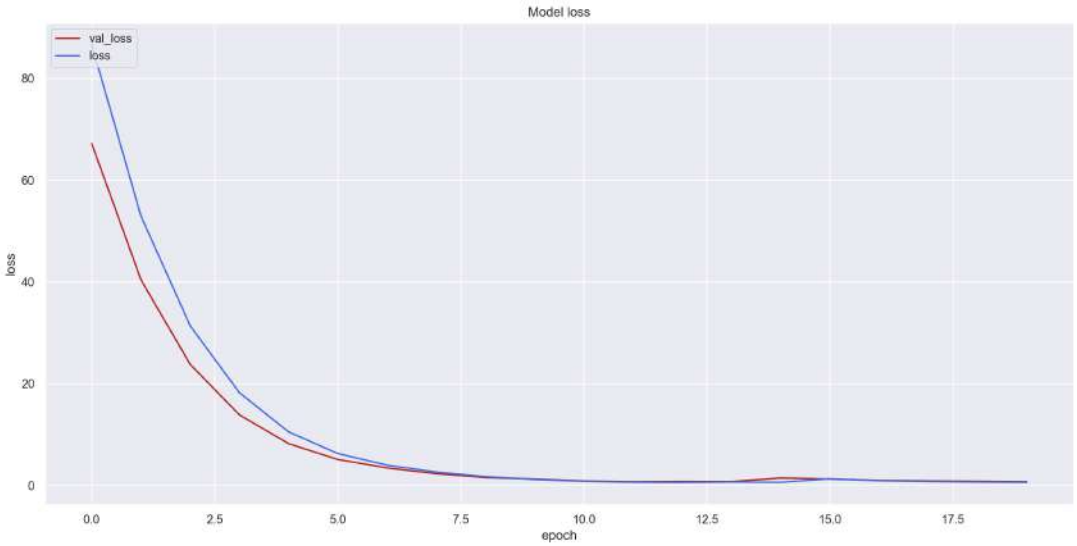

Figura 84: Función de pérdida del modelo multivariable

bién aumenta el coste de procesamiento, y que si las variables que se están añadiendo no son relevantes, el efecto que puede tener el incluirlas puede ser negativo.

Por este motivo, se realizó un experimento partiendo del modelo anterior que recibe como entrada únicamente la variable objetivo y la estacionalidad en forma de series de Fourier. Sobre ese modelo, se probó el efecto individual de añadir una a una las variables restantes, y se mantuvo la que producía una mejora mayor. Este último paso se realizó hasta que o bien se terminaron las variables o ninguna de ellas produjo una mejora. Este proceso se denomina comúnmente *forward selection*.

Tras llevarlo a cabo descubrimos que en la primera iteración se obtenía la variable *mels_N*, y en la segunda, la variable *air_temp*. En la tercera iteración ya no se producía ninguna mejora al añadir variables, por lo que se decidió proceder con el subconjunto obtenido.

Al observar los resultados, podemos concluir que funciona prácticamente igual que el modelo anterior (Figura 85), y aunque sería necesario estudiarlos más a fondo para poder obtener una conclusión, consideramos adecuado el descarte del resto de variables incluidas en el primer modelo.

En la Figura 86 podemos comprobar como de nuevo, la regularización presente en el modelo previene el sobreajuste, y el entrenamiento se lleva a cabo de una manera correcta.

