TRABAJO FIN DE MÁSTER

MÁSTER PROFESIONAL EN INGENIERÍA INFORMÁTICA

Deep learning para simulación energética de edificios

Autor
Jesús Mesa González

Director
Juan Gómez Romero

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 09 de Julio de 2020

Deep learning para simulación energética de edificios

Autor
Jesús Mesa González

Director
Juan Gómez Romero

Deep learning para simulación energética de edificios

Jesús Mesa González

Palabras clave: Edificios, sensores, predicción de consumo energético, series temporales, descomposición, autocorrelación, aprendizaje automático, aprendizaje profundo, XGBoost, perceptrón multicapa, redes neuronales recurrentes, redes neuronales convolucionales, redes sequence to sequence, análisis exploratorio, preprocesamiento, outliers, valores perdidos, selección de variables, correlación

Resumen

En los últimos años los edificios residenciales, de oficina e industriales han supuesto más de un tercio del consumo energético mundial, razón por la cual cada vez es mayor el interés en mejorar la eficiencia energética de los mismos. En este trabajo de fin de máster se ha abordado un problema de creación de modelos de predicción del consumo de energía en un edificio bajo las estrategias de operación normales —esto es, sin modificar los parámetros de control de los equipos de calefacción, aire acondicionado, iluminación, etc— a partir de datos históricos y con alto nivel de precisión, de forma que dichos modelos sirvan a los ingenieros de energía responsables del edificio para caracterizar el funcionamiento base del mismo e identificar oportunidades de mejora.

Específicamente, en este trabajo se ha aplicado conjunto de tareas de análisis y preprocesamiento a un conjunto de datos real que contiene valores históricos de consumo energético registrados a lo largo del año 2016 por los sensores de un edificio de oficinas denominado ICPE, situado en Rumanía. Todo esto con el objetivo de en última instancia poder aplicar a dicho conjunto de datos una serie de técnicas de aprendizaje automático y en especial de aprendizaje profundo que permitan predecir a partir de valores históricos de varios sensores el consumo futuro de energía destinada a los sistemas de calefacción del edificio.

Como consecuencia de las tareas de análisis y preprocesamiento solo se ha podido trabajar con valores históricos de sensores de una zona piloto del edificio correspondientes a los meses de Enero, Febrero y Marzo de 2016, lo que acarrea, entre otros, el principal inconveniente de no disponer de una gran variedad y cantidad de datos con los que entrenar los distintos modelos. Sin embargo, a pesar de esto hemos podido comprobar que las técnicas de aprendizaje profundo y en especial aquellas especializadas en el tratamiento de series temporales nos ofrecen modelos con una gran capacidad de aprendizaje y de generalización a la hora de resolver este problema de predicción planteado.

Deep learning for buildings energy simulation

Jesús Mesa González

Keywords: Buildings, sensors, energy consumption forecasting, time series, decomposition, autocorrelation, machine learning, deep learning, XGBoost, multilayer perceptron, recurrent neural networks, convolutional neural networks, sequence to sequence networks, exploratory analysis, pre-processing, outliers, missing values, variable selection, correlation

Abstract

In recent years residential, office and industrial buildings have accounted for more than a third of global energy consumption, so there is growing interest in improving their energy efficiency. In this work, a problem has been addressed of creating models to predict energy consumption in a building under normal operating strategies -that is, without modifying the control parameters of the heating, air conditioning, lighting, etc.- based on historical data and with a high level of accuracy, so that these models can be used by the energy engineers responsible for the building to characterize the basic operation of the building and identify opportunities for improvement.

Specifically, in this work a real data set with historical energy consumption values recorded throughout the year 2016 by the sensors of an office building called ICPE, located in Romania, has been taken and a set of analysis and pre-processing tasks have been applied to it. All this with the aim of ultimately being able to apply to this set a series of automatic learning techniques and in particular deep learning techniques that allow the future energy consumption for the building's heating systems to be predicted from historical values of several sensors.

As a consequence of the analysis and pre-processing tasks, it has only been possible to work with historical values of sensors from a pilot area of the building corresponding to the months of January, February and March 2016, which raises, among other things, the main inconvenience of not having a great variety and quantity of data with which to train the different models. However, in spite of this we have noticed that the techniques of deep learning and especially those specialized in the treatment of time series offer us models with a great capacity of learning and generalization at the time of solving this problem of prediction raised.

Yo, **Jesús Mesa González**, alumno del **Máster Profesional en Ingeniería Informática** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 50610887P, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Jesús Mesa González

Granada a 03 de Julio de 2020

D. **Juan Gómez Romero**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Deep learning para simulación energética de edificios***, ha sido realizado bajo su supervisión por **Jesús Mesa González**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 09 de Julio de 2020.

El director:

Juan Gómez Romero

Yo, **Jesús Mesa González**, alumno del **Máster Profesional en Ingeniería Informática** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 50610887P, declaro explícitamente que el trabajo presentado es original, entendido en el sentido que no he utilizado ninguna fuente sin citarla debidamente.

Fdo: Jesús Mesa González

Granada a 03 de Julio de 2020

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo de exponen la motivación principal para llevar a cabo este trabajo, los objetivos generales del mismo, la planificación temporal y estimación de costes de las tareas que llevaremos a cabo para alcanzar los objetivos, una breve exploración de otros trabajos recientes similares a este y la estructuración de los capítulos que componen esta memoria.

1.1. Motivación

De acuerdo con [**1**] los edificios residenciales, de oficinas e industriales suponen entre el 20 y el 40 % del consumo energético mundial. Así, en el año 2010 en EE.UU. el consumo energético de los edificios representó el 41 % de su consumo energético total [**2**] procediendo el 75 % de dicha energía de combustibles fósiles. Mientras, en 2012 en la Unión Europea los edificios consumieron aproximadamente el 40 % de la energía total utilizada [**2**].

Más de dos terceras partes de la energía consumida por los edificios va destinada a los sistemas de calefacción (37 %), calentamiento de agua (12 %), aire acondicionado (10 %) e iluminación (9 %). Además, varios casos de estudio han demostrado que la fase operacional de un edificio es la etapa de su ciclo de vida que más energía consume con un 90 % en edificios convencionales y un 50 % en edificios de baja energía [**3**].

Por tanto, en los últimos años cada vez es mayor el interés en mejorar la eficiencia energética de los edificios. Un interés impulsado de acuerdo con [**4**] y [**5**] por tres factores: el precio creciente de la energía, las políticas medioambientales cada vez más restrictivas y la mayor concienciación medioambiental por parte de los usuarios y consumidores. En todo el mundo se están desarrollando políticas destinadas a incrementar esta eficiencia, como reflejan los Objetivos de Desarrollo Sostenible de Naciones Unidas[1] y el Pacto Verde Europeo[2].

[1] `https://sustainabledevelopment.un.org/`
[2] https://ec.europa.eu/european-green-deal_es

Para mejorar la eficiencia energética de un edificio, en primer lugar es necesario estudiar cómo se produce este consumo y conocer los factores que mayor impacto tienen en el mismo. En particular, la predicción del consumo energético a partir de datos históricos permite a los operadores y gestores de los edificios anticipar picos de demanda energética, modificar los usos del edificio para desplazar esta demanda y planificar adecuadamente la operación de los equipos. Igualmente, la predicción precisa del consumo permite valorar la mejora en el rendimiento del edificio al realizar mejoras o implementar nuevas políticas energéticas, comparando el consumo real con el consumo estimado (o *línea base*).

En la literatura podemos encontrar numerosas propuestas para la estimación de consumo energético en edificios mediante predicción con series temporales. Tradicionalmente, estas aproximaciones se han basado en regresión numérica o modelos de medias móviles (ver Sección 1.4). Estas técnicas presentan diversas limitaciones que, en la actualidad, están siendo superadas por los métodos de aprendizaje automático basados en redes neuronales. Por otra parte, la aplicación de cualquier técnica a un problema real requiere un ingente trabajo de exploración y preprocesamiento, ya que los datos que se obtienen de los sistemas de control de un edificio (normalmente de tipo SCADA, *Supervisory Control And Data Acquisition*) suelen estar pobremente documentados y afectados de errores (ruido, incompletitud, etc.).

Estos problemas aparecieron en el proyecto europeo Energy IN TIME **[6]**, en el que participaron varios miembros del equipo de investigación dentro del que se desarrolla el presente Trabajo Fin de Máster. El objetivo de este proyecto consistía en la implementación de un sistema de control predictivo para mejora de la eficiencia energética de edificios no residenciales. El algoritmo de control del proyecto estaba basado en la simulación del funcionamiento del edificio utilizando un modelo físico (con ecuaciones diferenciales para cálculo de transferencia de energía), que permitía conocer el consumo esperado de energía ante una determinada secuencia de control y las condiciones de confort del edificio (temperatura, concentración de CO2, etc.). El principal obstáculo de este tipo de modelos es que las simulaciones físicas requieren un gran esfuerzo de desarrollo y mucho tiempo de ejecución, por lo que las posibilidades de mejora con un algoritmo de control de este tipo son reducidas.

En este Trabajo Fin de Máster se aborda un problema no resuelto en Energy IN TIME: la creación de modelos de predicción del consumo de energía en un edificio bajo las estrategias de operación normales —esto es, sin modificar los parámetros de control de los equipos— a partir de datos históricos y con alto nivel de precisión. Desde el punto de vista del Ingeniero en Energía, este tipo de modelos sirven para caracterizar el funcionamiento base de los edificios e identificar oportunidades de mejora. Desde el punto de vista del Ingeniero en Informática, el problema ofrece la oportunidad de estudiar, desarrollar y evaluar diferentes técnicas de preprocesamiento de

datos y aprendizaje automático en un contexto real, afrontando situaciones que no suelen considerarse cuando se trabaja con conjuntos de datos estándar de prueba.

Específicamente, en este trabajo se tomará un conjunto de datos real con valores de sensores de un edificio de oficinas denominado ICPE, situado en Rumanía y del que se dispone de datos de sensores relacionados con el consumo energético de varios años. A partir de los valores de algunos de los sensores de dicho conjunto se crearán modelos de predicción del consumo energético del edificio aplicando técnicas de aprendizaje automático y, más especialmente, de aprendizaje profundo. Nos centraremos en predecir el consumo de energía eléctrica destinada a la calefacción (consumo de calefacción a partir de ahora) del edificio, pues como hemos dicho al principio, los sistemas de calefacción son los que más energía consumen en un edificio.

1.2. Objetivos, resultados y tareas

El objetivo principal de este trabajo es investigar, construir y evaluar diversas técnicas actuales de aprendizaje profundo para la construcción de modelos de predicción del consumo energético (a corto plazo y debido a calefacción) a partir de los datos históricos disponibles del edificio ICPE. El resultado esperado es un análisis crítico de diferentes técnicas actuales de preprocesamiento y aprendizaje automático con series temporales en términos de precisión de los resultados en el contexto de la eficiencia energética. El tipo de aproximación concreta que se estudiará en el trabajo es la conocida como predicción multistep (predicción de varios valores en el futuro) de series temporales multivariable (valores pasados de varios sensores).

Para ello, será necesario llevar a cabo las siguientes tareas:

- Investigar acerca de técnicas de predicción de series temporales y de otros trabajos de investigación con una motivación y objetivos similares a este.

- Analizar el edificio ICPE para tener un mejor conocimiento de su estructura y poder elegir el conjunto de sensores concreto con el que vamos a trabajar.

- Obtener el conjunto de datos con los valores de los sensores del edificio.

- Analizar el conjunto de datos con el propósito de conocer las características de los sensores: Qué miden, en que zona del edificio se encuentran, etc. Esta tarea nos permitirá además identificar los sensores que miden el consumo de calefacción y seleccionar aquel o aquellos cuyos valores se pretenden predecir.

- Llevar a cabo un preprocesamiento del conjunto de datos obtenido en las tareas anteriores con el objetivo de poder aplicarle técnicas de aprendizaje automático para obtener modelos de predicción.
- Definir el problema concreto de predicción, estableciendo:
  - El horizonte temporal de la predicción del consumo de calefacción, es decir, en cuantas horas o días en el futuro deseamos predecir el consumo de calefacción.
  - Cuánto miramos hacia atrás para realizar la predicción, es decir, a partir de cuantos valores pasados de los sensores del edificio realizamos la predicción del consumo de calefacción.
- Obtener varios modelos con distintas técnicas para abordar el problema de predicción, compararlos y sacar conclusiones acerca de su rendimiento.

1.3. Planificación temporal y estimación de costes

A continuación vamos a realizar la planificación de las distintas tareas citadas en la sección 1.2 necesarias para alcanzar los objetivos de este proyecto además de una estimación de los costes que supondrá la realización de las mismas.

1.3.1. Planificación temporal

Para llevar a cabo la planificación temporal vamos a emplear un diagrama de Gantt que nos va a permitir organizar a lo largo del tiempo las tareas de nuestro proyecto estebleciendo su duración y las dependencias entre las mismas. De esta forma, en la siguiente figura podemos ver el diagrama de Gantt con la planificación de las tareas de este proyecto.

Figura 1.1: Planificación temporal del proyecto.

En la figura 1.1 vemos en la parte izquierda las tareas que llevaremos a cabo en el proyecto y en la parte derecha su duración estimada y fechas de inicio y fin (barras horizontales). En este diagrama se observa lo siguiente.

- Las tareas principales (resaltadas en negrita y con barra temporal más oscura) se llevan a cabo una detrás de otra de forma secuencial, pues dependen las unas de las otras.
- Dentro de cada tarea principal hay varias subtareas las cuales casi siempre se llevarán a cabo una detrás de otra debido a que solo van a poder ser realizadas por una persona y a las dependencias entre ellas, eso sí vemos que la subtarea de documentación de cada tarea principal si podremos llevarla a cabo en paralelo con otras tareas.
- Si observamos la barra temporal de arriba vemos que el proyecto se llevará a cabo durante los meses de Octubre, Noviembre y Diciembre de 2019 y de Enero, Febrero, Marzo, Abril, Mayo y Junio de 2020 empezando el 1 de Octubre y terminando el 30 de Junio, pues para Julio ya debería estar acabado y listo para entregar.
- También en la barra temporal podemos observar tres triángulos de color azul invertidos, los cuales indican la fecha antes de la cual debemos alcanzar los hitos principales del proyecto, que son Obtener los datos preprocesados, finalizar la experimentación con los modelos de predicción y tener el proyecto listo para entregar.
- Para cada tarea principal podemos ver los días totales que hemos estimado que nos llevará su realización sin contar los días festivos y los fines de semana. De esta manera si sumamos los días de todas las tareas principales obtenemos un total de 114 días (sin contar los días

de la tarea de terminar y revisar la documentación debido a que para entonces tendrá que estar casi todo terminado y solo faltará revisar). Por tanto, suponiendo que trabajemos en media 3 horas diarias en el proyecto le habremos dedicado al final un total de 342 horas que es un poco más de las 300 horas que se supone que le tenemos que dedicar a este trabajo de fin de máster.

Este diagrama de Gantt ha sido resumido para que la imagen cuadrase en esta memoria. Un diagrama de Gantt y otro de actividades o PERT se encuentran el los ficheros planificacion.jpg y actividades.pdf respectivamente en la carpeta material_complementario. En ellos se puede ver de forma un poco más detallada la dependencia entre tareas, los hitos, las tareas críticas y los recursos personales y físicos asignados al proyecto.

1.3.2. Estimación de costes

A continuación se llevará a cabo la estimación de costes de gastos de personal, de materiales y de servicios y suministros necesarios para abordar el proyecto.

1.3.2.1. Costes de personal

De acuerdo con **[7]** el sueldo medio de un analista de datos en España ronda los 24000 € netos anuales. Para cobrar esa cantidad anual el sueldo neto mensual debería ser de 2000 €. Sin embargo, eso es el sueldo neto libre de deducciones de IRPF y seguridad social, por lo que el sueldo bruto debe ser mayor. Con esta información, obtenemos el siguiente desglose de la nómina mensual de un analista de datos que cobra aproximadamente 2000€ mensuales mediante la calculadora de sueldo neto del periódico EL PAÍS [8] que nos permite obtener el sueldo neto y sus retenciones teniendo en cuenta la situación laboral y familiar del empleado y el IRPF actual en España.

- Salario bruto mensual (SBM): 3466,67€.
- Deducciones:
  - Seguridad Social (6 % del SBM) = 160€
  - IRPF (17,19 % del SBM) = 458,40€
  - Total: 618,40€
- Cuota Patronal (30 % del SBM) = 800€.
- Nómina neta mensual = 2048,27€.

Vemos que para poder cobrar aproximadamente 2000€ mensuales es necesario tener un sueldo bruto de 2666,67€. Por tanto, considerado que el proyecto dura 9 meses, que es llevado a cabo únicamente por un analista y teniendo en cuenta la cuota patronal mensual el coste total será aproximadamente de 38400€.

1.3.2.2. Costes materiales

Todo el proyecto se desarrollará con un ordenador de escritorio valorado (incluyendo periféricos) en 2500€ con un tarjeta gráfica de última generación para poder acelerar el entrenamiento de los modelos de aprendizaje profundo. Por tanto si el proyecto dura 9 meses y los ordenadores de escritorio se amortizan en unos 48 meses, en este proyecto se tendrá que pagar una parte de su precio total proporcional a su tiempo de uso en el proyecto. Esta parte es 9 meses / 48 meses = 0,1875, de manera que el coste total aproximado del material necesario para el proyecto es de 2500€ x 0,1875 = 468,75€.

1.3.2.3. Costes de prestación de servicios y suministros

Finalmente vamos a desglosar los costes de luz, agua e internet que deberemos afrontar para el desarrollo del proyecto. Debido a que el proyecto se desarrollará íntegramente en un domicilio particular se proporcionan los precios mensuales reales que se están pagando actualmente por cada uno de estos servicios.

- Servicio de internet y teléfono de 400 Mb con jazztel: 45€/mes x 9 meses = 405 €.
- Luz: 55€/mes x 9 meses = 495 €.
- Agua: 20€/mes x 9 meses = 180 €.

Sumando los costes de cada servicio obtenemos un coste total de 1080€. Sin embargo, todos estos servicios no van dirigidos únicamente al proyecto. De esta forma, puesto que el proyecto durará un total de 114 días y le dedicaremos unas 3 horas diarias (como vimos en la sección 1.3.1) tenemos un total de 342 horas que con respecto a las 2736 horas totales de todos esos días supone un 0,125, que es la proporción del tiempo total del proyecto en la que estos tres servicios van dedicados al mismo. Por lo tanto, el coste total final aproximado de estos servicios es de 1080€ X 0,125 = 135€.

1.3.2.4. Resumen costes

Finalmente, en la siguiente tabla podemos ver la estimación de los costes de cada una de las anteriores tres categorías y el coste total.

Categoría coste	*Precio*
Personal	38400€
Material	468,75€
Servicios y suministros	135€
Total	**39003,75€**

Tabla 1.1: Resumen de los costes estimados del proyecto.

En la tabla 1.1 podemos ver que el coste estimado final del proyecto será de unos 39003,75€.

1.4. Trabajos relacionados

Para predecir el consumo energético en edificios o en cualquier otro ámbito, ya sea el consumo de energía eléctrica destinada a iluminación, a calefacción, a aire acondicionado, etc, existen multitud de técnicas presentes en la literatura, las cuales se pueden dividir en dos categorías principales [**1**]:

- Modelos basados en principios físicos: Típicamente consisten en crear un modelo de simulación del edificio que capture con la suficiente fidelidad sus propiedades físicas y funcionamiento energético y utilizarlo para predecir el consumo energético del edificio en el futuro bajo unas determinadas condiciones. De estos modelos ya hemos hablado en el apartado 1.1 y hemos comentado sus principales inconvenientes.
- Modelos estadísticos y de aprendizaje automático: Consisten en la creación de modelos de predicción que a partir de los valores pasados o históricos de consumo energético y otras variables sean capaces de predecir el consumo energético en el futuro.

Esta última categoría se divide a su vez en dos subcategorías:

- Modelos puramente estadísticos: Consisten en obtener una ecuación a partir de los valores históricos de consumo de energía que capture las dependencias temporales entre dichos valores y utilizar esta ecuación para predecir los valores futuros de consumo energético. En esta categoría destacan las técnicas ARIMA [**9**] o modelos autorregresivos integrados de media móvil y sus variantes como ARIMAX que para realizar la predicción utiliza datos históricos de otras variables además de los de consumo de energía.
- Modelos de aprendizaje automático: Las técnicas de aprendizaje automático permiten crear modelos de predicción del consumo energético

a partir de valores históricos de varias variables. Estos modelos buscan aproximar una función f que permita relacionar los valores de entrada del modelo (valores históricos) con los valores de salida (valores de consumo de energía en el futuro). A su vez, estos modelos se pueden dividir en otras dos subcategorías:

- Modelos de aprendizaje automático más tradicionales como SVM [**10**], XGBoost [**11**], Random forest [**12**], etc.
- Modelos de aprendizaje profundo, los cuales son modelos de redes neuronales artificiales que en los últimos años están ofreciendo cada vez mejores resultados que el resto de técnicas en varios problemas de predicción energética, como el preceptrón multicapa, las redes neuronales recurrentes, las redes neuronales convolucionales, etc.