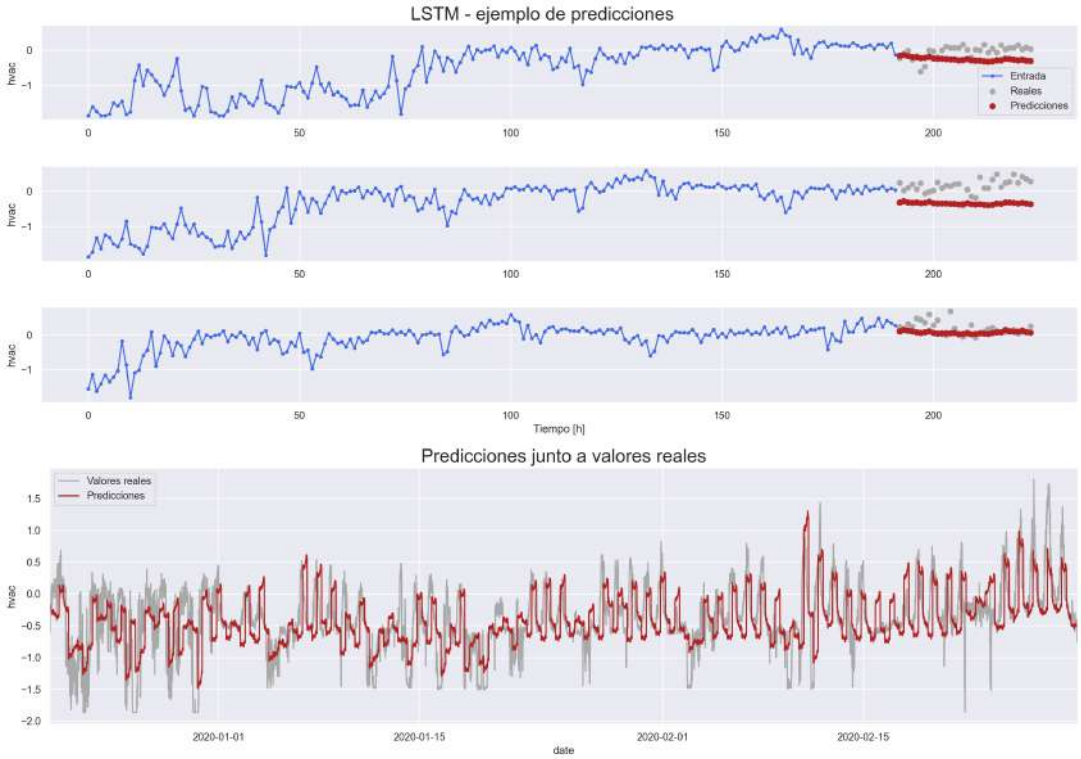

Figura 85: Predicciones obtenidas con el modelo multivariable tras seleccionar variables

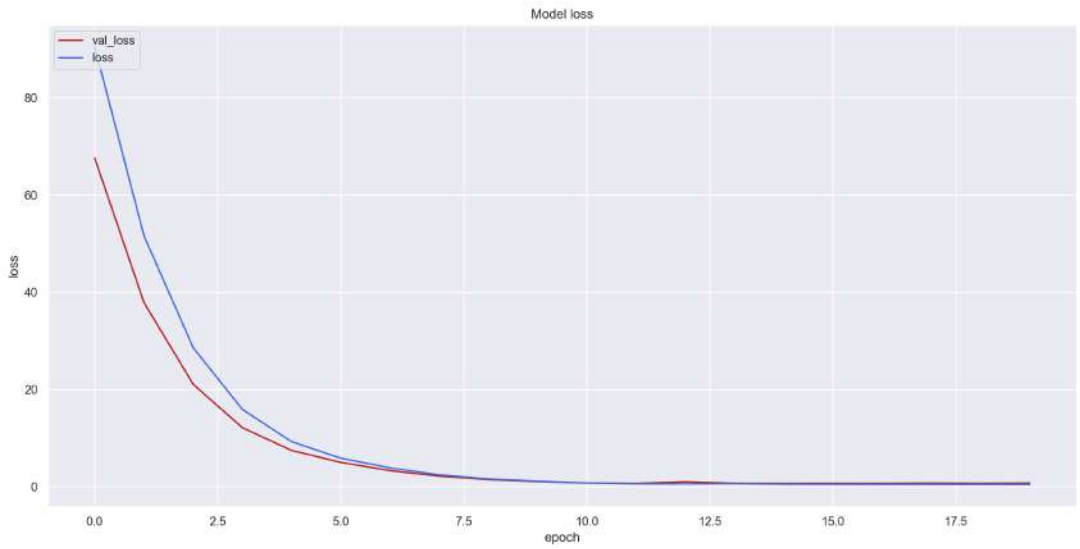

Figura 86: Función de pérdida del modelo multivariable al reducir el número de variables

11.3 ANÁLISIS DE RESULTADOS

A continuación vamos a comparar los resultados obtenidos en los experimentos que acabamos de estudiar. Para cada tamaño de ventana, se compararan tres medidas de rendimiento de cada algoritmo entrenado: la raíz del error cuadrático medio (*RMSE*), el error absoluto medio (*MAE*) y el error porcentual absoluto medio (*MAPE*).