Pueden encontrarse una gran variedad de trabajos de investigación en los que se han aplicado técnicas como ARIMA para predecir el consumo energético. Así, en [**13**] se utilizaron modelos ARIMA y SARIMA (ARIMA estacional) para predecir la demanda de energía primaria en Turquía desde 2005 hasta 2020. Un modelo ARIMA también fue utilizado en [**14**] para estimar el consumo de energía eléctrica doméstica mensual en una provincia del Este de Arabia Saudí. Por otra parte, en [**15**] compararon los resultados de un modelo tradicional ARIMA univariable con métodos de regresión lineal (técnicas de aprendizaje automático tradicionales) a la hora de predecir el consumo de energía a corto plazo teniendo en cuanta la relación entre las condiciones meteorológicas y el consumo energético.

También podemos encontrar una gran cantidad de trabajos que emplean métodos de aprendizaje automático tradicionales, pues en [**16**] compararon el rendimiento de una red neuronal con un modelo de regresión lineal tradicional para predecir el pico de consumo de energía eléctrica regional en Taiwan, utilizando como datos históricos los valores de consumo eléctrico en cada región de Taiwan entre los años 1981 y 2000. Este trabajo demostró que la red neuronal cometía un error mucho menor que el modelo de regresión. En [**17**] se utilizó un modelo de regresión lineal múltiple para predecir el consumo de electricidad en Nueva Zelanda a partir de variables como el producto interior bruto, la cantidad de población y el precio de la electricidad. En [**18**] se aplicó un modelo de SVM para predicción del consumo eléctrico, demostrando ser superior a otros métodos de regresión lineal. En [**19**] se comparó la habilidad de tres métodos para predecir el consumo de energía en Grecia a largo plazo: Redes neuronales, regresión lineal y SVM. Los resultados indicaron que los modelos de red neuronal y SVM obtuvieron una gran precisión en sus estimaciones.

Todos los trabajos hasta ahora mencionados son más antiguos y emplean modelos estadísticos y métodos de aprendizaje automático tradicionales. En

trabajos más recientes se suelen emplear técnicas de aprendizaje profundo a la hora de predecir el consumo de energía. De esta forma, en [**1**] se compara el rendimiento de una red neuronal recurrente con capas LSTM frente a una red neuronal con arquitectura sequence to sequence (Seq2Seq) a la hora de predecir el consumo de energía eléctrica sobre un conjunto de datos con una hora de resolución y sobre otro con un minuto. Los resultados mostraron que ambos modelos presentaban un gran rendimiento a la hora de predecir el consumo con una hora de resolución, mientras que solo la red neuronal Seq2Seq obtenía buenos resultados para predecir el consumo con un minuto de resolución. Por otra parte, [**20**] propone una arquitectura de red neuronal profunda novedosa para predecir el consumo eléctrico horario de una ciudad del norte de China. En concreto, la red utiliza distintos componentes de redes neuronales profundas como capas convolucionales, capas recurrentes y capas totalmente conectadas, con cada una de las cuales trata datos históricos de distinta naturaleza demostrando unos errores en la predicción muy bajos en comparación con otras técnicas como SVM y regresión lineal. Por último, en [**21**] se utilizaron redes neuronales recurrentes para predicción horaria del consumo de energía eléctrica de un edificio cuando no se tienen apenas datos pasados de consumo. En este trabajo las redes neuronales recurrentes demostraron un gran rendimiento cuando solo hay información meteorológica disponible para abordar la predicción.

Como hemos podido observar, las técnicas de aprendizaje automático son las que más se vienen empleando últimamente a la hora de predecir el consumo de energía en edificios, barrios, ciudades, países, etc siendo también las que están aportando resultados más prometedores. De ahí, que en este proyecto optemos por aplicar principalmente estas técnicas para afrontar nuestro problema de predicción.

1.5. Estructura de la memoria

La memoria se ha organizado en 5 capítulos sin contar este capítulo introductorio, en los cuales se abordan con más detalle cada una de las tareas citadas en la sección 1.2 necesarias para alcanzar nuestro objetivo. Estos capítulos son los siguientes:

1. Métodos: En este capítulo se abordan desde un punto de vista teórico los distintos conceptos, métodos y tecnologías empleados para llevar a cabo las tareas planteadas en la sección 1.2.

2. Datos: En este capítulo se describe el edificio ICPE y las características generales de sus sensores, para acabar profundizando en las características de los sensores de una zona piloto del mismo y de otras variables adicionales con las cuales vamos a trabajar.

3. Análisis y preprocesamiento: En este capítulo se describirán las tareas (y sus resultados) de obtención de datos, análisis exploratorio y preprocesamiento llevadas a cabo con el propósito de adecuar los datos del edificio ICPE para poder aprender con ellos modelos de predicción y mejorar la calidad de los mismos.

4. Modelos de predicción: En este capítulo se describe el problema de predicción a resolver y su división en tres subproblemas distintos, tras lo cual veremos el proceso de experimentación llevado a cabo para la obtención de distintos modelos de predicción para abordar cada problema y sus resultados.

5. Conclusiones y trabajo futuro: En este capítulo se exponen las conclusiones finales en relación con el principal objetivo planteado en este proyecto extraídas tras la realización de las tareas de adquisición de datos, preprocesamiento y experimentación con distintos modelos de predicción. Tras lo cual se comentan las posibles vías de investigación y de trabajo futuro que se pueden seguir para tratar de profundizar y mejorar distintos aspectos de este proyecto.

Capítulo 2

Métodos

En este capítulo se abordan desde un punto de vista teórico los distintos conceptos, métodos y tecnologías empleados para llevar a cabo las tareas planteadas en la sección 1.2

Antes de ver de una forma más concreta y práctica las tareas que se han llevado a cabo para poder aprender los distintos modelos de predicción del consumo de calefacción del edificio ICPE y los resultados de los mismos, es necesario explicar brevemente los conceptos básicos, métodos, tecnologías, etc, empleados durante la realización de las mismas.

El contenido de este capitulo se estructurará en seis secciones. En la sección 2.1 se abordarán los conceptos básicos sobre series temporales, pues los datos del edificio ICPE son series temporales y estas tienen diferencias con respecto a los conjuntos de datos tradicionales. En la sección 2.2 abordaremos el aspecto teórico detrás de las acciones que se realizarán para conocer más en profundidad los datos con los que vamos a trabajar. En la sección 2.3 se explicarán las distintas tareas de preprocesamiento que se llevarán a cabo sobre los datos con el propósito principal de adecuarlos para que a partir de ellos podamos aprender modelos de aprendizaje automático. En la sección 2.4 se describirán los conceptos básicos acerca de las redes neuronales y de las distintas capas y arquitecturas que se explorarán en este trabajo. En la sección 2.5 veremos como aplicar una técnica de aprendizaje automático más tradicional como XGBoost para predecir el consumo de calefacción. En la sección 2.6 se describirán de una manera esquemática las diferentes tareas que desempeñaremos para abordar el problema así como las principales tecnologías empleadas para ello.

2.1. Series Temporales

Una serie temporal es una sucesión de observaciones o_1, o_2, o_3, ... o_n cada una de las cuales ha sido realizada en un instante de tiempo concreto t_1, t_2, t_3, ... t_n. Cada serie temporal se caracteriza por las siguientes propiedades:

- Número de variables: Una serie temporal puede ser univariable o multivariable. Si es univariable solo tendremos un valor en cada observación (una sola columna o variable). En cambio, si es multivariable, tendremos k valores (columnas o variables) en cada observación.
- Regularidad: Una serie temporal puede ser regular o irregular. Si es regular todas sus observaciones están separadas por intervalos de tiempo iguales. En caso contrario, estamos ante una serie temporal irregular.

En la siguiente tabla podemos ver un ejemplo de serie temporal multivariable, en la cual vemos las distintas observaciones en las filas y las variables en las columnas.

Intante de tiempo	*Polución*	*Presión*	*Temperatura*
2016-12-31T22:00Z	3.34564	11.37223	-4.00000
2016-12-31T22:15Z	3.37541	11.36582	-4.03750
2016-12-31T22:30Z	3.67834	11.35768	-4.06250
2016-12-31T22:45Z	3.69237	11.35111	-4.08750
2016-12-31T23:15Z	3.45785	11.33675	-4.10000
2016-12-31T23:30Z	3.12034	11.32907	-4.10000
2016-12-31T23:45Z	3.9873	11.32299	-4.10000
2017-01-01T00:15Z	4.0123	11.31233	-3.91250

Tabla 2.1: Ejemplo de serie temporal multivariable.

En la tabla 2.1 vemos que la serie temporal es multivariable, pues tiene 3 variables o columnas (*Polución*, *Presión* y *Temperatura*) y regular, ya que si observamos la columna *Instante de tiempo* la separación temporal entre cada observación es siempre de 15 minutos.

El conjunto de datos del que partiremos en este proyecto es una serie temporal multivariable regular, tal y como veremos más adelante.

A diferencia de los conjuntos de datos tradicionales, en una serie temporal hay una variable implícita adicional al resto, el tiempo. Esto hará que la manera de tratar nuestros conjuntos de datos sea distinta a la forma en la que trabajaríamos con conjuntos de datos tradicionales. La principal razón de esto es la relación temporal existente entre las distintas observaciones de una serie temporal, por la cual, una observación cualquiera o_i está estrechamente relacionada con las observaciones o_{i-1}, o_{i-2}, o_{i-3}...

Por tanto, a la hora de tratar con series temporales es necesario utilizar métodos que nos permitan conocer con profundidad sus características. Para

ello, existen dos enfoques distintos: la descomposición y la autocorrelación, los cuales se aplican por separado a cada variable (columna) de una serie temporal multivariable como si de una serie temporal univariable se tratase.

2.1.1. Descomposición

Las variables de una serie temporal multivariable pueden albergar una enorme variedad de patrones y para su análisis puede ser útil descomponerlas en las componentes individuales que las forman. Este proceso de descomposición consiste típicamente en extraer de la serie temporal otras tres series temporales, cada una de las cuales representa un componente básico de la original. Estas componentes son las siguientes:

- Tendencia: Expresa la progresión a largo plazo de las observaciones de la serie temporal. En otras palabras, mide si la media de las observaciones crece, decrece o se mantiene constante a largo plazo.
- Estacionalidad: Refleja los desplazamientos (ascensos y descensos) de las observaciones de la serie temporal que se repiten sistemáticamente dentro de un periodo de tiempo constante. En otras palabras mide los patrones que se repiten cada día, cada mes, cada cierto número de meses, etc. La estacionalidad es siempre de un período fijo y bien conocido.
- Irregularidad: Es todo lo que queda de la serie temporal después de quitarle las componentes de tendencia y estacionalidad. Típicamente, se corresponde con patrones aleatorios aunque no siempre tiene porque ser así, pudiendo mostrar patrones y ciclos de periodo impredecible.

En la siguiente figura podemos ver un ejemplo de descomposición de una serie temporal.

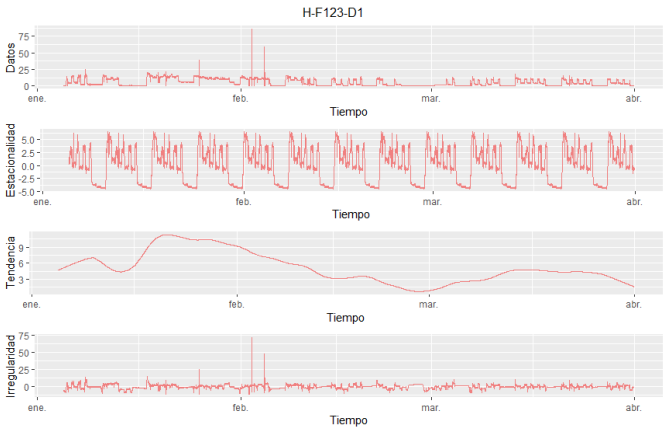


Figura 2.1: Ejemplo de descomposición de una serie temporal

En la figura 2.1 podemos ver de manera visual la descomposición de una serie temporal (gráfica de arriba) en sus componentes de estacionalidad (segunda gráfica desde arriba), tendencia (tercera gráfica desde arriba) e irregularidad (última gráfica desde arriba).

Hay distintas técnicas para descomponer una serie temporal, de entre las cuales la más comúnmente conocida es la de medias móviles. Sin embargo, hay métodos más robustos y versátiles como la descomposición STL (Seasonal and trend decomposition using losses) [**22**]. Esta última será la que utilicemos a la hora de analizar series temporales.

El conocimiento de las componentes de tendencia y estacionalidad de una serie temporal puede ser muy útil para su análisis, pues a partir de ellas podemos saber como se relaciona cada observación con las observaciones pasadas y futuras, predecir las observaciones futuras, hacia que tienden estas, etc.

2.1.2. Autocorrelación

La autocorrelación es una medida estadística que representa el grado de similitud de una serie temporal consigo misma y más concretamente con otras versiones «adelantadas» o «atrasadas» de si misma. Más concretamente, el cálculo de la autocorrelación es similar al cálculo de la correlación (en la sección 2.3.5.1.1 se explicará más detalladamente como se calcula la correlación) entre dos series temporales distintas, con la diferencia de que las dos series temporales son la misma y una de ellas está .ªdelantada.º .ªtrasadaünos instantes de tiempo con respecto a la otra.

La idea original de medir la autocorrelación, es que la predicción (y la imputación) de una serie temporal es posible porque normalmente sus observaciones del futuro están relacionadas o dependen de las pasadas. Esta relación entre observaciones del presente y del pasado es lo que mide precisamente la autocorrelación. Así, si obtenemos valores elevados de autocorrelación significa que los valores del futuro dependen en gran medida de los del pasado y por tanto se puede predecir la serie temporal a partir de ellos.

Al calcular la autocorrelación de una serie temporal, se obtienen varios valores de correlación, ñlps cuales, varían entre 1 y -1, indicando un valor de 1 una relación perfecta positiva, un valor de -1 una relación perfecta negativa y un valor de 0 que no existe relación alguna.

En la siguiente figura podemos ver un ejemplo visual de autocorrelación de una serie temporal.

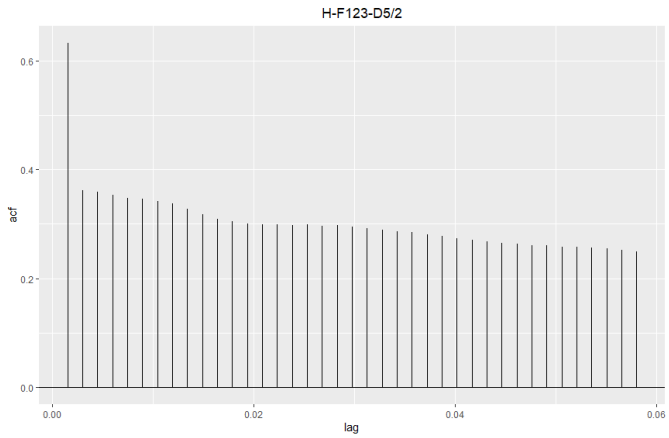


Figura 2.2: Ejemplo autocorrelación de una serie temporal.

En la figura 2.2 podemos ver gráficamente la autocorrelación de una serie temporal. En el eje x vemos los distintos instantes de tiempo o lags en los que se ha atrasado una de las dos versiones de la serie temporal (la otra permanece inmutable) y en el eje y se observa el nivel de correlación alcanzado para cada lag..

2.2. Análisis exploratorio de datos

El análisis exploratorio de datos (EDA) [**23**] es un proceso aplicado típicamente sobre un conjunto de datos con el objetivo principal de mejorar el conocimiento y comprensión acerca de los mismos y de extraer alguna hipótesis inicial sobre los patrones que podemos extraer de ellos. Más en concreto, el análisis exploratorio de los datos es útil para:

1. Mejorar la comprensión de cada una de las variables del conjunto de datos.

2. Formular hipótesis tempranas acerca de las posibles relaciones entre las distintas variables.

3. Determinar las herramientas y técnicas a aplicar sobre un el conjunto de datos en la fase de preprocesamiento.

Algunos autores consideran que el análsis exploratorio de los datos es una fase temprana del preprocesamiento de los mismos, mientras que otros consideran que son procesos independientes, siendo el análisis exploratorio el que precede al preprocesamiento. En nuestro caso, hemos optado por la segunda definición para separar y organizar mejor las distintas tareas de ambos procesos.

El análisis exploratorio de los datos engloba un amplio conjunto de métodos, de los cuales en nuestro caso aplicaremos tres:

- Cambio de nombre de las variables: No es en sí un método de análisis exploratorio, pero es una tarea necesaria para poder llevar a cabo las siguientes. Consiste en modificar el nombre de cada una de las variables de nuestro conjunto de datos por uno más intuitivo y legible, con el objetivo de facilitar las tareas de análisis posteriores.
- Resumen estadístico de las variables: Consiste en obtener una serie de medidas estadísticas de cada variables de un conjunto de datos. Esto nos permite obtener una abstracción de los valores de cada variable, facilitándonos su análisis.
- Visualización: Consiste en visualizar de manera gráfica los valores de cada una de las variables. De esta forma, se consigue analizar de manera gráfica e intuitiva los patrones de cada variable.

2.3. Preprocesamiento

Desde el punto de vista del aprendizaje automático el preprocesamiento es un proceso consistente en la aplicación de una serie herramientas, procesos, metodologías, etc sobre un conjunto de datos con el fin de que dichos datos puedan ser utilizados para para aprender modelos de predicción y mejorar la calidad del entrenamiento de los mismos.

Al igual que ocurre con el análisis exploratorio, el preprocesamiento engloba un amplio conjunto de herramientas, tareas, métodos, etc, y en nuestro caso vamos a llevar a cabo, un total de cuatro tareas de preprocesamiento: Transoformación de los datos, visualización de los datos transformados, tratamiento de outliers, tratamiento de valores perdidos y selección de variables. A continuación se describen de forma más detallada cada una de estas tareas.

2.3.1. Agregación, modificación y eliminación de datos

Esta primera tarea de preprocesamiento consiste en llevar a cabo una serie de tranformaciones a nivel de variables (columna) y observación (fila) sobre un conjunto de datos. Estas transformaciones consisten básicamente en eliminar y agregar variables y en modificar sus valores.

En nuestro caso en esta tarea se llevarán a cabo las siguientes subtareas:

1. Eliminación tanto de variables como de observaciones que no aportan información útil. Por ejemplo, una variable cuyos valores sean 0.
2. Agregación de dos variables distintas dando lugar a una totalmente nueva y más util que las otras dos por separado.

3. Cambio de unidad de los valores de varias variables.

2.3.2. Visualización de los datos transformados

Esta tarea consiste en la visualización de los datos transformados en la tarea anterior, pues estos pueden contener conocimiento que pudo no haber sido extraído durante la fase de análisis exploratorio. En nuestro caso particular, como trabajamos con series temporales, se han empleado la descomposición y autocorrelación (secciones 2.1.1 y 2.1.2 para llevar a cabo esta tarea.

2.3.3. Tratamiento de outliers

Los outliers o valores atípicos son valores de una variable que representan cambios abruptos que no concuerdan con el resto. Estos son valores o secuencias de valores demasiado elevados o demasiado pequeños en comparación con los demás y pueden deberse al ruido presente en los datos, a fallos en la recolección de los datos, etc. También pueden ser datos correctos que simplemente se salen de la norma.

2.3.3.1. Importancia de los outliers

Si una variable presenta observaciones anormalmente elevadas o reducidas y no se tratan adecuadamente pueden llegar a dominar las predicciones de los modelos de aprendizaje automático que aprendamos a partir de ellas. Estos modelos, durante su entrenamiento suelen darle una mayor importancia a los valores muy elevados o reducidos. En consecuencia estamos dando una mayor importancia a unos pocos valores en lugar de al resto de valores que conforman la mayoría de observaciones normales o típicas de la variable. Por tanto, esto puede afectar a los parámetros aprendidos por el modelo y empeorar las predicciones obtenidas.

En la figura 2.3 podemos ver un ejemplo visual de outliers (círculos de color azul) en los valores de una variable.

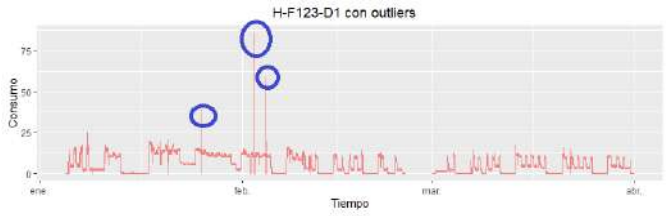


Figura 2.3: Ejemplo de outliers en una variable (serie temporal).

2.3.3.2. Detección de outliers

Para tratar los outliers en una serie temporal, es necesario identificarlos en primer lugar. Existen métodos muy diversos para identificar puntos outlier individuales en una serie temporal. Dichos métodos se pueden clasificar en cuatro grandes grupos [**24**]:

- Métodos basados en modelos de predicción: Consisten en averiguar la probabilidad de que una observación concreta de la serie temporal sea outlier mediante un modelo de predicción. Mas en concreto, con estos métodos se ajusta un modelo para predecir el valor de una observación de una serie temporal concreta en el instante de tiempo t, la cual se compara con el valor verdadero de la serie temporal en dicho instante. De esta forma si la diferencia es muy elevada es bastante probable que es punto sea outlier, pues el modelo se entrena para predecir los patrones regulares de la serie temporal. En este sentido hay trabajos que han utilizado modelos de perceptrón multicapa [**25**], máquina de soporte de vectores (SVM) [**26**], modelos ARIMA [**27**], etc.

- Métodos basados en similitud de perfil: Consisten en mantener un perfil regular de la serie temporal, el cual contiene únicamente los patrones regulares de la misma. Entonces, dada una observación en el instante de tiempo t, se compara esta con el perfil para decidir si es o no un punto outlier. En [**28**] se utiliza un perfil y un vector de varianzas de la serie temporal. Así, se compara una observación en el instante de tiempo t con el perfil y el vector y se calcula una puntuación para dicha observación, la cual cuanto más elevada sea indicará que dicha observación tendrá más probabilidad de ser un outlier. En [**29**] se utiliza una red neuronal para construir el perfil.

- Métodos basados en desvíos (deviants): Consisten en la detección de puntos deviants en una serie temporal los cuales si son eliminados dan lugar a una serie temporal con una menor desviación, es decir, más regular. En [**30**] proporciona una solución que emplea programación dinámica.

- Métodos basados en umbral: Consisten, en general, en establecer dos umbrales: uno mínimo y otro máximo. De forma que todas las observaciones de la serie temporal que estén por debajo del umbral mínimo o por encima del máximo serán considerados outliers. Así, por ejemplo, en [**31**] crean un umbral multiplicando la media de la serie temporal por un valor que es un múltiplo de su desviación típica.

2.3.3.3. Estrategias para el tratamiento de outliers

Una vez detectados los outliers en nuestras series temporales, hay que decidir qué hacer con ellos. Como dice [**24**] hay dos maneras de tratar los outliers en una serie temporal:

- Sustituirlos por valores más regulares. De esta forma conseguimos una serie temporal sin anomalías, aunque eso sí, perdemos información, pues como dijimos antes los outliers a pesar de ser anomalías pueden ser valores auténticos.
- Modelar la serie temporal teniendo en cuenta los outliers y sus efectos.

2.3.4. Tratamiento de valores perdidos

Se dice que una variable tiene valores perdidos cuando no se conoce el valor de varias observaciones de la misma. El origen de estos valores perdidos puede ser de distinto tipo:

- Desconocimiento de los valores sin factores aleatorios. Por ejemplo, cuando un sensor deja de funcionar en un edificio porque están de obras.
- Variable no aplicable a una observación concreta. Por ejemplo, el color de pelo de una rana.
- Error de origen aleatorio:
  - Totalmente aleatorio. Por ejemplo, fallo de origen desconocido de un sensor.
  - Aleatorio condicionado. Por ejemplo, fallo de un sensor que falla más cuando llueve.

2.3.4.1. Estrategias para tratar valores perdidos

A la hora de manejar los valores perdidos de las distintas variables de un conjunto de datos se puede optar por dos estrategias principales:

- Eliminación de valores perdidos, pudiendo eliminar:
  - Variables que tengan una gran cantidad de valores perdidos (más del 40 % por ejemplo), pues es inviable trabajar con una variable si no conocemos la mayoría de sus observaciones.
  - Observaciones que contengan valores perdidos, siempre que dicha eliminación no nos deje con pocas observaciones restantes.

- Sustitución de los valores perdidos: Consistente en emplear métodos que permitan sustituir los valores perdidos de cada variable por valores aproximados. Esta estrategia es comúnmente conocida como imputación de valores perdidos. Eso sí, es importante tener en cuenta que los valores perdidos imputados no son los valores auténticos sino aproximaciones a los mismos que nos evitan tener que descartar variables u observaciones completas sin causar mucho impacto en las estadísticas generales de cada variable y por tanto en el aprendizaje de modelos predictivos. Dicho esto, existe una gran variedad de métodos de imputación de valores perdidos, los cuales se pueden dividir en dos grupos:

  - Métodos univariable: Estos imputan los valores de cada variable por separado. Típicamente sustituyen cada valor perdido por una medida estadística (media, mediana, moda, etc) de la propia variable.
  - Métodos multivariable: Para imputar los valores perdidos de una variable utilizan las relaciones o dependencias de esta con el resto de variables.

2.3.4.2. Imputación de series temporales

La gran mayoría de métodos de imputación univariable y multivariable están pensados para conjuntos de datos tradicionales y no para series temporales. Es por eso que la mayoría no emplean el carácter temporal de los datos para llevar a cabo la imputación.

Por tanto, para imputar valores perdidos en una serie temporal se necesitan métodos que tengan en cuenta el carácter temporal de los datos. De acuerdo con **[32]** estos métodos se pueden clasificar en:

- Métodos univariable: Aceptan como entrada una serie temporal univariable (en nuestro caso una variable o culumna de nuestra serie temporal multivariable) y llevan a cabo la imputación de los valores perdidos de la misma mediante predicciones a partir de las observaciones pasadas. Entre estos métodos destacan técnicas como StructTS **[33]** e interp **[34]**.

- Métodos multivariable: Aceptan como entrada una serie temporal multivariable e imputan los valores perdidos de cada una de las variables a partir de las relaciones y dependencias de la misma con el resto y en algunos casos también a partir de la relación de dicha variable consigo misma. Entre estos métodos destacan técnicas como Irmi **[35]**, Mice **[36]**, Mtsdi **[37]** y Amelia **[38]**.

En este trabajo intentaremos imputar los valores perdidos presentes en nuestras variables probando métodos de imputación de series temporales

univariable y multivariable como los que hemos visto. Los métodos concretos que probaremos y su explicación se dejan para el apartado en el que se explicará cómo hemos tratado los valores perdidos.

2.3.5. Selección de variables

La selección de variables es un proceso consistente en la reducción de la dimensionalidad de un conjunto de datos que tiene, entre otros, dos objetivos principales:

- Permitir que los algoritmos de aprendizaje utilizados posteriormente se ejecuten de forma más eficiente tanto en tiempo como en memoria.
- Mantener únicamente las variables más relevantes para predecir la variable objetivo, mejorando así la calidad del modelo obtenido.

2.3.5.1. Métodos de selección de variables

Existe una amplia variedad de métodos para reducir la dimensionalidad de un conjunto de datos, los cuales pueden pertenecer a tres enfoques diferentes:

- Wrapper o envoltorio: Consiste en crear varios modelos con distintos (ensayo/error) subconjuntos de variables y quedarnos con aquel subconjunto que nos proporciona un modelo de mejor calidad. El principal problema de este enfoque es que incluso con conjuntos de datos de pocas variables puede ser inviable desde un punto de vista computacional.
- Selección incluida: Hay algoritmos de aprendizaje que incluyen de forma implícita la selección de las variables más relevantes. Por ejemplo, árboles de decisión.
- Filtrado: Consiste en utilizar una técnica que nos permita reducir la dimensionalidad del conjunto de datos antes de entrenar con ellos el modelo predictivo.

En nuestro caso, para realizar la selección de variables vamos a llevar a cabo en primer lugar un análisis de la correlación entre las mismas, tras lo cual aplicaremos el algoritmo de aprendizaje automático XGBoost (Extreme Gradient Boosting) no para resolver nuestro problema de predicción, sino para obtener información adicional a la obtenida durante el análisis de correlación acerca de la importancia de cada variable a la hora de predecir las variables objetivo. Por tanto, vamos a utilizar un método de filtrado (análisis de correlaciones) y otro de selección incluida (algoritmo XGBoost) para llevar a cabo la selección de variables. A continuación vamos a explicar de más detalladamente estas dos técnicas para seleccionar las variables y el motivo de su elección.

2.3.5.1.1 Análisis de correlaciones

El análisis de correlaciones es un método de selección de variables a partir del estudio de la correlación existente entre las mismas. En este caso, la correlación entre dos variables es una medida estadística que mide el grado de similitud o dependencia entre las mismas. Así, la correlación entre dos variables cualesquiera a y b puede ser un valor:

- Cercano a 1: Indica que entre las variables a y b existe una relación positiva o directamente proporcional, lo que significa que cuando las observaciones de a aumentan también lo hacen las de b y viceversa.
- Cercano a -1: Indica que entre las variables a y b existe una relación negativa o inversamente proporcional, de manera que cuando las observaciones de a aumentan las de b disminuyen y viceversa.
- Cercano a 0: Indica que las variables a y b son independientes y que cambios en un sentido concreto de las observaciones de a no tienen porque darse en el mismo sentido en las observaciones de b.

¿Cómo nos puede ayudar el análisis de correlaciones a llevar a cabo la selección de variables? Pues este análisis nos puede ayudar a estudiar:

- La correlación entre cada variable predictora y la variable objetivo, de forma que aquellas variables predictoras que estén altamente correladas con la variable objetivo tanto en sentido positivo como negativo, son buenas variables predictoras ya que un cambio en sus observaciones normalmente llevan a cambios en el mismo sentido o en el contrario en las observaciones de la variable objetivo. Así, por ejemplo podemos establecer un umbral de correlación con la variable objetivo de manera que solo aquellas variables que superen el umbral serán elegidas para la predicción.
- La correlación entre las variables predictoras, de manera que las variables más correladas entre sí, tanto en sentido negativo como positivo están aportando la misma información al modelo de predicción. Por tanto, podriamos agrupar las variables predictoras según su correlación y establecer otro umbral, de forma que de aquellos grupos de variables en los que la correlación entre las mismas supere el umbral solo nos quedaremos con una variable, descartando el resto.

2.3.5.1.1.1 Coeficientes de correlación

A la hora de calcular la correlación entre dos variables, no existe una única manera sino que hay multitud de métodos estadísticos para hacerlo, los cuales comúnmente se denominan coeficientes de correlación. Así, por un

lado podemos encontrar los coeficientes de correlación más conocidos, como Pearson [**39**] que determina la relación lineal entre dos variables, Spearman [**40**] que determina relaciones monótonas entre dos variables y Kendall [**41**] que también determina relaciones monótonas entre dos variables. Por otro, encontramos coeficientes de correlación algo más complejos como MIC [**42**] que busca dependencias lineales y no lineales entre dos variables.

Todos estos coeficientes se pueden aplicar sobre variables continuas, como en nuestro caso, pero de todos ellos nos bastaría con utilizar un coeficiente capaz de extraer relaciones lineales entre las variables, ya que simplemente queremos ver si cuando las observaciones de una variable suben o bajan las observaciones correspondientes de la otra variable también lo hacen. Por tanto, podríamos elegir el coeficiente de correlación de Pearson el cual es el más utilizado en estos casos. Sin embargo, estos coeficientes mencionados suponen que las observaciones dentro de cada variable son independientes las unas de las otras, lo cual no es cierto en el caso de las series temporales. Es aquí donde entra en escena la medida de correlación que hemos acabado utilizando en este trabajo, la cual es conocida como correlación cruzada.

En la siguiente subsección vamos a explicar los conceptos básicos sobre la correlación cruzada y cómo calcularla desde su forma más básica hasta lo que se conoce como Time Lagged Cross Correlation.

2.3.5.1.1.2 Correlación cruzada

En procesamiento de señales es frecuente querer averiguar si dos señales distintas son similares, en otras palabras, comprobar si ambas señales están correladas o no.

Para ello se puede calcular la correlación cruzada entre las dos señales $x_1[n]$ y $x_2[n]$ (ambas con n elementos u observaciones) de la siguiente forma.

$$corr(x_1, x_2) = \sum_{i=0}^{n-1} x_1[i]x_2[i] \tag{2.1}$$

La anterior expresion obtiene la correlación entre $x_1[n]$ y $x_2[n]$ considerando que ambas son vectores de tamaño n y obteniendo el producto interior (inner product) entre ambas. El resultado de este producto puede ser:

- Positivo: Si las componentes i-ésimas de ambas señales tienden a tener el mismo signo obtendremos un valor positivo. De forma que cuanto mayor sea dicho valor más correladas estarán las señales, pues esto significa que cuando una señal aumenta su valor la otra también y cuando disminuye la otra también.
- Negativo: Si las señales tienden a tener valores contrarios en cada componente i-ésima se obtiene un valor negativo que indica que ambas variables son independientes (si el valor está cercano a 0) o inversamente proporcionales (si el valor es muy negativo).

Ahora bien, esta medida presenta los siguientes problemas:

1. Si tenemos dos señales A y B, ambas con los mismos signos en cada una de sus componentes pero B siendo de una magnitud mayor (valores más grandes) entonces la correlación entre A y B es mayor que la de A consigo mismo, lo cual no es aceptable para una medida de correlación.

2. El valor de la correlación entre dos señales dependerá del tamaño n de la muestra.

Para resolver estos problemas se suelen normalizar las señales y obtener el promedio del producto interior. Asi, la fórmula de la correlación sería la siguiente.

$$corr(\overline{x}_1, \overline{x}_2) = \frac{1}{N} \sum_{i=0}^{N-1} \overline{x}_1[i]\overline{x}_2[i] \tag{2.2}$$

Donde $\overline{x}_1$ y $\overline{x}_2$ son los vectores $x_1[n]$ y $x_2[n]$ normalizados, es decir, con sus valores centrados en 0 y con desviación típica 1.

Con esta fórmula se pueden obtener los siguientes resultados a la hora de calcular la correlación entre dos señales:

- Valor cercano a 0: Indica que ambas señales son independientes.

- Valor cercano a 1: Indica que ambas señales son altamente dependientes la una de la otra. Una señal consigo misma tiene correlación 1.

- Valor cercano a -1: Indica que ambas señales son inversamente proporcionales.

La ecuación 2.2 es la fórmula del coeficiente de correlación de Pearson anteriormente mencionado, el cual presupone que las observaciones dentro de cada una de ambas variables son independientes las unas de las otras.

Para resolver este problema se suele emplear la técnica conocida como Time Lagged Cross Correlation, que consiste en calcular el nivel de correlación entre dos señales con la ecuación 2.2 pero rotando la primera señal k timesteps hacia la derecha o la seguna k timesteps a la izquierda (es lo mismo). De esta forma la fórmula quedaría como sigue:

$$corr(\overline{x}_1, \overline{x}_2) = \frac{1}{N} \sum_{i=0}^{N-1} \overline{x}_1[i]\overline{x}_2[i+k] \tag{2.3}$$

Hasta ahora solo hemos dicho cómo calcular las correlación cruzada entre dos señales. Sin embargo, también se puede calcular de la misma forma entre dos series temporales, pues tanto las señales como las series temporales son variables de 1 dimensión dependientes del tiempo.

Así, al obtener la correlación cruzada entre dos series temporales, estamos viendo sus correlaciones en distintos desfases temporales de la una con respecto a la otra, con lo que también estamos teniendo en cuenta las dependencias temporales.

2.3.5.1.2 Algoritmo XGBoost

El algoritmo XGBoost (Extreme Gradient Boosting) [**11**] es un algoritmo de aprendizaje automático muy popular hoy en díaa tanto para tareas de clasificación como de regresión. Este ha demostrado, desde su creación, mejor rendimiento que muchos otros algoritmos de aprendizaje automático a la hora de trabajar sobre datos estructurados (conjuntos de datos).

Entre sus principales características destacan su velocidad y rendimiento, su capacidad de paralelización tanto por CPU como por GPU y su gran cantidad de parámetros modificables.

Dicho esto, se trata de un algoritmo de aprendizaje automático, lo que significa que su principal función es resolver problemas de predicción, en los cuales se parte típicamente de un conjunto de datos con k variables predictoras y una variable objetivo cuyos valores deseamos predecir o clasificar. Así, este algoritmo es capaz de aprender a partir de un conjunto de datos de entrenamiento un modelo que al recibir como entrada un vector con k valores, uno por cada variable predictora nos devolverá (estimará) el valor correspondiente de la variable objetivo.

Más concretamente, XGBoost pertenece a la familia de los algoritmos de boosting, que es una técnica secuencial consistente en que en el instante o iteración t se crea a partir del conjunto de datos de entrenamiento del que partimos un clasificador débil, que desde el punto de vista de un problema de clasificación binaria es un clasificador cuya precisión es ligeramente superior al 50 %, es decir, que es un poco mejor que un clasificador aleatorio. Entonces con dicho clasificador se etiquetan los individuos del conjunto de datos de entrenamiento y se les da mas peso a aquellos en los que falla al clasificarlos, buscando de esta forma que el clasificador creado en la siguiente iteración priorize clasificar esos individuos bien. Este proceso, se repite n veces dando lugar a n clasificadores débiles, los cuales finalmente se unen de forma ponderada dando lugar a un clasificador (ensemble) con un rendimiento bastante aceptable.

Entonces XGBoost se trata de un algoritmo de boosting que emplea árboles de regresión y clasificación como clasificadores débiles. Cada uno de estos árboles consta de una serie de nodos, que «cuelgan» de un primer nodo denominado raíz, en cada uno de los cuales los ejemplos del conjunto de entrenamiento se separan en dos grupos o más en base a una regla formulada sobre una variable concreta (por ejemplo: grupo de los individuos con altura superior a 1.89 m y grupo con los individuos de altura inferior a 1.89 m). De esta forma para clasificar o predecir, el árbol recibe un ejemplo (fila del

conjunto de datos) como entrada y en base a los valores de sus variables se va desplazando desde el nodo raíz (primer nodo) hacia los nodos hoja (nodos finales o sin hijos) en los cuales se le asocia una etiqueta (clasificación) o un valor real concreto (regresión). Durante la construcción de cada árbol el algoritmo XGBoost debe seleccionar en cada nodo la variable más relevante para separar el conjunto de entrenamiento para lo cual se pueden utilizar multitud de criterios. De esta manera, cuando se termina de ejecutar el algoritmo XGBoost las variables más relevantes para predecir la variable objetivo serán aquellas que más veces se han empleado para crear nodos en todos y cada uno de los árboles generados durante el entrenamiento.

Dicho esto, vamos a emplear este algoritmo para aprovechar su capacidad de detectar las variables predictoras más relevantes a la hora de predecir una variable objetivo.

2.4. Redes Neuronales

Como ya se mencionó en el apartado 1.2 el objetivo principal de este trabajo es tratar de predecir el consumo futuro de energía eléctrica destinada a la calefacción del edificio ICPE empleando para ello modelos de redes neuronales profundas, es decir, técnicas de aprendizaje profundo. Por ello, es necesario explicar brevemente la teoría subyacente a este tipo de modelos de predicción. De esta manera, en esta sección vamos a ver qué son las redes neuronales, sus componentes, las topologías o tipos de redes que hemos empleado en este trabajo y los conceptos básicos acerca del entrenamiento de las mismas.

2.4.1. Definición

De acuerdo con **[43]** las redes neuronales artificiales (RNA) son redes de elementos o unidades de procesamiento simples interconectadas entre sí mediante conexiones sinápticas, las cuales tienen un peso (fuerza) que se ajustan a partir de la experiencia (datos).

A las unidades de procesamiento simples de una RNA se les denomina comúnmente neuronas, las cuales reciben como entrada la salida (estímulo) de otras neuronas y producen una salida de acuerdo con su función de activación. De esta manera, tienen un funcionamiento inspirado en las neuronas biológicas humanas.

Un modelo de RNA concreto se caracteriza por una topología determinada, la cual establece el número de neuronas que posee y como estas se organizan e interconectan entre sí.

De esta forma, desde el punto de vista del aprendizaje supervisado (que es el que nos incumbe), las RNA proporcionan un modelo de cómputo paralelo y distribuido, capaz de aprender a partir de ejemplos (datos). Este aprendizaje a partir de ejemplos se logra mediante un algoritmo de aprendizaje, el cual

modifica los pesos de la red a medida que le vamos mostrando ejemplos.

2.4.2. Neuronas

Como hemos dicho en la sección 2.4.1, las RNA están formadas por un conjunto de neuronas interconectadas entre sí de una manera concreta. En la siguiente imagen podemos ver más formalmente el modelado computacional de una neurona artificial.

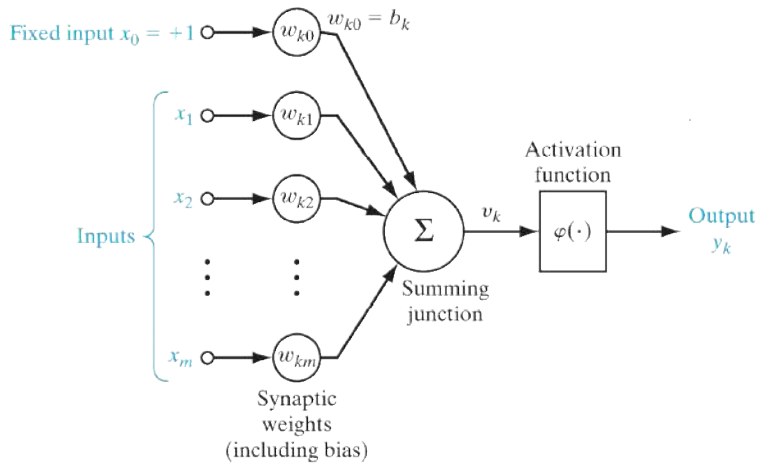


Figura 2.4: Modelo de neurona artificial [**44**].

En la figura 2.4 podemos ver cómo es el cálculo en una neurona artifical típica. Si la miramos de izquierda a derecha, vemos que recibe una serie de entradas (inputs) x_1, x_2, ..., x_m, cada una de las cuales es un valor numérico que se le proporciona directamente o que puede venir procedente de las salida (estímulo) de otra neurona. En este caso, cada entrada i vemos que está asociada a un peso concreto w_i, el cual es el peso de la conexión sináptica correspondiente a dicha entrada. Además de estas m entradas también vemos una entrada denotada como x_0, la cual está siempre fijada a 1 y asociada a un peso w_0. Esta entrada permite que durante el aprendizaje de la red neuronal, cada neurona aprenda no solo los pesos de sus entradas, sino también este peso w_0 que se corresponde con el sesgo b de la neurona, el cual determina directamente el umbral de su función de activación.