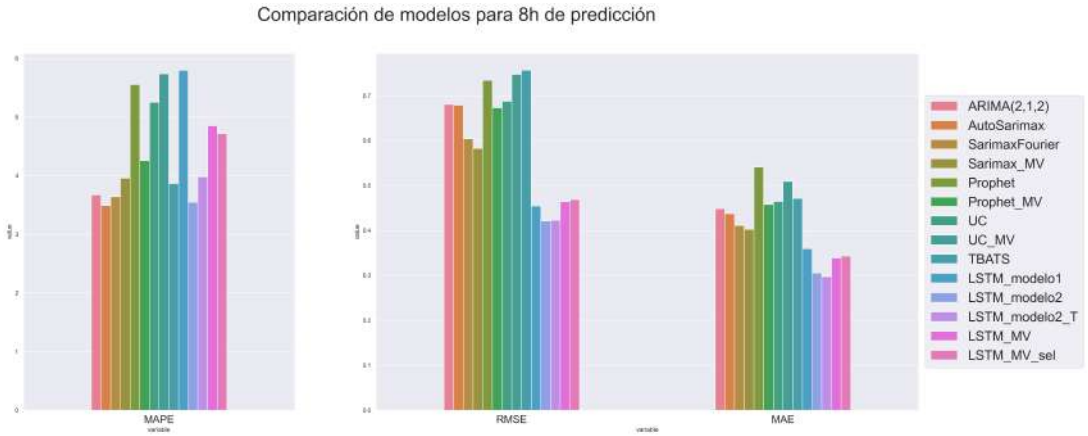

Figura 87: Comparación de modelos para el tamaño de ventana principal

Podemos determinar que las redes neuronales obtenidas se comportan mejor que los modelos de Machine Learning (Figura 87), pues tienen un MAE y un MSE menor. Sin embargo, no se puede decir lo mismo del MAPE, lo que nos lleva a deducir que las predicciones de estos modelos están más alejadas en términos porcentuales de los valores reales. Sería necesario determinar qué medida tiene más peso para poder tomar una decisión, por una parte, el MAPE proporciona una medida que es especialmente útil cuando se trata de evaluar los costos o la eficiencia energética, pues se trata de una medida porcentual. Por otra parte, el MAE y el MSE proporcionan información sobre la magnitud absoluta de los errores y pueden ser útiles para evaluar la precisión del modelo.

En nuestro caso, al ser el objetivo suavizar los picos de consumo, resultarán más útiles las dos últimas, pues nos permiten determinar si un modelo realiza predicciones precisas o no.

También cabe destacar el aumento en el MSE y MAE al añadir las variables exógenas en los algoritmos *SARIMAX*, Prophet y Unobserved Components. Podemos determinar entonces que la inclusión de todas las variables es perjudicial para el modelo y sería necesario realizar un estudio más concreto de cuáles resultarían útiles para mejorar las predicciones.

En la Figura 88 podemos estudiar los resultados obtenidos al hacer predicciones de 16 horas. En este caso únicamente comparamos los algoritmos de Machine Learning, y podemos determinar que *SARIMAX* con la estacionalidad como serie de Fourier es la mejor alternativa. Por otra parte, destaca el rendimiento negativo del algoritmo