Vemos como todas estas entradas junto con sus pesos se agregan mediante una suma ponderada que sigue la siguiente fórmula.

$$v_k = \sum_{i=1}^{m} x_i w_i + b \tag{2.4}$$

Dando lugar así al valor v_k, que como vemos en la figura 2.4 es pasado como entrada a la función de activación de la neurona, la cual se denota

como $\varphi()$ y da como salida el valor y_k, el cual es el valor que da como salida la neurona y puede ir dirigido a otras neuronas.

Dicho esto, existen distintos tipos de funciones de activación siendo cada una más apropiada para unos problemas y menos para otros. Estas se pueden clasificar según el valor que devuelven como salida.

- Funciones de activación binarias: Dan 0 o 1 como salida, dependiendo el valor de v_k, de forma que si v_k está por encima del umbral establecido por el sesgo b de la neurona devuelve 1 y 0 en caso contrario. Se suelen utilizar en las neuronas de salida en problemas de clasificación.
- Funciones de activación de tipo sigmoidal: Dan como salida un valor real situado en un intervalo [min, max]. Hay muchas funciones de este tipo como la función logística, relu, tangente hiperbólica, la propia sigmoidal etc. Se suelen utilizar en las neuronas de las capas ocultas tanto en problemas de regresión como de clasificación y en las capas de salida en problemas de regresión.

Como nuestro problema es de regresión, utilizaremos funciones de activación sigmoidales en las neuronas de nuestros modelos de RNA.

2.4.3. Topologías

Como se dijo en la sección 2.4.1 la topología o arquitectura de una RNA viene determinada por su número de neuronas, la organización de las mismas y sus conexiones. De manera general, todas las redes neuronales se caracterizan por estar compuestas por una capa de entrada, encargada de recibir los datos, una o varias capas ocultas encargadas de procesar y encontrar patrones en dichos datos para clasificarlos o predecirlos y una capa de salida con una o varias neuronas encargadas de clasificar o predecir.

De acuerdo con nuestro problema (de predicción multistep, sección 1.2) todas nuestras redes deben de:

- Tener una capa de entrada que les permita recibir una matriz que contiene una serie temporal multivariable con los valores históricos o pasados de varias variables a partir de los cuales queremos predecir los siguientes n valores en el futuro de una variable objetivo.
- Tener una capa de salida con n neuronas, de manera que la neurona i será la encargada de predecir el valor i futuro de la variable objetivo.

Lo que verdaderamente caracteriza a una RNA son sus capas ocultas, pues hacen que tenga su arquitectura propia diferente del resto. En base a sus capas ocultas, las RNA se pueden clasificar en grandes grupos o familias de RNA. En esta subsección vamos a describir, de más simple a más complejo, los cuatro tipos de arquitectura que se han empleado en este trabajo para abordar el problema de predicción de consumo de calefacción.

2.4.3.1. Perceptrón multicapa

La primera arquitectura que utilizaremos en este trabajo para abordar el problema de predicción será la conocida como perceptrón multicapa (MLP), la cual es la arquitectura más clásica de las redes neuronales feed forward. En la siguiente imagen vemos un ejemplo de esta arquitectura.

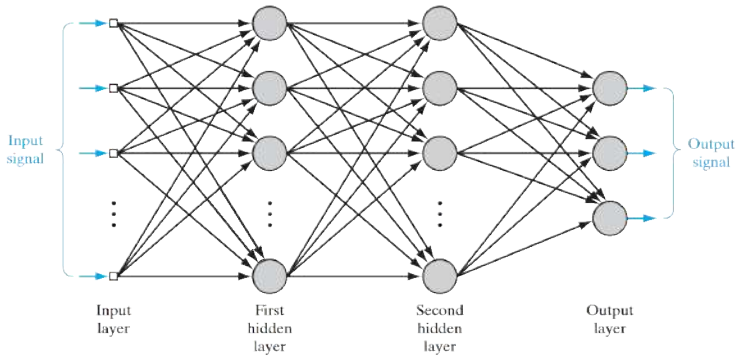


Figura 2.5: Grafo de la arquitectura de un perceptrón multicapa [**44**].

En la figura 2.5 podemos ver un grafo de ejemplo que describe de manera visual la arquitectura MLP. Vemos que está formado por una capa de entrada y otra de salida entre las cuales vemos dos capas ocultas. Como podemos observar, en los MLP una neurona pertenceciente a la capa k recibe como entrada la salida de todas y cada una de las neuronas de la capa k-1, es decir, la capa anterior, razón por la cual a estas capas se les denomina ”totalmente conectadas”. Dicho esto, se puede ver que los datos de entrada que recibe el MLP van viajando capa a capa desde la capa de entrada hasta la capa de salida ”hacia delante”(feed forward).

En este caso, un MLP puede estar formado por tantas capas ocultas como queramos, cada una de las cuales puede contener al mismo tiempo tantas neuronas como deseemos.

2.4.3.2. Redes neuronales recurrentes

De acuerdo con [**45**] una de las principales características de las redes neuronales convencionales como el perceptrón multicapa, es que no tienen memoria, es decir, cada entrada que le pasamos es procesada de forma independiente al resto de entradas, sin mantener ningún estado entre ellas. Esto hace que al procesar series temporales con estas redes, tengamos que codificarlas en un vector para que la red lo pueda procesar de una sola vez extrayendo dependencias entre los valores de la serie temporal de cara a realizar la predicción. Sin embargo, un MLP no es capaz de extraer las dependencias temporales entre los distintas observaciones (filas o instantes de tiempo) de una serie temporal, pues como hemos dicho procesa los datos de

la serie temporal de una sola vez, sin tener en cuenta el orden cronológico o temporal de las observaciones, lo cual como ya se dijo en el apartado 2.1 es muy importante a la hora de realizar predicción de series temporales.

Como solución a este problema surgen las redes neuronales recurrentes (RNN), las cuales pueden verse como capas totalmente conectadas pero con la salida de cada neurona conectada con su entrada de manera recurrente tal y como vemos en la figura 2.6,

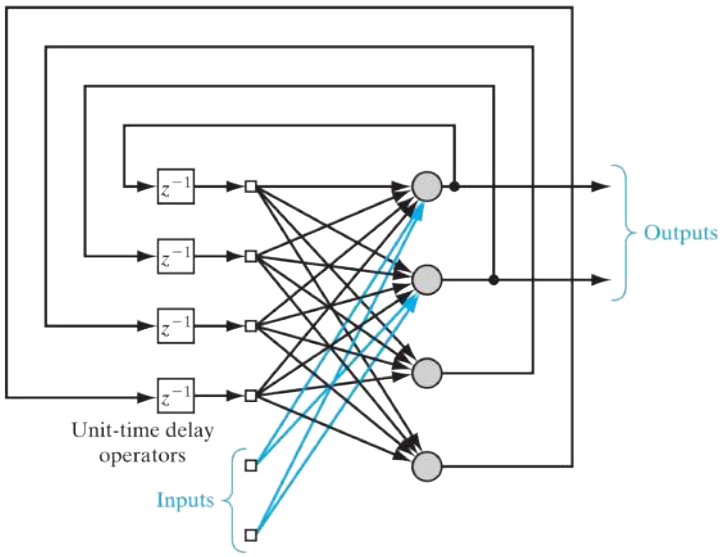


Figura 2.6: Topología de una red neuronal recurrente [**44**].

Ahora bien, ¿cómo procesa una RNN como la de la figura 2.7 una serie temporal (univariable o multivariable) de n observaciones? Pues bien, para ilustrarlo mejor podemos ver la RNN como un conjunto de n celdas como las de la figura 2.7.

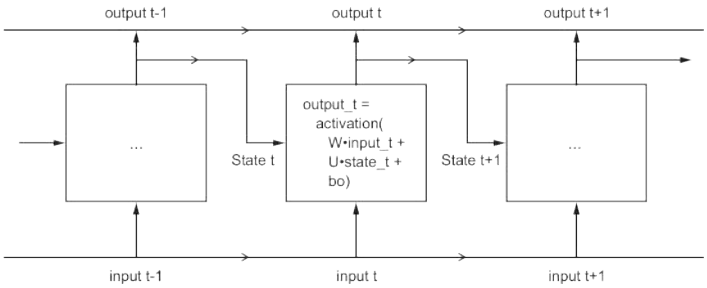


Figura 2.7: RNN desenrollada a lo largo del tiempo. [**45**].

En la figura 2.7 podemos ver una RNN como la de la figura 2.6 desenrro-

llada en forma de celdas. Cada celda simplemente ilustra el procesamiento por parte de la RNN de una observación concreta de la serie temporal. Por lo que si esta última consta de n observaciones la RNN tendrá n celdas, de manera que cada celda t:

1. Tiene m neuronas que denominaremos ordinarias y otras m que denominaremos recurrentes y recibe como entrada los valores de la observación o_t de la serie temporal.

2. Cada valor de o_t se pasa como entrada a todas y cada una de las m neuronas ordinarias.

3. La salida de todas las neuronas, tanto ordinarias como recurrentes de la celda anterior (t-1) se pasa como entrada a todas las neuronas de la celda t.

4. Cada neurona ordinaria agrega sus entradas y pasa el resultado a su función de activación obteniendo la salida correspondiente a la observación o_t, la cual es un vector de tamaño m que en la posición i contiene la salida de la neurona ordinaria i.

Con esto, vemos que la celda t genera una salida a partir de la observación o_t y de la información o conocimiento relativo al procesamiento en orden cronológico de las anteriores t-1 observaciones, siendo así una RNN capaz de tener en cuenta el orden cronológico y las dependencias temporales entre las distintas observaciones de la serie temporal a diferencia de la arquitectura MLP.

Todo lo que hemos explicado hace referencia a una RNN simple de una capa o a una capa recurrente, pues una RNN puede tener una o varias capas recurrentes como la descrita apiladas de manera secuencial una detrás de la otra. De esta forma, una capa recurrente B que tenga detrás otra capa recurrente A, recibirá como entrada una matriz de n filas y m columnas donde en la fila t estarán almacenadas la salida de cada una de las m neuronas ordinarias de la capa recurrente A han generado al procesar la observación o_t. Así se pueden apilar, una tras otra, tantas capas recurrentes como se desee.

Por último, una arquitectura RNN también tiene al igual que la MLP una capa de entrada y otra de salida, Entre las cuales no solo tiene por qué haber capas RNN sino que también pueden haber por ejemplo, capas totalmente conectadas situadas detrás de las capas recurrentes. La idea principal de esto es que las capas recurrentes procesen en primer lugar la seerie temporal de entrada en orden cronológico extrayendo sus características o dependencias temporales, las cuales se pasan a las capas totalmente conectadas para que realicen finalmente la predicción. Este tipo de arquitectura RNN será la que utilicemos en este trabajo tal y como veremos más adelante pues combinan

la capacidad de las capas recurrentes para extraer características temporales entre los datos y la capacidad de aprendizaje de las capas totalmente conectadas.

2.4.3.2.1 Capa LSTM

La capa recurrente clásica descrita anteriormente presenta un gran problema: aunque teóricamente debería ser capaz de retener a la hora de procesar la oservación o_t, la información de todos las observaciones procesadas con anterioridad, en la práctica esto no sucede. Esto se debe al problema del desvanecimiento del gradiente [**46**], el cual intuitivamente consiste en que la capa recurrente, a la hora de procesar la observación o_t no es capaz de retener en memoria la información de las observaciones que sean más antiguas que la observación t-n (para un n cualquiera) .

Para hacer frente a este problema se diseñó la capa LSTM (Long Short-Term Memory) [[**47**], la cual es una variante de la capa recurrente clásica anteriormente descrita que añade un mecanismo para llevar información a través de varias observaciones en el futuro. Se puede ver como si tuviéramos una cinta transportadora que nos permite llevar información intacta a lo largo de todos los instantes de tiempo, la cual es utilizada para calcular la salida de la capa en cada instante de tiempo t. En la figura 2.8 podemos ver las celdas de una capa LSTM.

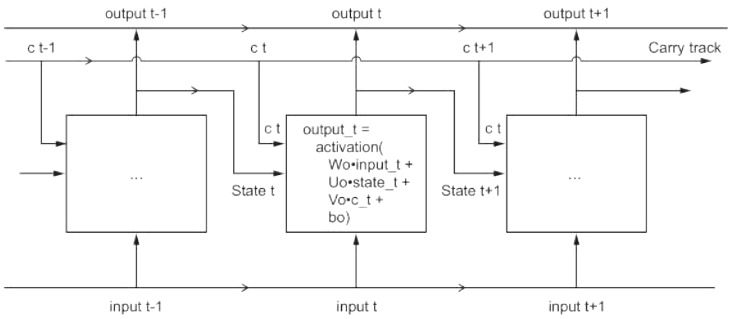


Figura 2.8: LSTM desenrollada a lo largo del tiempo [**45**].

En la figura 1.8 podemos ver que la celda de la LSTM que procesa la observación o_t es igual a la celda de una capa recurrente clásica, con la diferencia de que, como vemos, la celda de la LSTM ahora recibe también como entrada un valor c_t que es información que puede proceder de varios instantes de tiempo del pasado. Esta información se pasa como entrada tanto a las neuronas ordinarias como a las recurrentes de la celda influyendo por tanto en la generación del estado y de la salida de la misma. De esta manera, esta capa nos permite guardar de una forma más efectiva información de los primeros instantes de tiempo y utilizarla en instantes mucho más posteriores,

evitando así, el problema del desvanecimiento del gradiente.

Por estas ventajas, en este trabajo se han empleado capas recurrentes LSTM.

2.4.3.3. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) **[43]** **[45]** funcionan especialmente bien para resolver problemas de visión por computador, debido a que trabajan de manera convolucional extrayendo características relevantes de regiones locales de las imágenes permitiendo obtener una representación modular de las mismas y una mayor eficiencia en su procesamiento. Esta extracción de características relevantes de una imagen se realiza de la siguiente forma:

- Se toma una máscara de convolución 2D, que es una matriz de tamaño m x m siendo m un número impar, la cual contiene en cada posición (f,c) (f es la fila y c la columna) un valor numérico o peso.
- Se superpone la máscara sobre cada píxel p_{ij} (i es la fila y j la columna de la imagen en las que se encuentra el píxel) de la imagen de manera que este coincida con el centro de la misma. De esta forma, el valor de cada posición (f,c) de la máscara se multiplica por el valor del píxel de la imagen que le corresponde y se suman todas las multiplicaciones.
- El resultado de la suma se es el valor del píxel p_{ij} de la imagen que resulta de pasar la máscara de convolución sobre la imagen original.

La imagen resultante de este proceso de convolución se denomina mapa de características, el cual en su píxel p_{ij} almacena información relevante de la región local del píxel p_{ij} de la imagen original.

Las CNN constan de varias capas convolucionales seguidas de varias capas totalmente conectadas, de manera que las primeras aplican varias máscaras de convolución sobre una imagen de entrada para extraer varios mapas de características relevantes que facilitan en gran medida la tarea de clasificación o predicción a las capas totalmente conectadas.

Las mismas características que hacen a las CNN excelentes para la visión por computador también hacen que sean altamente utilizables para el procesamiento de series temporales. En este caso tratamos el tiempo de la serie temporal como una dimensión, al igual que la altura y anchura de una imagen 2D.

Estas redes neuronales convolucionales 1D pueden ser competitivas con las RNN en ciertas tareas de procesamiento de series temporales siendo además más eficientes computacionalmente, debido a que . De hecho, las CNN 1D se han aplicado con éxito en tareas como generación de audio y traducción.

2.4.3.3.1 Convolución 1D para series temporales

Las capas convolucionales 1D funcionan de la misma forma que las 2D, con la diferencia de que la máscara de convolución ahora es un vector de tamaño m (m siendo un número impar) en lugar de una matriz. De esta forma, una capa de convolución 1D aplica n máscaras de convolución de tamaño m sobre una serie temporal (suponemos aquí por simplicidad que la serie temporal es univariable, si fuese multivariable tendríamos además una máscara de convolución 1D por cada varaible) de entrada dando lugar a n mapas de características. Para entender mejor el proceso de aplicación de una máscara de convolución podemos observar la figura 2.9.

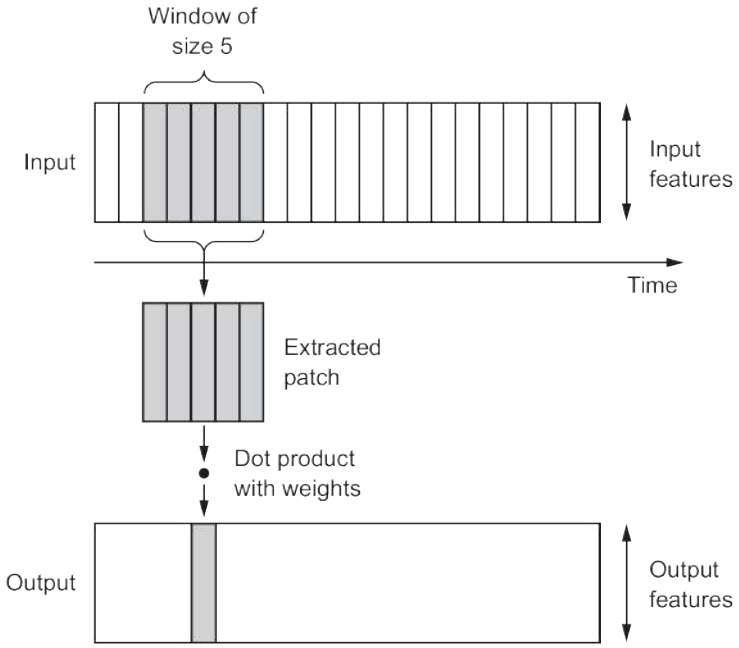


Figura 2.9: Ejemplo de convolución 1D [**45**]

En la figura 2.9 podemos ver un ejemplo de aplicación de una máscara de convolución 1D de tamaño 5 sobre una serie temporal univariable. Con esta figura vemos que lo que realmente hacen las capas de convolución 1D es reconocer patrones locales en una serie temporal, pues cada máscara de convolución 1D es aplicada a todas las regiones locales de la serie temporal («paseamos» la máscara por cada observación) lo cual permite que un patrón sea detectado en cualquier parte o región de la serie temporal. Por esto mismo se dice que las CNN 1D son invariantes a la traslación o traslación temporal en este caso.

2.4.3.3.2 Pooling para series temporales

Las operaciones o capas de pooling se utilizan el las CNN 2D para reducir el tamaño de los mapas de características resultantes de aplicar la convolución. En este caso, al igual que sucede con la convolución 2D, el pooling 2D también tiene su equivalente 1D, el cual consiste también en «pasear» una máscara de tamaño m (de tamaño par o impar) por las observaciones de la serie temporal (pero en este caso, la máscara se desplaza de m en m observaciones) y quedarnos en cada momento con la media (average pooling 1D) o el máximo (max pooling 1D) de los valores de la serie temporal que coinciden con la máscara.

Al igual que con las CNN 2D, el pooling se utiliza en las CNN 1D para reducir el tamaño de cada una de las secuencia de salida de una capa convolucional y así evitar el crecimiento exponencial de parámetros (pesos) de la red.

2.4.3.3.3 Combinación de CNN y RNN

Debido a que las CNN 1D procesan regiones locales de una serie temporal de manera independiente, no son sensibles al orden cronológico de las observaciones más allá de en una escala local. Debido a esto surge una estrategia [**45**] para combinar la ligereza y eficiencia de las CNN 1D con la sensibilidad al orden de las observaciones de una RNN en una misma arquitectura. Esta está formada al principio por una o varias capas convolucionales 1D que se encargan de extraer de manera automática patrones locales la serie temporal, seguidas de una o varias capas recurrentes encargadas de procesar los patrones locales extraídos en orden cronológico y finalmente tendríamos una o varias capas totalmente conectadas encargadas de procesar todas las características y dependencias extraídas en las anteriores capas y realizar la predicción a partir de ellas. Este será el tercer tipo de arquitectura de red neuronal que utilizaremos en este trabajo y la denotaremos como CNN.

2.4.3.4. Redes sequence to sequence

Por último, en este trabajo se empleará una arquitectura un poco más sofisticada para tratar de abordar el problema de predicción, la cual es conocida como sequence to sequence (Seq2Seq). Esta ha sido utilizada con éxito en reconocimiento de voz y de objetos y en predicción del consumo eléctrico [**48**], [**49**], [**50**].

Desde el punto de vista de nuestro problema de predicción, una arquitectura Seq2Seq está formada por dos partes bien diferenciadas. Por un lado está el codificador, el cual recibe como entrada una serie temporal con los valores históricos de consumo y la codifica en un vector de longitud fija (siempre tendrá el mismo tamaño sea cual sea la longitud de la serie temporal de entrada). Por otro lado, está el decodificador, que recibe el vector

codificado e información contextual de los instantes de tiempo futuros en los cuales queremos predecir el consumo de calefacción y devuelve como salida el consumo de calefacción predicho en los siguientes n instantes de tiempo.

La principal aportación novedosa de esta arquitectura con respecto a las demás es la utilización de información contextual futura para realizar la predicción, pues la tres arquitecturas mencionadas anteriormente solo utilizan valores históricos de consumo para llevar a cabo la predicción del consumo de calefacción, mientras que la arquitectura Seq2Seq puede utilizar por ejemplo las predicciones meteorológicas (por ejemplo temperatura) de los instantes de tiempo futuros para predecir en ellos el consumo de calefacción.

Más específicamente, la arquitectura Seq2Seq utiliza una capa LSTM como codificador y otra como decodificador. El codificador procesa observación a observación la serie temporal recibida como entrada produciendo como salida una codificación de la misma. Esta codificación es un vector que contiene el estado o salida producida por todas las neuronas de la capa LSTM tras procesar la última observación. Entonces, el vector de estados generado por el codificador se utiliza como estado inicial del decodificador y éste último empieza a procesar una serie temporal que contiene en la observación o_t uno o varios valores de variables contextuales (por ejemplo temperatura) correspondientes al instante de tiempo t del futuro en el que queremos predecir el consumo de calefacción. Tras procesar el decodificador en orden cronológico toda la serie temporal de información contextual produce una salida que es pasada a una capa totalmente conectada con n neuronas en la cual la neurona t da como salida el consumo de calefacción predicho instante de tiempo t. Esta descripción de la arquitectura Seq2Seq se puede ver en la figura 2.10.

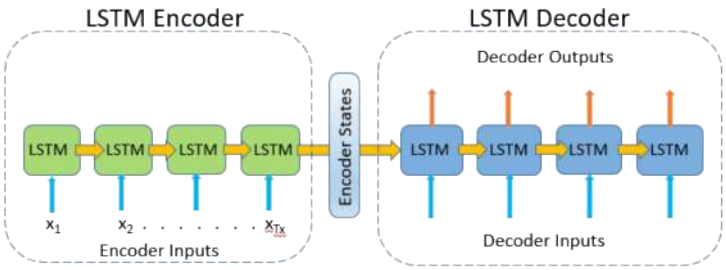


Figura 2.10: Arquitectura Seq2Seq. [**51**]

Por último cabe destacar varios aspectos acerca de esta topología.

- El número de neuronas la capa LSTM del codificador condiciona la longitud del vector codificado generado a partir de la serie temporal, pues el tamaño del vector será igual al número de neuronas de la capa LSTM.

- Podemos aumentar la potencia de procesamiento de la arquitectura añadiendo más capas LSTM tanto al codificador como al decodificador, teniendo en cuenta que ambos tienen que tener siempre las mismas capas LSTM y en cada capa el mismo número de neuronas ordinarias y recurrentes. De esta forma, la capa LSTM i del decodificador se inicializará con los estados dados como salida por la capa LSTM i del codificador.

- Como hemos dicho, el decodificador procesa observación a observación los valores de una o varias variables contextuales en los instantes de tiempo del futuro en los cuales queremos realizar la predicción junto con la información del pasado procesada y codificada por el codificador. Esto en teoría debería de ayudar a mejorar las predicciones, pues disponemos de información del futuro.

2.4.4. Entrenamiento

Tras describir las cuatro topologías de RNA que vamos a emplear en este trabajo para abordar el problema de predicción planteado, vamos a explicar los aspectos básicos del entrenamiento de una RNA Así se va a explicar en primer lugar cómo es el proceso de entrenamiento o de aprendizaje de una RNA en general para finalmente abordar los distintos aspectos y parámetros de los que depende dicho aprendizaje.

2.4.4.1. Proceso de aprendizaje

Desde la perspectiva de nuestro problema concreto de predicción, el entrenamiento de una RNA consiste en ir tomando series temporales de una determinada longitud (por ejemplo, con valores históricos de cuatro días), las cuales se denominan muestras o ejemplos de entrenamiento, y pasarlas como entrada a la RNA para que esta nos dé como salida los valores de consumo de calefacción de los siguentes n instantesde tiempo situados a continuación del último instante de tiempo de la serie temporal recibida como entrada. Tras esto, se comparan mediante una función de error los n valores predichos por la red con los verdaderos n valores, obteniendo una medida del error que comete la red, la cual nos indica la distancia existente entre los valores predichos por la red y los valores auténticos, y en base a este error se modifican los pesos de las distintas conexiones sinápticas de la red con el propósito de minimizarlo y así predecir mejor los valores de consumo de calefacción a partir de dicha serie temporal.

Más específicamente, el entrenamiento de una red neuronal se lleva a cabo durante un número determinado de iteraciones que se conocen como épocas de entrenamiento. Una época es un período de tiempo en el cual se le muestran a la RNA un subconjunto amplio o la totalidad de las muestras del conjunto de entrenamiento y consta de los siguientes pasos:

1. La red agrupa de manera determinista o aleatoria las m muestras de entrenamiento en lotes (batches) de tamaño k. De este modo se forman m/k lotes cada uno con k muestras.

2. Se toma un lote, y se hace lo siguiente:

   a) Se cogen todas sus muestras y se le pasan de una en una a la red obteniendo por cada una de ellas los siguientes n valores de consumo de calefacción predichos.

   b) Se comparan los valores predichos con los verdaderos y se obtiene el error cometido al predecir cada muestra.

   c) Finalmente se toman los errores correspondientes a todas las muestras y se calcula el error medio del lote, el cual mediante el algoritmo de backpropagation **[43]** se "propaga"hacia atrás desde la capa de salida hasta la capa de entrada para modificar los pesos de la red, reduciendo así el error que esta comete a la hora de predecir las muestras de este lote.

3. Se repite el paso 2 para el resto de lotes.

Vamos a ver con un poco más de detalle los aspectos y parámetros de los que depende este proceso de entrenamiento descrito.

2.4.4.2. Algoritmo de optimización

En el proceso de entrenamiento de una red neuronal hemos dicho que el error que comete la red al predecir cada lote de muestras se mide de alguna forma y se propaga hacia atrás. Pues bien, este se mide mediante una función f a la que denominamos función de error o pérdida. Por tanto, el problema de aprendizaje en RNA puede verse como un problema en el que tratamos de minimizar la función de pérdida, es decir, un problema de optimización.

En el entrenamiento de una RNA, pasamos a la red un lote de muestras para que esta las prediga y comparando dichas predicciones con los verdaderos valores a predecir de las muestras obtenemos el valor de la función de pérdida respecto a los pesos. Entonces, se calcula el gradiente de dicha función, que no es más que el vector de derivadas parciales de la función de pérdida (f a partir de ahora) y que nos indica la dirección de máxima variación del error.

Más formalmente, todos los pesos de cada una de las capas de una RNA, definen un espacio t-dimensional, donde t es el número total de pesos de la red. En dicho espacio, un conjunto de valores concreto de los pesos de la red es un punto. Entonces, durante el proceso de aprendizaje de la red queremos ir actualizando en un proceso iterativo (épocas) los pesos de la misma hasta obtener el conjunto de pesos óptimo que nos proporcione el mínimo error

posible sobre el conjunto de datos con el que estamos entrenando. Por tanto, podemos imaginarnos que lo que hacemos durante el entrenamiento de la red es ir moviéndonos de un punto a otro en el espacio t-dimensional que nos proporcione un menor error, lo que coincide con un problema de optimización. Por tanto, la función f depende de los pesos de cada una de las unidades (neuronas) de las capas de la red, siendo una función t-dimensional..

El gradiente de la función f nos va a indicar la dirección en la que tenemos que variar los pesos de nuestra red para movernos en el espacio t-dimensional hacia el conjunto óptimo de pesos que nos proporcione el mínimo error. Dicha actualización, de los pesos, como ya hemos mencionado previamente, se realiza "propagando" el error hacia atrás mediante el algoritmo backpropagation.

Para llevar a cabo este proceso de optimización de la función f existen multitud de algoritmos de aprendizaje, la mayoría basados en el algoritmo backpropagation, de los cuales el más clásico es el algoritmo de gradiente descendente estocástico (SGD) **[43]**, que realiza la actualización de los pesos en la dirección del gradiente de la función f de la siguiente forma:

$$\Delta w = -\eta f'(w) \tag{2.5}$$

Siendo Δw la variación de los pesos, η la tasa de aprendizaje que es un valor que regula el "salto" en la dirección del gradiente y $f'(w)$ la derivada (gradiente) de la función de pérdida.

Este algoritmo (SGD) tiene el principal problema de que emplea una tasa de aprendizaje (η) fija o estática, por lo que que tiene que ser el propio usuario el que mediante ensayo y error encuentre un valor de η adecuado, pues:

- Cuanto mayor sea el valor de la tasa de aprendizaje, mayor será la variación de los pesos de la red en la dirección del gradiente de f en cada actualización. Un valor demasiado alto, puede provocar que nos "pasemos" del conjunto de pesos óptimo y empecemos a "oscilar" entorno a él sin llegar nunca a dicho conjunto.

- Cuanto menor sea el valor de la tasa de aprendizaje, menor será la variación de los pesos de la red en la dirección del gradiente de f en cada actualización. Un valor demasiado bajo, puede provocar que nos quedemos "estancados" a medida que convergemos hacia el conjunto de pesos óptimo de la red.

Debido a esto, en algunos problemas puede llegar a ser muy complicado converger al conjunto de pesos óptimo empleando el algoritmo SGD. Entonces, de acuerdo con **[43]**, existen distintas técnicas que nos pueden ayudar a acelerar la convergencia:

- Momentos: Técnica que consiste en sumar el salto en los pesos $\Delta w(t-1)$ en el instante de tiempo anterior multiplicado por un parámetro μ (momento) al salto en los pesos en el instante de tiempo actual. Esto lo vemos en la siguiente ecuación.

$$\Delta w(t) = \mu \Delta w(t-1) - \eta \frac{\partial f}{\partial w} \tag{2.6}$$

  La actualización μ tendrá valores más pequeños cuanto mayor sea el gradiente de la función de error en el instante actual y más grandes cuanto menor sea el gradiente de la función. Esto nos permite aprender o converger con tasas de aprendizaje fijas que causarían convergencias divergentes en el gradiente descendente. El método de Nesterov **[52]** es un ejemplo de algoritmo de optimización que emplea esta técnica.

- Tasa de aprendizaje adaptativa: Esta técnica consiste en emplear por cada peso de la red una tasa de aprendizaje distinta, la cual depende de la magnitud del gradiente que tenga dicho peso. Hay varios algoritmos de optimización que emplean esta técnica, como Rmsprop **[53]** y Adam **[54]**.

- Técnicas de optimización de segundo orden: Tratan de determinar de manera adicional cuál es la mejor dirección en la que modificar el conjunto de pesos, empleando para ello la segunda derivada de la función de error. El método de Newton **[43]** es un ejemplo de este tipo de técnicas.

Para entrenar nuestras redes utilizaremos técnicas que empleen tasas de aprendizaje adaptativas como Rmsprop y Adam, pues son las más comúnmente utilizadas en aprendizaje profundo, nos quitan la responsabilidad de tener que entrenar cada red probando distintos valores de η o μ y son menos costosas que las técnicas de segundo orden.

Cabe destacar que sea cual sea el algoritmo de optimización que empleemos, el proceso de backpropagation en las capas LSTM se realiza mediante una adaptación del algoritmo backpropagation denominada backpropagation through time **[55]**.

2.4.4.3. Modos de aprendizaje

Dependiendo del tamaño de lote (batch) que utilicemos, existen varios modos o formas de aprendizaje **[43]**:

- Aprendizaje online: En este modo, se emplea un tamaño de lote de una muestra, por lo que en cada época daremos tantos pasos como muestras haya en el conjunto de entrenamiento. Esto implica que se actualizan los pesos con el error que comete la red al predecir cada muestra

de entrenamiento, lo que suele tener la principal ventaja de una convergencia mucho más rápida pues damos muchos "saltos"pequeños en la dirección del gradiente. Sin embargo, presenta el inconveniente de que se ejecuta en cada época el algoritmo de optimización un elevado número de veces lo que vuelve lento el entrenamiento.

- Aprendizaje por lotes: Consiste en utilizar lotes del mismo tamaño que el conjunto de entrenamiento, por lo que en cada época se calcula el error que comete la red en todas las muestras de entrenamiento y este se propaga hacia atrás. Tiene la principales ventajas de que las condiciones de convergencia teóricas son bien conocidas y de que se puede analizar la convergencia mucho mejor teóricamente, pues siempre propagamos el error global. Eso sí, el número de actualizaciones de pesos (uno por época) es mucho menor, por lo que tarda más en converger.

- Aprendizaje por mini lotes: Consiste en emplear un tamaño de lote intermedio (entre 2 y num_muestras - 1). Presenta la ventaja de que se realizan menos cálculos que en el aprendizaje online y un mayor número de actualizaciones de los pesos que en el aprendizaje por lotes, por lo que con este modo de entrenamiento, si se emplean lotes de muestras distribuidas de forma equilibrada, se suelen conseguir mejores resultados y más rápidamente que con los dos modos de aprendizaje anteriores.

2.4.5. Evaluación

A la hora de entrenar un modelo de RNA (y cualquier modelo de aprendizaje automático en aprendizaje supervisado), debemos dividir en dos las muestras de las que disponemos dando lugar así a los siguientes dos conjuntos:

- Conjunto de entrenamiento: Suele estar formado por el 70 % o 80 % de las muestras.

- Conjunto de validación: Suele estar formado por el 30 % o el 20 % de las muestras.

Ambos conjuntos deben de contener muestras similarmente distribuidas, para lo cual típicamente estas se depositan en un conjunto u otro de forma aleatoria. Entonces, las muestras del conjunto de entrenamiento se utilizan para entrenar el modelo y las del conjunto de validación, con las cuales no ha entrenado, para evaluarlo. Esta evaluación consiste en obtener el error que comete el modelo al predecir las muestras de validación, el cual será un error aproximado del que cometerá realmente el cuando se enfrente a muestras del mundo real.

Es lógico pensar que si una red tiene un error bajo a la hora de predecir las muestra de entrenamiento, también sea bajo el error que comete con las muestras de validación. Pues nada más lejos de la realidad, ya que si la red tiene un error muy bajo sobre el conjunto de entrenamiento significa que no solo ha aprendido los patrones que le serán útiles de cara a la predicción del conjunto de validación sino que también ha podido aprender los detalles particulares presentes en el conjunto de entrenamiento que no se encuentran en el de validación ni en muestras del mundo real, por lo que el error que comete el modelo sobre el conjunto de validación será más elevado de lo esperado. Este fenómeno se conoce como sobreaprendizaje del conjunto de entrenamiento y provoca que nuestro modelo no generalice bien.

Para evitar el sobreaprendizaje se puede optar por entrenar con más datos, sin reducir los datos de validación. Pero esto no siempre es posible y nuestro caso no es una excepción. Por tanto, otra estrategia a la que se puede recurrir es ajustar los parámetros de la red para que esta tenga la capacidad de aprendizaje justa para aprender los patrones útiles del conjunto de entrenamiento sin aprender sus particularidades. Siguiendo esta estrategia podemos por un lado limitar la capacidad de la red modificando su topología, reduciendo su número de capas y/o el número de neuronas por capa. Por otro lado, podemos limitar la capacidad de la red actuando sobre su algoritmo de aprendizaje, para lo cual existen varias técnicas **[43]**:

- Early Stopping: Consiste en entrenar la red, hasta que su rendimiento sobre el conjunto de validación (medido al final de cada época de entrenamiento) empiece a empeorar, momento en el cual se detiene el entrenamiento.
- Weight decay: Es una técnica consistente en imponer una penalización a los pesos grandes, reduciendo así la capacidad de aprendizaje de las neuronas de las capas ocultas.
- Ruido: Consiste en añadir ruido a los pesos de la red.
- Dropout: Técnica consistente en eliminar aleatoriamente de una capa oculta concreta un porcentaje de las neuronas para cada caso de entrenamiento. De esta forma se evita que las neuronas ocultas dependan/confíen demasiado en el trabajo de otras neuronas de su misma capa. Pues si una neurona oculta sabe que existen otras neuronas, estas pueden co-adaptarse para aprender el conjunto de entrenamiento, pero estas co-adaptaciones pueden no funcionar sobre el conjunto de validación.

Para nuestro problema emplearemos principalmente Dropout, pues a parte de las ventajas comentadas, es además una manera de combinar múltiples modelos en uno, ya que cuando entrenamos, por cada muestra de entrenamiento estamos probando varias arquitecturas diferentes (al eliminar unas

neuronas y otras no de forma aleatoria) por lo que el modelo final entrenado es un promedio de varios modelos o arquitecturas distintas.

Por último cabe destacar, que el Dropout se aplica normalmente a las capas totalmente conectadas pero también se puede aplicar sobre las capas LSTM, mediante la técnica de dropout recurrente [**45**].

2.5. XGBoost

Además de técnicas de aprendizaje profundo, también vamos a utilizar alguna técnica de aprendizaje automático más tradicional para abordar nuestro problema de predicción y de esta forma ver si las técnicas de aprendizaje profundo ofrecen una mejora de rendimiento significativa que justifique su uso. Dicho esto, la técnica seleccionada ha sido XGBoost debido a que también se va a utilizar para la selección de variables y a las buenas características que posee como algoritmo de aprendizaje automático ya comentadas en el apartado 2.3.5.1.2 y del rendimiento que se ha visto que puede ofrecer en la predicción de series temporales [**56**].

Los algoritmos de aprendizaje automático más tradicionales como XGBoost tienen los siguientes dos problemas a la hora de realizar predicción de series temporales en múltiples instantes de tiempo en el futuro:

1. A diferencia de las RNA no pueden recibir como entrada directamente una serie temporal multivariable, pues esta está almacenada en una matriz de m filas por n columnas, y estos modelos solo pueden recibir vectores unidimensionales.

2. También a diferencia de las RNA, estos modelos solo pueden dar como salida un único valor, por lo que no pueden predecir los valores de la variable objetivo en varios instantes de tiempo en el futuro.

El primer problema se soluciona simplemente convirtiendo la serie temporal de entrada en un vector unidimensional de tamaño m x n. Para resolver el segundo problema se pueden optar por varias estrategias [**57**], [**58**]:

- Estrategia recursiva: Consiste en predecir a partir de una serie temporal de entrada el valor de la variable objetivo en el siguiente instante de tiempo para a continuación incorporar dicho valor a la serie temporal de entrada y continuar prediciendo los siguientes valores de forma recursiva. Esta estrategia presenta el principal inconveniente de que como el modelo utiliza sus propias predicciones como entrada, las cuales presentan errores, a medida que se van realizando cada vez más predicciones se va acumulando dichos errores. Por esta razón, esta estrategia no es apta para tareas de predicción en las cuales hay que predecir el valor de la variable bastantes instantes de tiempo en el futuro.

- Estrategia directa: Consiste en entrenar de forma independiente n modelos con las mismas muestras o series temporales, de forma que el modelo i es entrenado para predecir únicamente el valor de la variable objetivo en el instante de tiempo i futuro. Esta estrategia tiene la principal ventaja de que no utiliza valores aproximados para realizar la predicción, por lo que no sufre del problema de acumulación de errores. Sin embargo, como los modelos se entrenan de forma independiente, las n predicciones de la variable objetivo no presentan dependencias estadísticas entre ellas y además la obtención de un modelo mediante esta estrategia es bastante costosa desde el punto de vista computacional, pues tenemos que entrenar n modelos.
- Estategia mixta: Combina las anteriores dos estrategias, empleando varios modelos para predecir el valor de la variable objetivo en cada instante de tiempo en el futuro y reutilizando los valores predichos de nuevo como entrada.

Estas tres estrategias presentan además otro inconveniente, pues al igual que sucede con la arquitectura MLP (sección 2.4.3.1) los modelos resultantes de estas estrategias procesan la serie temporal de entrada de una sola vez codificada en forma de vector unidimensional sin tener en cuenta el orden cronológico de sus observaciones.

En nuestro caso, vamos a optar por la estrategia directa, pues aunque implique un coste computacional mayor, no presenta el problema de la acumulación de errores en las predicciones. Esto nos dará lugar a un modelo formado a su vez por n modelos de XGBoost entrenados cada uno con los mismos datos y mismos parámetros. Así, este modelo recibe como entrada una serie temporal aplanada en forma de vector, la cual se le pasa como entrada a cada modelo de XGBoost. Entonces las salidas de los n modelos se almacenan en un vector que en la posición u observación i contendrá la predicción de la variable objetivo en el instante de tiempo i futuro, la cual es realizada por el modelo i de XGBoost.

2.6. Aproximación metodológica

Tras describir la teoría subyacente a las diferentes técnicas y métodos que aplicaremos con el propósito de resolver nuestro problema de predicción, vamos a describir de una manera esquemática las distintas tareas que desempeñaremos para abordar el problema, tras lo cual mencionaremos las distintas tecnologías que se utilizarán para llevar a cabo dichas tareas.

2.6.1. Diagrama de tareas

El siguiente diagrama organiza de forma general el proceso que realizaremos para abordar el problema de predicción.