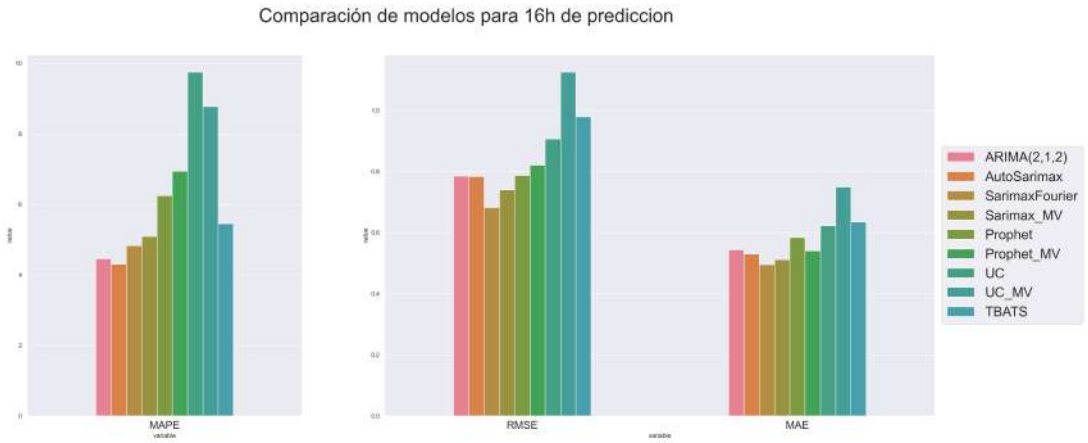

Figura 88: Comparación de modelos con 16 horas de predicción

Unobserved Components con las variables exógenas, que puede deberse, como ya hemos comentado, a ruido innecesario.

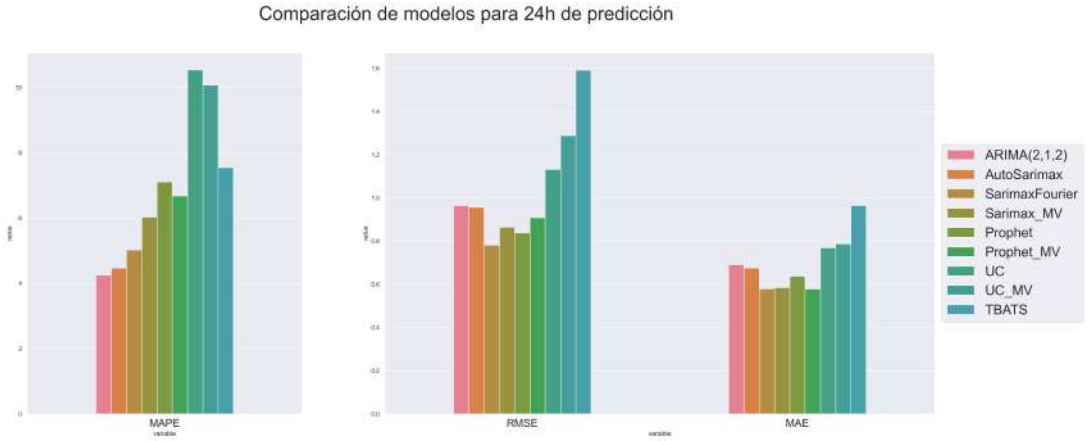

Figura 89: Comparación de modelos con 24h de predicción

En el caso de realizar predicciones con 24h de antelación (Figura 89), se mantienen las observaciones realizadas en el tamaño de ventana anterior.

Sin embargo, al predecir directamente todo el conjunto de prueba (Figura 90), comprobamos como los modelos *ARIMA* y *SARIMAX* en su versión univariable se comportan de una manera prácticamente idéntica, pues se tratan simplemente de la convergencia en el infinito de los procesos estocásticos correspondientes.

Destaca el buen rendimiento en las tres métricas tanto de *SARIMAX* y Prophet multivariable como de las dos alternativas de Unobserved Components. No obstante, esto es un ejemplo de métricas que pueden llevar a engaño, pues basta con observar el resultado visual de las predicciones en secciones anteriores para determinar que ninguna alternativa es adecuada.

Se puede consultar un resumen de las conclusiones y resultados obtenidos en la Figura 91.