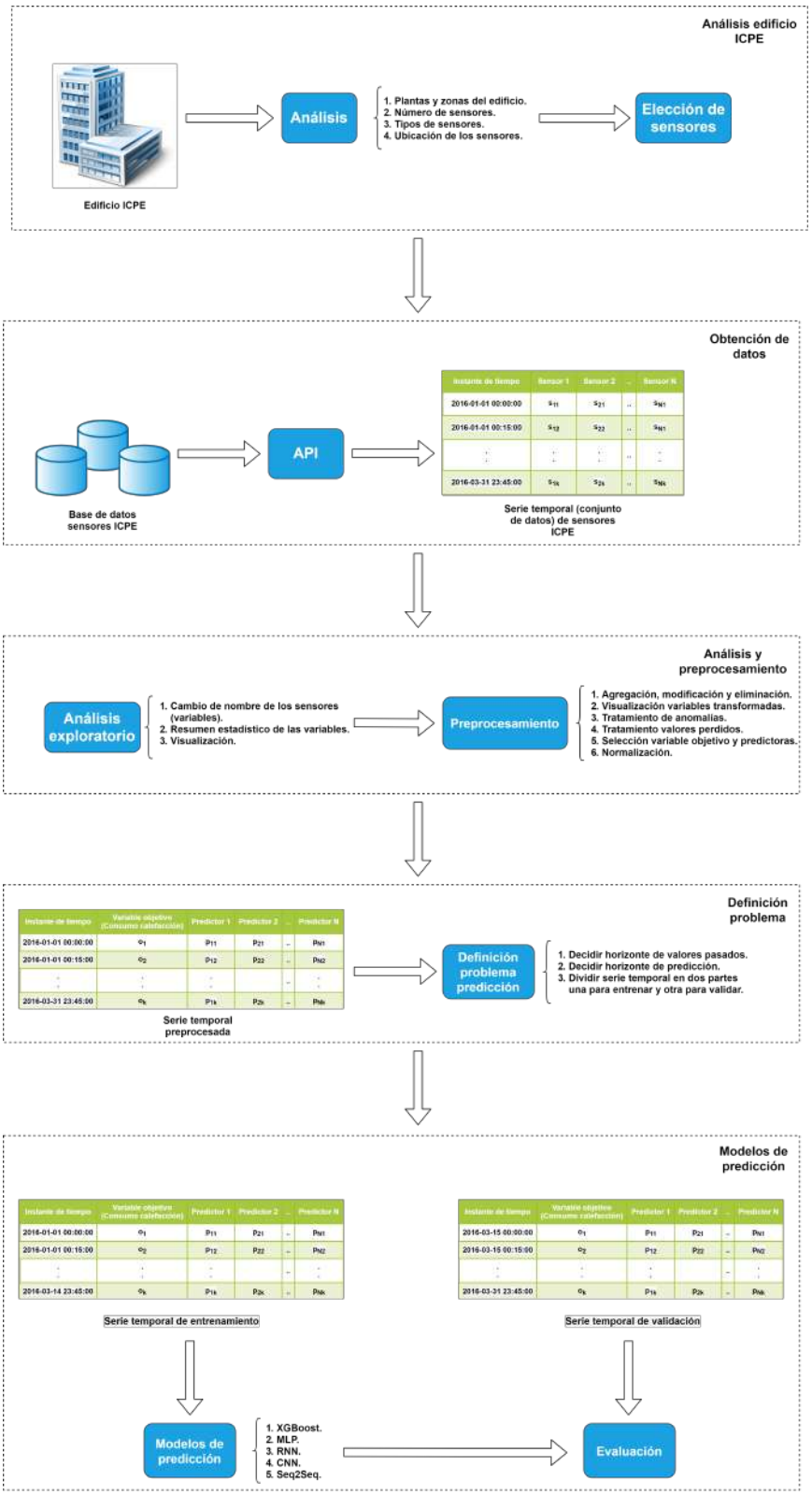


Figura 2.11: Diagrama de tareas.

En la figura 2.11 podemos ver las distintas tareas lque llevaremos a cabo para predecir el consumo de calefacción en el edificio ICPE organizadas en cinco bloques (rectángulos con bordes discontinuos), los cuales están ordenados de arriba hacia abajo y cada uno de ellos agrupa varias tareas (rectángulos de color azul) ordenadas de izquierda a derecha. Vamos entonces a comentar en este orden las tareas de cada bloque.

1. Análisis del edificio ICPE: En este primer bloque vemos que en primer lugar llevaremos a cabo un análisis del edificio ICPE con el propósito de conocer cuantas plantas y zonas tiene, el número de sensores, el típo (qué miden) y su localización (donde miden). Tras lo cual, se elegirán el conjunto de sensores concreto que vamos a emplear para abordar nuestro problema de predicción.

2. Obtención de datos: Tras haber obtenido un mayor conocimiento sobre el edificio y haber elegido el conjunto de sensores con los que vamos a trabajar, en este segundo bloque se llevarán a cabo tareas encaminadas a extraer mediante un API REST los valores de dichos sensores de una base de datos y almacenarlos en un conjunto de datos o serie temporal en forma de matriz, en la cual tendremos en las filas los distintos instantes de tiempo en orden cronológico y en las columnas los valores de los sensores (s_{ij}).

3. Análisis y preprocesamiento: La serie temporal obtenida en el bloque anterior se tomará en este tercer bloque y se le aplicarán en orden las tareas de análisis exploratorio y de preprocesamiento:

   - Análisis exploratorio: Con el objetivo de conocer mejor los sensores o variables de la serie temporal se le aplican a esta última una serie de tareas de análisis:
     a) Cambiar el nombre de los sensores para identificarlos más fácilmente.
     b) Obtener varias estadísticas generales de cada sensor.
     c) Visualizar en una gráfica la evolución a lo largo del tiempo de los valores de cada sensor.
   - Preprocesamiento: Con el conocimiento extraído acerca de las variables en el apartado anterior se aplican un conjunto de transformaciones a la serie temporal para poder aplicarle técnicas de aprendizaje automático y decidir el problema concreto a resolver. Vemos que se llevarán a cabo 5 tareas de preprocesamiento en el siguiente orden:
     a) Agregación, modificación, eliminación de variables y filas de la serie temporal, con el objetivo de desechar variables e instantes de tiempo que que no nos resultan útiles.

b) Visualización de cada una de las variables transformadas mediante su descomposición y autocorrelación.

c) Tratamiento de valores anómalos de cada una de las variables identificados en la tarea anterior.

d) Tratamiento de valores perdidos mediante varias técnicas de imputación de series temporales como mice, mstdi, irmi, etc.

e) Selección de la variable objetivo a predecir que mida el consumo de energía eléctrica destinada a la calefacción y de las variables más relevantes para predecirla (variables predictoras) empleando para ello un estudio de la correlación entre variables y el algoritmo XGBoost.

f) Normalizar la variable objetivo y sus predictoras para mejorar el entrenamiento de los modelos de predicción.

4. Definición del problema: En este cuarto bloque se parte de la serie temporal preprocesada, la cual vemos que solo contiene los valores de la variable objetivo y de sus variables predictoras más relevantes. Entonces, lo que se hará en este punto es decidir en cuantos instantes de tiempo en el futuro deseamos predecir los valores de la variable objetivo y a partir de cuántos valores en el pasado de ella misma y del resto de variables predictoras llevamos a cabo dicha predicción. Tras lo cual se divide la serie temporal en dos, generando una serie de entrenamiento y otra de validación.

5. Modelos de predicción: En este último bloque se parte de las series temporales de entrenamiento y validación obtenidas en el bloque anterior. Estas dos series vemos que respetan el orden cronológico sus instantes de tiempo (filas), pues los instantes de tiempo de la serie de entrenamiento son más antiguos que los de la serie de validación. Dicho esto, vemos que con la serie temporal de entrenamiento se aprenden modelos de predicción de XGBoost, MLP, RNN, CNN y Seq2Seq para predecir las variable objetivo, mientras que la serie temporal de validación es utilizada para evaluar cada uno de estos modelos para ver cual ofrece un mejor rendimiento.

2.6.2. Tecnología

Vamos ahora a describir el hardware y el software principales que se emplearán para llevar a cabo el proyecto.

2.6.2.1. Hardware

El proyecto se desarrollará íntegramente en un PC de sobremesa con las siguientes especificaciones técnicas:

- Procesador: AMD Ryzen 7 1800X 8 núcles / 8 núcleos virtuales.
- Memoria RAM: 16GB.
- Targeta gráfica: NVIDIA RTX 2080 Super 8GB VRAM GDDR6 1830 MHz 3072 núcleos CUDA.

Vemos que el PC que se utilizará dispone de una tarjeta gráfica de última generación con elevada velocidad de reloj y una gran cantidad de núcleos CUDA, que es indispensable a la hora de entrenar más rápido los modelos de aprendizaje profundo de manera paralela.

2.6.2.2. Software

El proyecto se desarrollará en el sistema operativo Ubuntu 18.04 LTS [**59**] empleando el lenguaje de programación Python 3.6 [**60**] con el IDE PyCharm 2019.3.5 [**61**]. Empleamos Python debido a su facilidad de uso, la familiaridad que poseemos con él y a que las bibliotecas más importantes para implementar modelos de redes neuronales están disponibles en este lenguaje. Además de Python, para llevar a cabo las tareas de tratamiento de outliers y de valores perdidos emplearemos el lenguaje R 3.6.1 [**62**] debido a que posee una mayor variedad de bibliotecas enfocadas a dichas tareas.

Para implementar los modelos de redes neuronales existen multitud de bibliotecas disponibles para Python que nos evitan tener que programarlos desde cero y nos permiten aprovechar la capacidad de cómputo paralelo de nuestra tarjeta gráfica para entrenarlas. En nuestro caso, hemos elegido la biblioteca Tensorflow [**63**] en su versión 1.4.3 por los siguientes motivos:

- Es la biblioteca más conocida y empleada en la actualidad. Más que otras como Caffe [**64**], PyTorch [**65**], Theano [**66**] o CNTK [**67**]. Por tanto, es la que más documentación tiene disponible el la web. De hecho, la mayor parte de los tutoriales que podemos encontrar en la web son de Tensorflow y emplean el lenguaje Python [**68**].
- Nos permite implementar todos los modelos de redes neuronales que deseamos probar en este proyecto.
- Proporciona la herramienta Tensorboard que permite monitorizar el entrenamiento de las redes neuronales.
- Es compatible con las bibliotecas de CUDA [**69**] y cuDNN [**70**], con las cuales podemos utilizar nuestra tarjeta gráfica NVIDIA para acelerar el entrenamiento de las redes neuronales mediante computación paralela.

Sin embargo, en lugar de utilizar la biblioteca Tensorflow directamente emplearemos también la biblioteca Keras [**71**], la cual es una interfaz de alto

nivel para la programación de redes neuronales que puede hacer uso «por debajo» de la biblioteca Tensorflow y es lo que haremos nosotros. Empleamos Keras principalmente por los siguientes motivos:

- Tensorflow es una biblioteca de bajo nivel y Keras nos permite seguir utilizando Tensorflow pero con una capa más de abstracción haciendo más fácil su uso.
- Al igual que Tensorflow es una biblioteca que posee una amplia comunidad y documentación.
- Estamos bastante familiarizados con su uso.
- A pesar de ser de más alto nivel que Tensorflow nos sigue aportando la flexibilidad suficiente para implementar los modelos de red neuronal que deseamos probar en este proyecto.

Capítulo 3

Datos

En este capítulo se describe el edificio ICPE y las características generales de sus sensores, para acabar profundizando en las características de los sensores de una zona piloto del mismo y de otras variables adicionales con las cuales vamos a trabajar.

Antes de enfrentarnos al problema de predicción planteado es necesario describir brevemente el edificio ICPE y las características de sus sensores, con el propósito de tener un contexto y conocimiento básicos acerca del problema para poder abordarlo de manera adecuada. Así, este capítulo se ha dividido en 6 secciones: En Descripción del edificio, se expondrán algunas de las características básicas del mismo. En Características generales de los sensores veremos algunas de las propiedades básicas que comparten todos los sensores del edificio. En Sensores de la zona piloto describiremos con una mayor profundidad los sensores de una zona piloto del edificio, con los cuales vamos a trabajar. Mientras que, finalmente, en Variables adicionales describiremos dos variables adicionales con las cuales también trabajaremos para tratar de resolver el problema de predicción.

3.1. Descripción del edificio

ICPE es un edificio de oficinas situado en la ciudad de Bucarest, la cual está localizada en el sudeste del país de Rumanía y es su capital municipal, cultural, industrial y financiera. En la siguiente imagen podemos ver el edificio desde una posición aérea.

Figura 3.1: Vista aérea del edificio ICPE.

En la figura 3.1 vemos una imagen del edificio ICPE desde arriba, el cual está resaltado en color verde. Este está compuesto por un sótano y tres plantas, las cuales tienen la misma forma y se distribuyen en las mismas cinco áreas: D1, D2, D3, D4 y D5. De estas, las áreas D1, D2, D3 y D4 están situadas en la parte norte del edificio, mientras que el área D5 se encuentra en la parte sur. En la figura 3.2 podemos ver la distribución de las cinco áreas en cada planta.

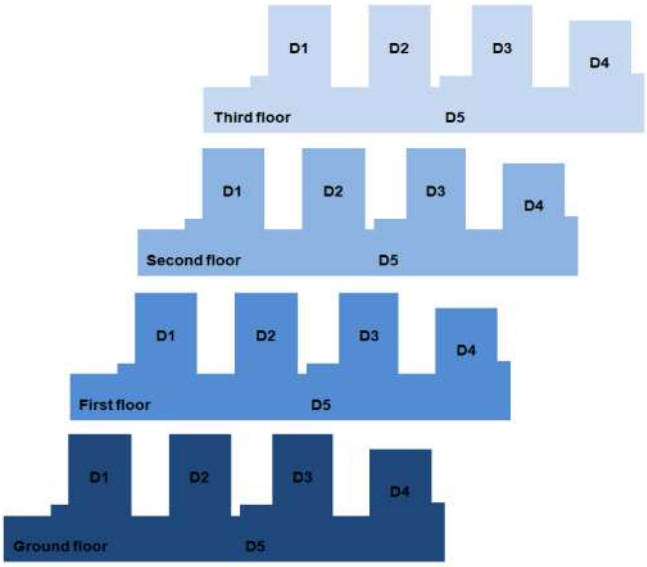


Figura 3.2: Distribución de las áreas D1, D2, D3, D4 y D5 en las plantas del edificio ICPE.

En las áreas de las plantas 1, 2 y 3 solo hay oficinas, mientras que en el

sótano hay en el área D5 un par de tiendas y salas de exposiciones y en las áreas D1, D2, D3 y D4 laboratorios de investigación.

3.2. Características generales de los sensores

El edificio ICPE consta de 278 sensores distribuidos en distintas zonas interiores y exteriores, los cuales registran valores de temperatura, presión, humedad, consumo eléctrico, consumo de los sistemas de calefacción, consumo de agua, etc. Todos y cada uno de estos sensores se caracterizan por:

- Zona: Cada sensor mide una magnitud física determinada en una zona concreta, la cual puede ser interior o exterior. Así, por ejemplo, hay sensores que miden el consumo de electricidad en una zona concreta y un sensor general que mide el consumo eléctrico en todo el edificio.
- Magnitud: Un sensor mide una magnitud física concreta, como la temperatura o el consumo eléctrico a la calefacción calefacción.
- Resolución: Todos y cada uno de los 278 sensores del edificio ICPE realizan una medición cada 15 minutos, lo que hace que en un día completo cada sensor realice un total de 96 mediciones.
- Identificador: Cada sensor posee un identificador (ID) unívoco, que es un número de 4 o 5 cifras.

3.3. Sensores de la zona piloto

En el proyecto EIT [**6**] no trabajaron con todas las plantas y áreas del edificio, puesto que como es un edificio viejo con espacios distintos para diferentes propósitos, por lo que tuvieron que centrarse en una zona concreta. Así, tendiendo en cuenta los diferentes subsistemas del edificio (calefacción, iluminación, eléctrico, etc) decidieron «partir» el edificio por la mitad y centrarse solo en la mitad izquierda (si miramos el edificio desde arriba como en la figura 3.2). A esta mitad se le denomina zona piloto ya que fue la elegida para aplicar el proyecto EIT y hacer la demostración. Esta está formada por tres plantas (sin el sótano) y por las áreas D1, D2 y la mitad izquierda del área D5 que ahora pasa a llamarse D5/2. En la figura 3.3 podemos ver las plantas y áreas de la zona piloto.

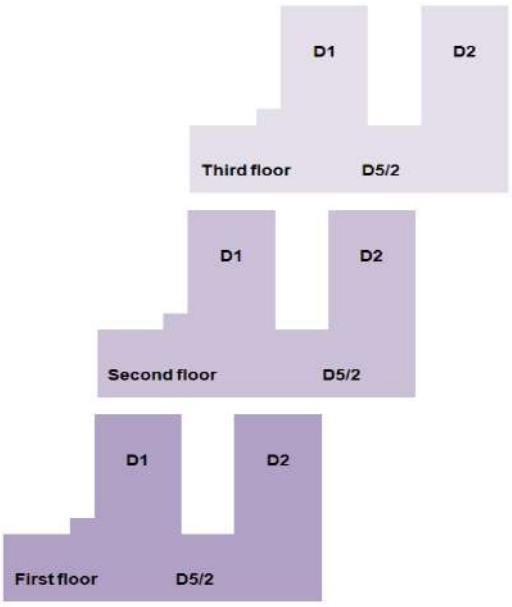
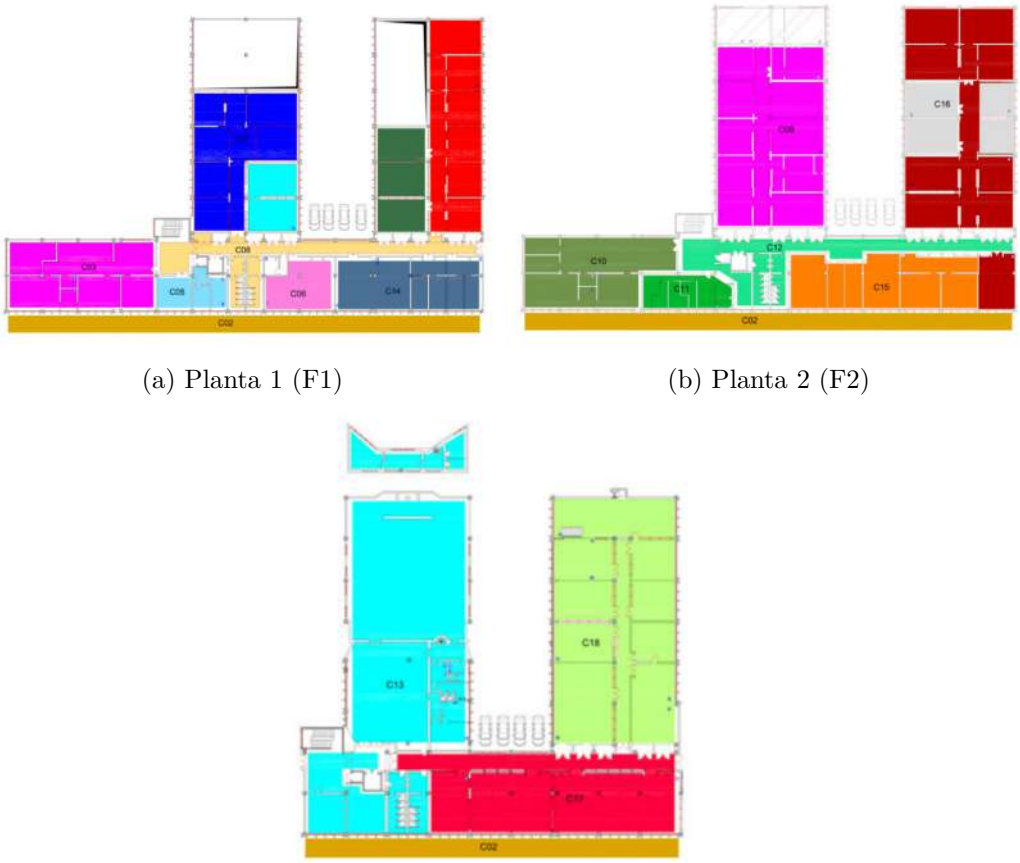


Figura 3.3: Distribución de las áreas D1, D2 y D5/2 en las plantas de la zona piloto del edificio ICPE.

En nuestro caso también vamos a centrarnos en esta zona piloto y además para afrontar nuestro problema de predicción del consumo de calefacción solo emplearemos los valores de los sensores de esta zona que fueron emplearon en el proyecto EIT. Esto lo hacemos por los siguientes motivos:

- Estos sensores se han empleado previamente en el proyecto EIT, lo cual hace que estos sean fiables en el sentido de que tendrán menos valores perdidos.
- No sabemos prácticamente nada acerca de los sensores que no fueron utilizados en el proyecto EIT. Solo conocemos sus valores y su identificador, pero desconocemos la magnitud física que miden y la zona en la que se encuentran.
- Estos sensores creemos que son suficientes para predecir a partir de sus valores pasados el consumo de calefacción, pues miden el consumo de electricidad, agua y calefacción.
- Al centrarnos únicamente en estos sensores simplificamos el problema, pues manejamos un menor número de sensores, lo cual facilita los procesos de preprocesamiento de los datos y de entrenamiento y evaluación de los modelos de aprendizaje.

En la siguiente figura podemos ver con más detalle la distribución de áreas y subáreas en las tres plantas de la zona piloto.



(a) Planta 1 (F1)

(b) Planta 2 (F2)

(c) Planta 3 (F3)

Figura 3.4: Plantas y áreas de la zona piloto del edificio ICPE.