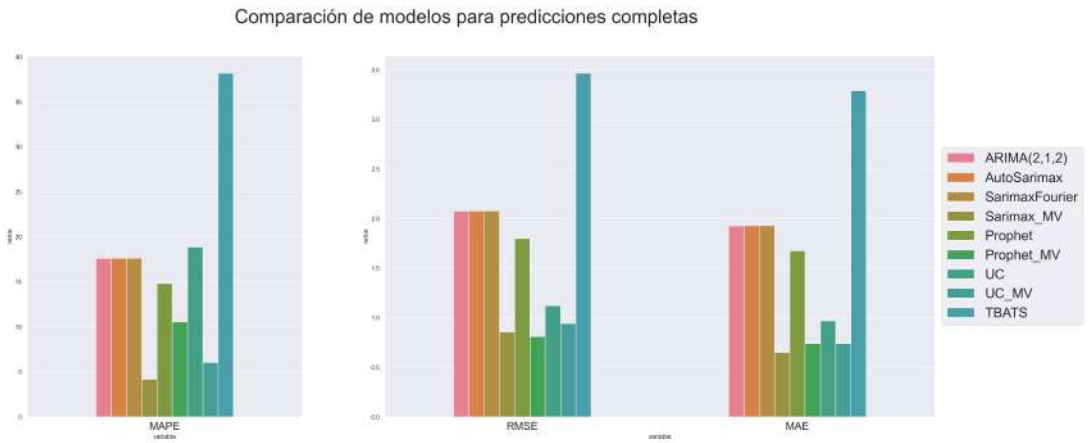

Figura 90: Comparación de modelos al hacer predicciones completas

Experimento	Ventana	MSE	MAE	MAPE	Conclusiones principales
ARIMA(2,1,2)	32	0,46	0,45	3,67	En el modelo que **no usa ventana móvil** las predicciones resultantes son una línea recta, que representa la convergencia del modelo hacia un valor de equilibrio a largo plazo, y por tanto **no son válidas.**
	64	0,62	0,54	4,46	
	96	0,93	0,69	4,25	
	Completo	4,31	1,93	17,62	
SARIMAX(3,1,1)x(2,0,1,4)	32	0,46	0,44	3,49	Las predicciones mejoran el modelo anterior, pero **la diferencia no es significativa**. En el modelo Completo volvemos a tener la **situación de equilibrio a largo plazo**.
	64	0,61	0,53	4,30	
	96	0,91	0,68	4,47	
	Completo	4,31	1,93	17,63	
SARIMAX(3,1,1)x(2,0,1,4) con series de Fourier	32	0,36	0,41	3,64	Podemos observar una **mejora relativamente importante** en la precisión de las predicciones. En el modelo Completo **converge** a la situación de equilibrio **de una manera más lenta**.
	64	0,47	0,50	4,83	
	96	0,61	0,58	5,03	
	Completo	4,31	1,93	17,63	
SARIMAX(3,1,1)x(2,0,1,4) con series de Fourier y multivariable	32	0,34	0,40	3,96	Es el experimento con el **coste más alto en tiempo de procesamiento**. Destacamos la **bondad de las métricas obtenidas en el modelo Completo**, aunque al observar la gráfica concluimos que **no resulta adeucado**.
	64	0,55	0,51	5,09	
	96	0,75	0,58	6,03	
	Completo	0,73	0,65	4,17	
Prophet	32	0,54	0,54	5,56	Las predicciones son **menos precisas en general** y podemos observar **menor diferencia en la calidad** de las predicciones **entre las distintas longitudes** de ventana móvil.
	64	0,62	0,58	6,25	
	96	0,70	0,64	7,11	
	Completo	3,24	1,68	14,84	
Prophet multivariable	32	0,45	0,46	4,26	La inclusión de variables exógenas **influencia negativamente en la calidad de las predicciones**. La versión completa captura **ciertos patrones** de la serie por primera vez.
	64	0,67	0,54	6,94	
	96	0,83	0,58	6,68	
	Completo	0,66	0,74	10,56	
Unobserved Components	32	0,47	0,46	5,25	Se **amplió el número de datos de entrada** para mejorar la precisión. En las versiones con ventana móvil **captura correctamente las fluctuaciones** pero en ocasiones da lugar a **valores extremos que pueden afectar a las métricas.**
	64	0,82	0,62	9,75	
	96	1,28	0,77	10,54	
	Completo	1,26	0,97	18,88	
Unobserved Components multivariable	32	0,56	0,51	5,74	Al predecir 32 observaciones **se soluciona el problema de valores extremos**. Además, la versión completa tiene mejores métricas, pero al observar la gráfica podemos concluir que **no captura correctamente los patrones de la serie**.
	64	1,27	0,75	8,77	
	96	1,66	0,79	10,07	
	Completo	0,89	0,74	6,04	
TBATS	32	0,57	0,47	3,87	Conlleva un **gran tiempo de entrenamiento** y **no produce buenas predicciones**, por lo que podemos determinar que **no resulta adecuado para la serie considerada**. Además, no permite incluir variables exógenas.
	64	0,96	0,64	5,45	
	96	2,53	0,96	7,55	
	Completo	12,03	3,29	38,20	
LSTM 1	32	0,21	0,36	5,80	Las redes LSTM son las que **mejores resultados producen**, en particular la **arquitectura compleja univariable**. En la versión multivariable, hemos detectado **ruido producido por las variables exógenas,** por lo que sería interesante hacer un estudio más a fondo de las mismas.
LSTM 2	32	0,18	0,31	3,55	
LSTM 2 reentrenado	32	0,18	0,30	3,98	
LSTM multivariable 1	32	0,22	0,34	4,85	
LSTM multivariable FS	32	0,22	0,34	4,72	