En la figura 3.4 vemos que en cada planta (subfiguras a, b y c) el área D1 está situada en el noroeste, el área D2 en el noreste y el área D5/2 al sur y que cada una de ellas es un subconjunto de varias oficinas. Para visualizar mejor estas tres áreas podemos mirar la subfigura c, ya que en ella vemos el área D1 en color azul, el área D2 en verde y el área D5/2 en rojo.

Un sensor de la zona piloto puede medir una magnitud concreta en el área de una planta concreta, en un área en las tres plantas al mismo tiempo, en un subárea dentro de un área de una planta concreta o en toda la zona piloto. En este caso, estos sensores se pueden dividir en tres grupos dependiendo de la magnitud que miden.

- Sensores que miden el consumo dirigido a iluminación y equipos electrónicos. (sensores de consumo eléctrico a partir de ahora).
- Sensores que miden el consumo eléctrico dirigido a los sistemas de

calefacción (sensores de consumo de calefacción a partir de ahora).

- Sensores que miden el consumo de volumen de agua (sensores de consumo de agua a partir de ahora).

A continuación, vamos a ir viendo por cada grupo de sensores una tabla que nos proporcionará distintas características (columnas) acerca de los sensores (filas) que pertenecen al mismo. Estas características son las siguientes:

- Sensor type: Es el tipo de sensor, que en este caso puede ser contador de consumo eléctrico, de calefacción o de agua.
- BEMS Name: Es el nombre del sensor y además nos indica algunas características extra como por ejemplo, si el sensor mide la energía consumida o producida, si pertenece a una zona concreta dentro de un área, etc.
- Served area: Indica la zona o zonas a las que pertenece el sensor y se caracteriza por dos parámetros:
  - Floor: Número de planta (1, 2, 3) del edificio ICPE en la que se encuentra el sensor.
  - Area: Área D1, D2 o D5/2 en la que se encuentra el sensor. Cabe destacar que hay algunos sensores que pueden medir el consumo de una de estas tres áreas en todas las plantas al mismo tiempo.
  - Equipment Status: Indica si el sensor está en funcionamiento o no.
  - ID Sensor: Indica el identificador unívoco del sensor.

Dicho esto, vamos a ver las tablas con las anteriores características de los sensores de cada uno de los tres grupos.

3.3.1. Sensores de consumo eléctrico

Cada uno de estos sensores mide el consumo eléctrico de un área en una planta concreta, a excepción de dos sensores, uno que mide el consumo eléctrico de toda la zona piloto y otro que mide la energía eléctrica producida por los paneles solares fotovoltaicos situados en la fachada del edificio ICPE. Todo esto lo podemos ver en la siguiente tabla.

Sensor type	*BEMS Name*	*Served Area*	*Equipment status*	*ID Sensor*
Energy meter	C01 - 3P - Imported active energy PA4	Floor G,1,2,3 Area D1, D2, D5/2	Fully functional	8963
Energy meter	C02 - 3P – Exported active energy PV facade	Entire building	Fully functional	8901
Energy meter	C03 - 3P - Input active energy F1 – D5/2	Floor 1 Area D5/2	Fully functional	8971
Energy meter	C05 - 3P - Input active energy PAS	Floor 1 Area D5/2	Fully functional	8978
Energy meter	C06 - 3P - Input active energy F1 – D5/2	Floor 1 Area D5/2	Fully functional	9020
Energy meter	C07 - 3P - Input active energy PAN	Floor 1 Area D1	Fully functional	9015
Energy meter	C08 - 3P - Input active energy F1 – D5/2	Floor 1 Area D5/2	Fully functional	9021
Energy meter	C09 - 3P - Input active energy F2 – D1	Floor 2 Area D1	Fully functional	8868
Energy meter	C10 - 3P - Input active energy PAW	Floor 2 Area D5/2	Fully functional	7426
Energy meter	C11 - 3P - Input active energy F2 – D5/2	Floor 2 Area D5/2	Fully functional	9024
Energy meter	C12 - 3P - Input active energy F2 – D5/2	Floor 2 Area D5/2	Fully functional	8865
Energy meter	C13 - 3P - Input active energy F3 – D1	Floor 3 Area D1	Fully functional	8929
Energy meter	C14 - 3P - Input active energy F1 – D5/2	Floor 1 Area D5/2	Fully functional	8273
Energy meter	C15 - Input active energy F2 – D5/2	Floor 2 Area D5/2	Fully functional	8933
Energy meter	C16 - Input active energy F2 – D2	Floor 2 Area D2	Fully functional	8937
Energy meter	C17 - 3P - Input active energy F3 – D5/2	Floor 3 Area D5/2	Fully functional	8951
Energy meter	C18 - 3P - Input active energy F3 – D2	Floor 3 Area D2	Fully functional	8886

Tabla 3.1: Sensores de consumo eléctrico.

En la tabla 3.1 vemos que todos los sensores de este grupo funcionan correctamente (columna Equipment Status), sin embargo, no tenemos ningún dato en todo 2016 de los sensores 8273 y 7426, por lo que no podremos trabajar con ellos. Además, vemos también en las columnas Sensor type y BEMS que todos los sensores son de tipo Energy meter, lo que significa que son contadores eléctricos, es decir, en el instante de tiempo t indican el consumo total hasta ese momento de energía eléctrica de un área determinada en un planta concreta. Por último, también cabe destacar que si miramos la columna Served Area podemos ver que tanto en la planta 1 como en la 2

hay 6 sensores, de los cuales uno mide todo el consumo en el área D1, otro en el área D2 y los cuatro restantes miden en distintas subáreas del área D5/2. Mientras, en la planta 3 solo hay 3 sensores, uno para cada área.

3.3.2. Sensores de consumo de calefacción

Los sensores que pertenecen a este grupo miden el consumo de calefacción en la zona piloto. Los sistemas de calefacción del edificio funcionan calentando el agua que circula por las cañerías subterráneas de la calle en la que se encuentra el edificio y poniéndola en circulación por las tuberías de calefacción del edificio. Estos sistemas tienen un diseño vertical lo que hace que algunos de los sensores midan el consumo de calefacción de un área en la tres plantas al mismo tiempo, tal y como podemos ver en la siguiente tabla.

Sensor type	*BEMS Name*	*Served Area*	*Equipment status*	*ID Sensor*
Heating agent meter	Heating Agent - Energy Counter: D1	Floor 1,2,3 – Area D1	Fully functional	9093
Heating agent meter	Heating Agent - Energy Counter: D2	Floor 1,2,3 – Area D2	Equipment failure	9092
Heating agent meter	Heating Agent - Energy Counter: D5E	Floor 1,2,3 – Area D5E	Fully functional	9091
Heating agent meter	Heating Agent - Energy Counter: D5W	Floor 1,2,3 – Area D5W	Fully functional	9074
Heating agent meter	Heating Agent - Energy Counter: PAN	Floor 1– Area D1 - PAN (included in D1)	Fully functional	9086
Heating agent meter	Heating Agent - Energy Counter: PAS	Floor 1– Area D5W - PAS (included in D5E)	Fully functional	9059
Heating agent meter	Heating Agent - Energy Counter: PAW	Floor 2– Area D5W - PAW (included in D5W)	Fully functional	9085

Tabla 3.2: Sensores de consumo de calefacción.

En la tabla 3.2 podemos ver en las columnas Sensor type y BEMS que todos los sensores de consumo de calefacción son de tipo Heating agent meter, lo que significa que también son contadores al igual que los sensores de consumo eléctrico. Por otra parte, en la columna Served Area vemos que que hay cuatro sensores (los cuatro primeros) que miden el consumo de calefacción de un área concreta en las tres plantas, de los cuales dos lo

hacen sobre el área D1 y D2 y los otros dos lo hacen para las subáreas D5E y D5W, estas dos últimas áreas hacen referencia a la parte Este y Oeste respectivamente del área D5/2. Estos cuatro primeros sensores son aquellos cuyo consumo queremos predecir ya miden el consumo de calefacción total en las áreas D1, D2 y D5/2. Sin embargo, vemos que uno de ellos (el segundo) experimentó fallos en su funcionamiento por lo que no podremos trabajar con él, lo que implica que no podremos predecir el consumo de calefacción en el área D2.

En cuanto al resto de sensores, vemos que hay dos de la planta 1 y uno de la planta 2 y cada uno de ellos mide el consumo de calefacción de un subárea concreta perteneciente. En este caso, podemos ver que las subáreas de estos tres últimos sensores son:

- PAN: Es un subárea al norte (Pilot area north) del área D1.
- PAS: Es un subárea al sur (Pilot area south) de la parte oeste del área D5/2.
- PAW: Es un subárea al oeste (Pilot area west) de la parte oeste del área D5/2.

Cabe destacar, que el consumo de los tres últimos sensores está incluido en los sensores que miden el consumo de calefacción en las plantas 1,2,3 en las áreas D1 y D5W.

3.3.3. Sensores de consumo de agua

En este grupo únicamente tenemos un sensor que mide el consumo de agua de todo el edificio tal y como vemos en la siguiente tabla.

Sensor type	*BEMS Name*	*Served Area*	*Equipment status*	*ID Sensor*
Domestic water meter	Domestic Water – Volumetric Counter	Entire building	Fully functional	9049

Tabla 3.3: Sensor de consumo de agua.

En la tabla 3.3 podemos ver en las columnas Sensor type y BEMS Name que este sensor de consumo de agua es también de tipo contador, por lo que sus valores indican el total de agua consumido en todo el edificio hasta un instante de tiempo t determinado. Por último vemos en la columna Equipment Status que este sensor está en perfectas condiciones.

3.4. Variables adicionales

Además de los sensores de la zona piloto, también disponemos de dos variables adicionales, las cuales nos aportan información bastante relevante

acerca del edificio ICPE. Estas son:

- Variable de temperatura: Recoge la temperatura exterior del edificio ICPE. Para obtenerla, debido a que se desconoce si alguno de los 278 sensores del edificio ICPE mide esta magnitud, se han tomado de del repositorio National Oceanic and Atmospheric Administration [**72**] las temperaturas registradas en 2016 en Bucarest cada 15 minutos.

- Variable de ocupación: Recoge el porcentaje de ocupación del espacio útil por parte de los trabajadores que en cada instante en el edificio ICPE. Esta, se ha obtenido a partir de una distribución de la ocupación basada en la agenda del edificio (no hay un sensor que mida la ocupación), la cual puede verse en la siguiente tabla.

From	*To*	*Mo*	*Tu*	*We*	*Th*	*Fr*	*Sa*	*Su*
0:00	7:20	0 %	0 %	0 %	0 %	0 %	0 %	0 %
7:20	7:30	Linearly 0-100 %	Linearly 0-100 %	Linearly 0-100 %	Linearly 0-100 %	Linearly 0-100 %	0 %	0 %
7:30	12:00	100 %	100 %	100 %	100 %	100 %	0 %	0 %
12:00	12:30	Linearly 100-75 %	Linearly 100-75 %	Linearly 100-75 %	Linearly 100-75 %	Linearly 100-75 %	0 %	0 %
12:30	13:00	Linearly 75-100 %	Linearly 75-100 %	Linearly 75-100 %	Linearly 75-100 %	Linearly 75-100 %	0 %	0 %
13:00	13:30	100 %	100 %	100 %	100 %	100 %	0 %	0 %
13:30	13:40	100 %	100 %	100 %	100 %	Linearly 100-0 %	0 %	0 %
13:40	16:00	100 %	100 %	100 %	100 %	0 %	0 %	0 %
16:00	16:10	100-0 %	100-0 %	100-0 %	100-0 %	0 %	0 %	0 %
16:10	0:00	0 %	0 %	0 %	0 %	0 %	0 %	0 %

Tabla 3.4: Distribución semanal de la ocupación del edificio ICPE.

En la tabla 3.4 podemos ver los porcentajes de ocupación semanal estimados en el edificio ICPE. En este caso, se puede ver que en los dias laborables (columnas de lunes (Mo) a viernes (Fr)) la ocupación es del 0 % durante la madrugada y por la tarde-noche mientras que de 7:20 a 16:00 hay intervalos horarios en los que crece y decrece de forma lineal y otros en los que permanece constante al 100 %. Mientras en las dos últimas columnas vemos que los sábados y los domingos no trabaja nadie en el edificio, pues la ocupación es del 0 % en todos los intervalos horarios. A partir de esta tabla se ha obtenido mediante interpolaciones lineales (pues la tabla indica que la ocupación varía de manera lineal) los valores de ocupación del edificio cada 15 minutos, los cuales conforman esta variable.

Capítulo 4

Análisis y preprocesamiento

En este capítulo se describirán las tareas (y sus resultados) de obtención de datos, análisis exploratorio y preprocesamiento llevadas a cabo con el propósito de adecuar los datos del edificio ICPE para poder aprender con ellos modelos de predicción y mejorar la calidad de los mismos.

En el capítulo anterior se describió de forma general las propiedades de los senores del edificio ICPE y más en profundidad las de los sensores de la zona piloto y las variables de ocupación y temperatura externa. Dicho esto, vamos a llevar a cabo las siguientes tareas necesarias para poder obtener modelos de predicción de las variables de consumo de calefacción que mencionamos en el apartado 3.3.2.

1. Recopilar los datos de los sensores de la zona piloto con los que se pretende entrenar los modelos de predicción.

2. Realizar un análisis exploratorio de los datos obtenidos con el fin de mejorar nuestra comprensión acerca de los mismos y de extraer alguna hipótesis inicial sobre los patrones que se podrían extraer de ellos.

3. Preprocesar los datos con el objetivo de que puedan ser utilizados para entrenar modelos de predicción de mayor calidad.

Tras estas tareas ya estaremos preparados para entrenar modelos de aprendizaje profundo con el propósito de predecir el consumo de calefacción en distintas zonas del edificio ICPE.

Dicho esto, vamos a ver en primer lugar cómo ha sido el procedimiento de obtención de los datos del edificio.

4.1. Adquisición de datos

Para obtener los datos de los sensores de la zona piloto se nos ha proporcionado un API . Mediante esta API podemos acceder a los valores registrados por cada uno de los 278 sensores del edificio ICPE a lo largo del año 2016 (es el único año en el cual tenemos valores disponibles). Más concretamente, el API nos permite obtener de cada sensor los valores que registra a lo largo de un día con una resolución de un cuarto de hora. Dicho esto, como cada sensor funciona las 24 horas del día, registra un total de 96 valores diarios. Los datos que nos proporciona esta API han sido sometidos a una fase de preprocesmiento previo consistente eliminar algunos valores anómalos y en alinear los valores de los sensores para que todos tengan una resolución de 15 minutos [**73**].

De esta manera, cada uno de los 96 registros diarios de un sensor concreto que nos devuelve el API, contiene la siguiente información.

- ID Sensor: Identificador unívoco del sensor.
- Building: El nombre del edificio (ICPE en nuestro caso) al que pertenece el sensor.
- day: El el año-mes-día en el que se ha registrado el valor.
- time: Es la hora en formato HH:MM:SS en la que se ha registrado el valor.
- value: Valor concreto captado por el sensor.

Con este API hemos recopilado los valores registrados por todos los sensores de la zona piloto a lo largo de 2016 y los hemos almacenado en un conjunto de datos en el cual cada fila contiene una observación realizada en una fecha y hora concretas (instante de tiempo) que tiene en cada columna el valor registrado por un sensor concreto en esa fecha y hora. Cabe destacar, que las filas están ordenadas cronológicamente desde el 5 de Enero hasta el 31 de Diciembre de 2016 y que la separación entre filas es de un cuarto de hora. Por tanto, tal y como vimos en la sección 2.1, nuestro conjunto de datos es una serie temporal regular multivariable. En la tabla 4.1 podemos ver las primeras 10 filas y 8 columnas de esta serie temporal.

Datetime	*8865*	*8868*	*8886*	*8929*	*8933*	*8937*	*8951*	*8971*
2016-01-05 00:00:00	459	12743	2807	653	15233	7072	57289	11965
2016-01-05 00:15:00	459	12743	2807	653	15233	7072	57289	11965
2016-01-05 00:30:00	459	12743	2807	653	15233	7072	57289	11965
2016-01-05 00:45:00	459	12743	2807	653	15233	7072	57289	11965
2016-01-05 01:00:00	459	12743	2807	653	15233	7072	57289	11965
2016-01-05 01:15:00	459	12743	2807	653	15233	7072	57289	11965
2016-01-05 01:30:00	459	12743	2807	653	15233	7072	57289	11965
2016-01-05 01:45:00	459	12743	2807	653	15233	7072	57289	11965
2016-01-05 02:00:00	459	12743	2807	653	15233	7072	57289	11965
2016-01-05 02:15:00	459	12743	2807	653	15233	7072	57289	11965

Tabla 4.1: Primeras 10 filas y 8 columnas de la serie temporal.

Tras obtener esta serie temporal, esta se ha dividido en dos, de manera que nos hemos quedado con dos series temporales o conjuntos de datos, uno con los valores registrados por los sensores en los meses de enero, febrero, marzo y abril, y otro con los valores de los sensores en los meses de septiembre, octubre, noviembre y diciembre. De esta manera, nos estamos quedando únicamente con los valores registrados por los sensores en los meses más fríos del año, lo cual se debe a dos motivos principalmente:

- Nuestro objetivo es predecir el consumo de calefacción, la cual solo funciona en los meses fríos.

- En los meses de calor (mayo, junio, julio y agosto) se llevaron a cabo obras en el edificio ICPE, por lo que muchos sensores no estuvieron activos, faltando así una gran cantidad de datos.

Para obtener cada uno de esos dos conjuntos de datos se ha hecho lo siguiente:

1. Por cada mes de cada conjunto de datos se ha buscado un rango de fechas los más amplio posible. De forma que todos los días entre la fecha inicial y final del mismo todos los sensores de la zona piloto realicen mediciones. Esto se hace para evitar tener en ambos conjuntos valores perdidos, pues puede haber días en los que uno o varios sensores no realizan medición alguna.

2. Una vez determinados los rangos de fechas por cada mes, se extrae del conjunto de datos original un conjunto de datos por cada mes con los valores registrados por los sensores de la zona piloto en el rango de fechas elegido para ese mes.

3. Tras esto, se unifican los conjuntos de datos de todos los meses. Obteniendo así, un conjunto de datos con los valores de los sensores de la zona piloto en enero, febrero, marzo y abril y otro con los valores de los sensores en los meses de septiembre, octubre, noviembre y diciembre.

En las siguientes tablas podemos ver para cada conjunto de datos obtenido el rango de fechas elegido por cada mes.

Mes	Rango fechas
Enero	2016-01-05 - 2016-01-31
Febrero	2016-02-01 - 2016-02-25
Marzo	2016-03-01 - 2016-03-31
Abril	2016-04-01 - 2016-04-25

Tabla 4.2: Rangos de fechas para el conjunto de datos de Enero, Febrero, Marzo y Abril.

Mes	Rango fechas
Septiembre	2016-09-02 - 2016-09-30
Octubre	2016-10-01 - 2016-10-31
Noviembre	2016-11-01 - 2016-11-11

Tabla 4.3: Rangos de fechas para el conjunto de datos de Septiembre, Octubre y Noviembre.

En la tabla 4.2 se puede observar que los datos que hemos recogido en los meses de enero, febrero, marzo y abril van desde el 5 de enero al 25 de abril de manera continua exceptuando los días 26, 27, 28 y 29 (2016 es año bisiesto) de febrero durante los cuales no estuvo en funcionamiento ningún sensor del edificio por causas desconocidas. Estos valores perdidos serán tratados posteriormente en la fase de preprocesamiento.

Por otra parte, en la tabla 4.3 vemos que los datos recogidos de los meses de septiembre, octubre, noviembre y diciembre van desde el 2 de deptiembre hasta el 11 de noviembre, día a partir del cual no realizan mediciones todos los sensores de la zona piloto. Razón por la cual no se ha podido recoger datos del mes de diciembre. Como veremos más adelante, los valores de los sensores en este conjunto no se llegarán a utilizar debido a que en estos meses no se registró consumo de calefacción.

4.2. Análisis exploratorio

Tras obtener los dos conjuntos de datos con los que vamos a trabajar vamos a llevar a cabo un análisis exploratorio de los mismos por los motivos ya expresados en la introducción de este capítulo. En la sección 2.2 ya se comentaron las tareas a realizar en esta etapa y el propósito de las mismas.

Por tanto, lo que nos queda es ver con más detalle como se ha llevado a cabo cada una de ellas.

4.2.1. Renombramiento de los sensores

En esta primera tarea se ha cambiado el nombre de cada columna de ambos conjuntos de datos (ambos conjuntos tienen las mismas columnas). De esta manera, se ha renombrado cada sensor con la siguiente nomenclatura: Tipo - Planta - Área - Subárea, donde:

- Tipo: Indica el tipo de sensor. Puede ser E (Electric), H (Heating) o W (Water).
- Planta: Indica la o las plantas en las que mide el sensor. Puede ser F1 (Planta 1), F2 (Planta 2), F3 (Planta 3), F123 (Plantas 1, 2 y 3), etc.
- Área: Área en la que mide el sensor. Puede ser D1, D2 o D5/2.
- Subárea: Zona concreta dentro del área en la que mide el sensor.

Esta nueva nomenclatura, que sustituye el ID de cada sensor, tiene el inconveniente de que puede resultar un poco larga pero a cambio podemos saber el tipo, planta, área y subárea de un sensor solo con observar su nombre sin necesidad de consultar o aprendérnoslo de memoria, lo que facilita bastante las tareas de análisis exploratorio.