Figura 91: Resumen de resultados

11.4 CONCLUSIONES Y TRABAJO FUTURO

Tras abordar el problema de la predicción del consumo energético utilizando diferentes enfoques, desde algoritmos clásicos de Machine Learning hasta redes neuronales LSTM, podemos afirmar que con la arquitectura más sofisticada de redes LSTM se han conseguido resultados más que aceptables para poder tomar decisiones administrativas en el edificio, pues este modelo es capaz de predecir tanto las subidas como las bajadas repentinas.

A pesar de haber obtenido un modelo que tiene un rendimiento positivo, es necesario destacar la complejidad del problema debido a la naturaleza cambiante del consumo energético, que se puede ver al observar la forma de la variable objetivo. Por este motivo, se hace difícil seleccionar periodos de entrenamiento y prueba, pues la estructura es completamente diferente.

Además, también es necesario mencionar una limitación importante que suele estar presente cuando se trata con datos de consumo energético, y son los datos nulos. En este trabajo nos hemos enfrentado a los datos nulos proponiendo una solución aceptable cuando el problema de la falta de datos afecta a varias variables al mismo tiempo y por tanto, no se puede proceder mediante alternativas multivariables. Sin embargo, los datos imputados modifican aún más la estructura ya difícil de los datos originales.

Hemos comprobado cómo los algoritmos de aprendizaje supervisado tienen potencial para solucionar el problema planteado, y sería pertinente por lo tanto llevar a cabo un estudio a fondo de la correlación e importancia de las variables exógenas con la variable objetivo, para poder solucionar el problema del ruido añadido al considerar todas las variables.

También se ha visto que una arquitectura de red neuronal más compleja puede mejorar considerablemente el problema, dejando la puerta abierta a experimentar con diferentes arquitecturas o incluso experimentar con técnicas de *Transfer Learning*, en la que se utilizan redes neuronales muy complejas ya entrenadas y se adaptan a los datos con los que se esté tratando.

Tanto las herramientas discutidas como el algoritmo de imputación desarrollado se ponen a disposición de los expertos en el campo, con la esperanza de que puedan ser utilizadas y adaptadas para abordar otros problemas relacionados con la predicción de series temporales en el ámbito del consumo energético. Se espera que este trabajo sirva como base para futuras investigaciones y desarrollos en esta área, contribuyendo así al avance del conocimiento y las soluciones en un campo tan relevante como la gestión de la energía.

BIBLIOGRAFÍA

[1] Kashif Abbass, Muhammad Zeeshan Qasim, Huaming Song, Muntasir Murshed, Haider Mahmood, and Ijaz Younis. A review of the global climate change impacts, adaptation, and sustainable mitigation measures. *Environ. Sci. Pollut. Res. Int.*, 29(28):42539–42559, 2022.

[2] Ethem Alpaydin. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 3 edition, 2014. ISBN 978-0-262-02818-9.

[3] G. E. P. Box and David A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65(332):1509–1526, 1970. doi: 10.1080/01621459.1970.10481180.

[4] Peter J. Brockwell and Richard A. Davis. *Time series: Theory and methods*. Springer New York, New York, NY, 1991.

[5] Matthieu Cord and Pádraig Cunningham, editors. *Machine learning techniques for multimedia: Case studies on organization and retrieval*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[6] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renew. Sustain. Energy Rev.*, 74:902–924, 2017. doi: 10.1016/j.rser.2017.02.085.

[7] M.H. DeGroot and M.J. Schervish. *Probability and Statistics*. Addison-Wesley, 2012. ISBN 9780321500465. URL https://books.google.es/books?id=4TlEPgAACAAJ.

[8] D. Dickey and Wayne Fuller. Distribution of the estimators for autoregressive time series with a unit root. *JASA. Journal of the American Statistical Association*, 74, 06 1979. doi: 10.2307/2286348.

[9] Soheil Fathi, Ravi Srinivasan, Andriel Fenner, and Sahand Fathi. Machine learning applications in urban building energy performance forecasting: A systematic review. *Renew. Sustain. Energy Rev.*, 133:110287, 2020. doi: 10.1016/j.rser.2020.110287.

[10] Everette S. Gardner and Ed. McKenzie. Forecasting trends in time series. *Management Science*, 31(10):1237–1246, 1985. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2631713.

[11] William H. Greene. *Econometric Analysis*. Pearson Education, fifth edition, 2003. ISBN 0-13-066189-9. URL http://pages.stern.nyu.edu/~wgreene/Text/econometricanalysis.htm.

[12] Frank E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969. doi: 10.1080/00401706.1969.10490657.

[13] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2006.03.001.

[14] Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and practice*. Otexts, 3rd edition, 2021. ISBN 978-0987507136.

[15] International Energy Agency. Net zero by 2050 – analysis. https://www.iea.org/reports/net-zero-by-2050, 2021. Accessed: 2023-2-13.

[16] Ben Kröse and Patrick van der Smagt. *An introduction to neural networks*. University of Amsterdam, 11 1996. URL https://www.infor.uva.es/~teodoro/neuro-intro.pdf.

[17] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*, pages 481–492, Berkeley and Los Angeles, 1951. University of California Press.

[18] Ke Li and Boqiang Lin. Impacts of urbanization and industrialization on energy consumption/CO2 emissions: Does the level of development matter? *Renew. Sustain. Energy Rev.*, 52:1107–1122, 2015.

[19] Na Luo, Zhe Wang, David Blum, Christopher Weyandt, Norman Bourassa, Mary Ann Piette, and Tianzhen Hong. A three-year dataset supporting research on building energy management and occupancy analytics. *Sci. Data*, 9(1):156, 2022.

[20] Jun Ma, Jack C P Cheng, Feifeng Jiang, Weiwei Chen, Mingzhu Wang, and Chong Zhai. A bi-directional missing data imputation scheme based on LSTM and transfer learning for building energy data. *Energy Build.*, 216:109941, 2020.

[21] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386, 2020.

[22] Dr Henry R. Neave. The teaching of hypothesis-testing. *Journal of Applied Statistics*, 3(1):55–63, 1976. doi: 10.1080/768371017.

[23] Thi-Thu-Hong Phan. Machine learning for univariate time series imputation. In *2020 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, 10 2020. doi: 10.1109/MAPR49794.2020.9237768.

[24] Stuart J. Russell. *Inteligencia artificial : un enfoque moderno / Stuart J. Russell y Peter Norvig ; traducción y revisión técnica Juan Manuel Corchado Rodríguez*. Prentice Hall, Madrid, 2ª ed. edition, 2006. ISBN 9788420540030.

[25] Michael Schomaker and Christian Heumann. Model selection and model averaging after multiple imputation. *Computational Statistics & Data Analysis*, 71(C): 758–770, 2014. doi: 10.5555/2749482.2749833.

[26] Saleh Seyedzadeh, Farzad Pour Rahimian, Ivan Glesk, and Marc Roper. Machine learning for estimation of building energy consumption and performance: a review. *Vis. Eng.*, 6(1), 2018.

[27] Aashish Sharma, Abhishek Saxena, Muneesh Sethi, Venu Shree, and Varun. Life cycle assessment of buildings: A review. *Renewable and Sustainable Energy Reviews*, 15(1):871–875, 2011. doi: 10.1016/j.rser.2010.09.008.

[28] Sean J. Taylor and Benjamin Letham. Forecasting at scale. *The American Statatistician*, 72(1):37–45, 2018. doi: 10.1080/00031305.2017.1380080.

[29] United Nations Environment Programme. Buildings and climate change: Summary for decision makers. 2009.

[30] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. ISBN 0-387-94559-8.