En la tabla 4.4 podemos ver el id de cada sensor junto con su nuevo nombre.

ID Sensor	Nuevo nombre
8971	E-F1-D5/2-C03
8978	E-F1-D5/2-C05
9020	E-F1-D5/2-C06
9015	E-F1-D1
9021	E-F1-D5/2-C08
8868	E-F2-D1
9024	E-F2-D5/2-C11
8865	E-F2-D5/2-C12
8929	E-F3-D1
8933	E-F2-D5/2-C15
8937	E-F2-D2
8951	E-F3-D5/2
8886	E-F3-D2
9093	H-F123-D1
9091	H-F123-D5/2E
9074	H-F123-D5/2W
9086	H-F1-D1-PAN
9059	H-F1-D5/2W-PAS
9085	H-F2-D5/2W-PAW
9090	H-PAW02
9073	H-PAW03
9094	H-PAW01
9049	W-ALL

Tabla 4.4: Nueva nomenclatura de los sensores de la zona piloto.

Por último, hay que mencionar que las variables ocupación y temperatura exterior se denominan Occupation y Ext-Temp respectivamente.

4.2.2. Resumen estadístico

Después de renombrar los sensores, vamos a obtener varias estadísticas de las columnas (variables) en ambos conjuntos de datos. En este caso, al tener dos conjuntos de datos vamos a obtener por cada variable dos conjuntos de estadísticas, uno con las estadísticas de dicha variable en los meses de Enero, Febrero, Marzo y Abril y otro con las de los meses de Septiembre, Octubre y Noviembre.

Dicho esto, por cada variable en cada conjunto se han obtenido las siguientes 8 medidas estadísticas:

- Nº Observaciones: Número de valores totales de la columna, es decir,

el número de filas del conjunto de datos.

- Media: Media de los valores de la columna.
- Desviación típica: Distancia que hay en media de cada valor de la columna al valor medio de la misma.
- Mínimo y Máximo: Valor mínimo y máximo respectivamente de la columna.
- Cuartiles 25, 50 y 75: Valores que tienen por debajo al 25, 50 y 75 por ciento de los valores de la columna.

En las siguientes dos tablas podemos ver las estadísticas obtenidas para cada variable en ambos conjuntos de datos.

Variable	Nº Observaciones	Media	Desviación típica	Mínimo	25 %	50 %	75 %	Máximo
E-F2-D5/2-C12	10368	539.52	47.28	459	494	545	585	608
E-F2-D1	10368	14380.94	739.57	12743	13801.75	14554.5	15087.25	15331
E-F3-D2	10368	3302.71	233.13	2807	3106.75	3348.5	3517.25	3643
E-F3-D1	10368	684.22	19.49	653	666.75	684.5	700.25	719
E-F2-D5/2-C15	10368	16994.74	918.06	15233	16264.75	17058.5	17809.25	18498
E-F2-D2	10368	7874.03	419.36	7072	7540.75	7911.5	8283	8464
E-F3-D5/2	10368	59187.02	809.38	57289	58679.75	59281.5	59826.25	60456
E-F1-D5/2-C03	10368	14209.89	911.29	11965	13531.75	14640.5	14940.25	15297
E-F1-D5/2-C05	10368	2650.68	199.42	2313	2468.75	2671.5	2828	2987
E-F1-D1	10368	8519.50	422.84	7597	8194.75	8586.5	8914.25	9113
E-F1-D5/2-C06	10368	4891.81	346.28	4292	4576.75	4917.5	5202.25	5465
E-F1-D5/2-C08	10368	127.59	0.49	127	127	128	128	128
E-F2-D5/2-C11	10368	7173.44	399.44	6416	6823.75	7166.5	7534.25	7849
W-ALL	10368	20339.40	11416.03	0	10037.5	21225.0	30502.5	39350
H-F1-D5/2W-PAS	10368	5491.03	401.41	4552	5275.75	5526.0	5920.25	5933
H-PAW03	10368	4225	545.22	3022	3850.75	4376.5	4756.25	4778
H-F123-D5/2W	10368	93604.06	14601.18	61173	83918.5	97352.0	107976.0	108545
H-F2-D5/2W-PAW	10368	10912.80	1089.63	8539	10153.75	11170.5	11994.25	12039
H-F1-D1-PAN	10368	36409.28	1262.41	33377	35690.25	36741.5	37665.5	37691
H-PAW02	10368	4578.03	544.67	3407	4193.75	4685.5	5129.25	5151
H-F123-D5/2E	10368	203881.98	49162.42	87690	177539.0	221940.5	247167.75	248628
H-F123-D1	10368	87678.40	13944.25	57411	78158.75	91402.0	101395.75	101950
H-PAW01	10368	2110	0	2110	2110	2110	2110	2110
Occupation	10752	24.18	42.46	0	0	0	0	100
Ext-Temp	10752	3.21	7.69	-21.1	-3.125	4.2625	9.075	20.6

Tabla 4.5: Tabla con las estadísticas de las variables en Enero, Febrero, Marzo y Abril.

Variable	Nº Observaciones	Media	Desviación típica	Mínimo	25 %	50 %	75 %	Máximo
E-F2-D5/2-C12	6816	691.29	26.34	660	667	682	712	751
E-F2-D1	6816	17052.74	394.89	16551	16726	16912	17368.25	17859
E-F3-D2	6816	4610.59	147.81	4427	4491	4564	4703	4922
E-F3-D1	6816	857.90	12.49	833	848	860	870	875
E-F2-D5/2-C15	6816	24807.62	785.45	23436	24238	24690.5	25462.25	26240
E-F2-D2	6816	9503.18	227.63	9249	9322	9397.5	9746	9939
E-F3-D5/2	6816	68690.75	932.02	66831	68188	68586.5	69362.25	70456
E-F1-D5/2-C03	6816	17035.78	545.13	16504	16612	16765	17434.25	18238
E-F1-D5/2-C05	6816	4018.77	153.40	3787	3891	3997	4144	4307
E-F1-D1	6816	11572.85	448.15	11047	11216	11382	11901.5	12509
E-F1-D5/2-C06	6816	7330.87	250.81	6924	7125	7305	7535.25	7783
E-F1-D5/2-C08	6816	129.55	0.49	129	129	130	130	130
E-F2-D5/2-C11	6816	9430.31	347.14	9009	9133	9311	9665.25	10154
W-ALL	6816	108143.81	8970.32	94370	100510	106960	115002.5	125070
H-F1-D5/2W-PAS	6816	5943.78	33.81	5933	5933	5933	5933	6142
H-PAW03	6816	4794.56	51.32	4778	4778	4778	4778	5065
H-F123-D5/2W	6816	108814.80	808.82	108545	108545	108545	108545	112862
H-F2-D5/2W-PAW	6816	12062.31	75.92	12039	12039	12039	12039	12469
H-F1-D1-PAN	6816	37711.65	81.48	37691	37691	37691	37691	38185
H-PAW02	6816	5157.89	24.79	5151	5151	5151	5151	5295
H-F123-D5/2E	6816	249974.23	4068.39	248628	248628	248628	248628	270968
H-F123-D1	6816	102194.44	756.19	101951	101951	101951	101951	106162
H-PAW01	6816	2110.0	0.0	2110	2110	2110	2110	2110
Occupation	6816	24.20	42.47	0	0	0	0	100

Tabla 4.6: Tabla con las estadísticas de las variables en Septiembre, Octubre y Noviembre.

Dicho esto, en las tablas 4.5 y 4.6 podemos observar lo siguiente.

- Tenemos un total de 25 variables de las cuales 23 son sensores de consumo eléctrico, de calefacción o de agua y las dos restantes almacenan los valores de ocupación y de temperatura exterior del edificio.
- En la tabla 4.5 vemos en la columna Nº Observaciones que todas las variables tienen 10368 observaciones a excepción de las variables Occupation y Ext-Temp las cuales cuentan con 10752. Esto se debe a que como dijimos en el capítulo 5, ningún sensor de consumo registró valores en los días 26, 27, 28 y 29 de Febrero. Mientras, en la tabla 4.6 vemos en la columna Nº Observaciones que todas las variables tienen el mismo número de observaciones.
- La media y los valores mínimo y máximo de cada sensor en ambas tablas nos indica que distintos sensores de consumo del mismo tipo miden valores de consumo muy distinto. Esto se puede deber a que el consumo en una zona es distinto al de las demás, pues este depende de la frecuencia con la que los trabajadores del edificio utilicen dicha zona, del equipamiento que haya en la misma, de su tamaño, del número e trabajadores habituales en ella, etc.
- El valor máximo de cada sensor de consumo en la tabla 4.5 es menor que su valor mínimo en la tabla 4.6. Esto se debe a que como dijimos en el capítulo 3, los sensores son contadores y por tanto, el consumo total en el día 25 de Abril (último día del primer conjunto de datos) es menor que el consumo total acumulado el día 2 de Septiembre (primer día el segundo conjunto de datos).

- En los sensores de consumo de calefacción el valor máximo en la tabla 4.5 es igual a su valor mínimo en la tabla 4.6. Esto nos indica que desde el 25 de Abril hasta el 2 de Septiembre el consumo total de calefacción ha permanecido constante, lo que significa durante todo ese tiempo no se utilizó la calefacción. Esto tiene sentido, pues entre Abril y Septiembre se encuentran las meses más calurosos.

- La variable H-PAW01 tiene el mismo mínimo y máximo en ambas tablas, lo que nos indica que en el área PAW01 (subárea de D5/2W) no ha habido consumo de calefacción en ningún momento o no se ha registrado.

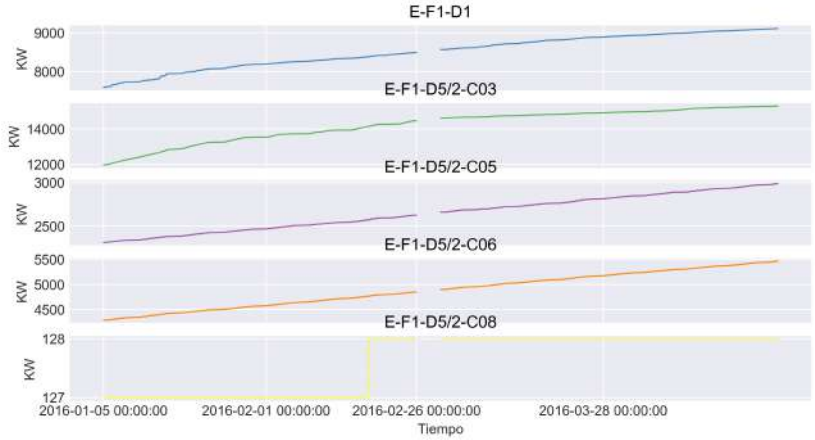
4.2.3. Visualización

Tras ver las estadísticas de cada una de las variables en nuestros dos conjuntos de datos, vamos a visualizar gráficamente la evolución de las observaciones de cada variable a lo largo del tiempo. Más en concreto, vamos a obtener por cada variable dos gráficas de líneas. Un con sus valores en los meses de Enero, Febrero, Marzo y Abril y otra con sus valores en los meses de Septiembre, Octubre y Noviembre. En las gráficas veremos el tiempo en el eje x y los valores de la variable en el eje y.

4.2.3.1. Sensores de consumo eléctrico

En primer lugar vamos a visualizar los valores de los sensores que miden el consumo eléctrico en la zona piloto. Para ello los hemos agrupado por plantas, de forma que primero vamos a visualizar las gráficas de los sensores de la planta 1, luego los de la planta 2 y finalizaremos con los de la planta 3.

En las siguientes dos figuras podemos ver pues los valores de los sensores de consumo eléctrico de la primera planta.



(a) Enero, febrero, marzo y abril.

(b) Septiembre, octubre y noviembre.

Figura 4.1: Valores de los sensores de consumo eléctrico de la planta 1.

En la figura 4.1 podemos ver la evolución en el tiempo de los valores de los sensores de consumo eléctrico de la primera planta de la zona piloto. Observando ambas subfiguras vemos que:

- En la planta 1 tenemos cinco sensores que miden el consumo eléctrico, de los cuales uno lo hace en todo el área D1 y el resto en distintas subáreas del área D5/2. Podemos ver que nos falta el sensor que mide el consumo eléctrico en el área D2, pues este es el sensor con ID 7426, sobre el cual no tenemos ninguna observación tal y como vimos en la sección 3.3.1.

- El área D5/2 se compone de múltiples subáreas o habitaciones y en

algunas de ellas se registra un mayor consumo que en el área D1. Esto nos puede estar indicando que en la planta 1 el área D5/2 es la más utilizada.

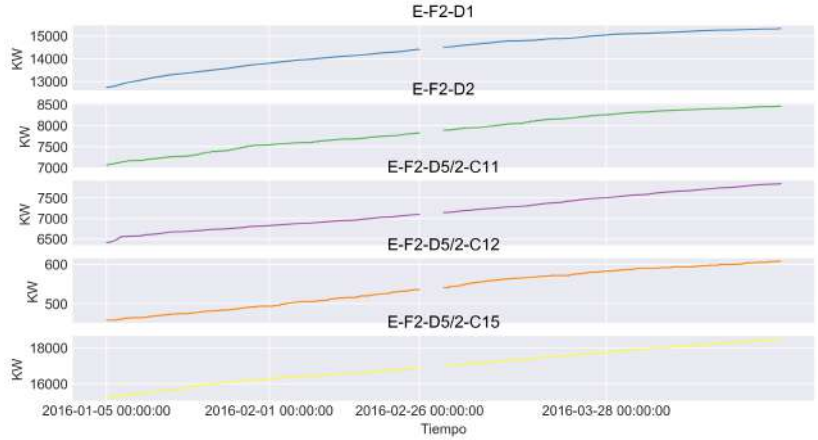
- El consumo registrado por los sensores en ambas subfiguras incrementa de forma lineal con el paso del tiempo, con la excepción de pequeños períodos de tiempo en los que las gráficas permanecen constantes. Estos períodos son aquellos en los que en el área o subárea en la que mide el sensor no se está consumiendo energía eléctrica.

- En cada subfigura hay períodos de tiempo similares en todas sus gráficas durante los cuales estas permanecen constantes. Esto nos indica que durante dichos períodos de tiempo no hay consumo de electricidad. Por tanto, pueden ser períodos de tiempo en los que el edificio está cerrado (horas no laborales, dias festivos, fines de semana, etc) o en los que la primera planta no se utiliza.

- Tanto en los meses de Enero, Febrero, Marzo y Abril como Septiembre, Octubre y Noviembre el sensor del subárea C08 del área D5/2 solo ha consumido 1 KW, hecho que ya se observó en la sección 4.2.2.

- En la subfigura a podemos ver como a finales de Febrero se rompe la continuidad de todas las gráficas debido a los valores perdidos en los días 26, 27, 28 y 29 de Febrero. Esto como veremos será así para todos los sensores.

A continuación vamos con los sensores de consumo eléctrico de la segunda planta.



(a) Enero, febrero, marzo y abril.

(b) Septiembre, octubre y noviembre.

Figura 4.2: Valores de los sensores de consumo eléctrico de la planta 2.

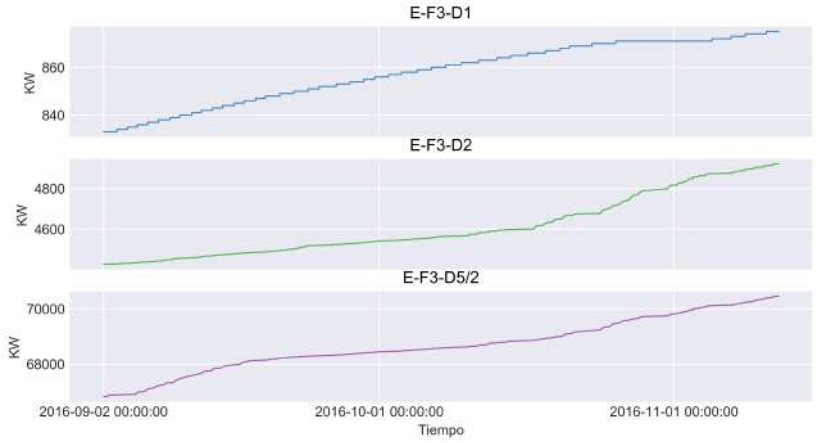
En la figura 4.2 podemos ver la evolución en el tiempo de los valores de los sensores de consumo eléctrico de la segunda planta en enero, febrero, marzo y abril (subfigura a) y en Septiembre, Octubre y Noviembre (subfigura b). En ella podemos observar lo siguiente:

- En la planta 2 hay 5 sensores que miden el consumo eléctrico, de los cuales uno mide el consumo de todo el área D1, otro el del área D2 y los tres restantes el de varias subáreas del área D5/2. En este caso tenemos información de consumo de todas las áreas.
- En esta segunda planta también vemos que el área D5/2 es la que más consumo registra al igual que sucede en la primera planta.

- Las observaciones realizadas en la planta 1 también tienen lugar en este caso (valores perdidos y crecimiento lineal).

Finalmente vamos a ver gráficamente los valores de los sensores de consumo eléctrico de la tercera planta.

(a) Enero, febrero, marzo y abril.



(b) Septiembre, octubre y noviembre.

Figura 4.3: Valores de los sensores de consumo eléctrico de la planta 3.

En la figura 4.3 podemos observar los valores de los sensores de consumo eléctrico de la tercera planta de la zona piloto en Enero, Febrero, Marzo y Abril (subfigura a) y en Septiembre, Octubre y Noviembre (subfigura b). En ella vemos lo siguiente:

- En la tercera planta hay tres sensores, uno de los cuales mide el consumo eléctrico del área D1, otro el del área D2 y otro el del área D5/2.

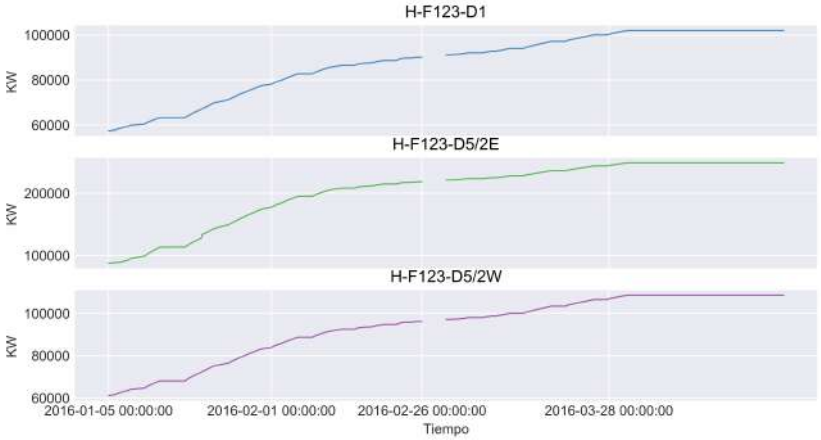
En esta planta , al igual que en la planta 2, tenemos datos de consumo de las tres áreas.

- Se observa también en esta planta que el área D5/2 es la que más consume.
- El resto de observaciones realizadas en las planta 1 y 2 también se dan en esta tercera planta (valores perdidos y crecimiento lineal).

4.2.3.2. Sensores de consumo de calefacción

A continuación vamos a visualizar los valores registrados por los sensores de consumo de calefacción de la zona piloto. Para ello, los hemos agrupado, al igual que los sensores de consumo eléctrico, por plantas. De esta manera, por cada grupo de sensores vamos a visualizar dos figuras, una con las gráficas de sus valores a lo largo de los meses de enero, febrero, marzo y abril y otra con las gráficas de sus valores a lo largo de los meses de septiembre, octubre y noviembre.

En primer lugar vamos a ver las gráficas de los sensores que miden el consumo de calefacción de un área concreta en las tres plantas en la zona piloto.



(a) Enero, febrero, marzo y abril.

(b) Septiembre, Octubre y Noviembre.

Figura 4.4: Valores de los sensores que miden el consumo de calefacción en las plantas 1, 2 y 3.

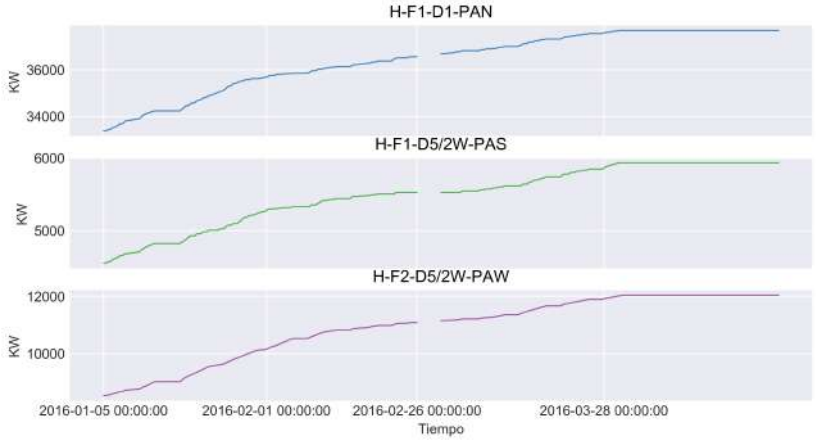
En la figura 4.4 podemos ver los valores de los sensores que miden el consumo de calefacción en las plantas 1, 2 y 3 de la zona piloto a lo largo de los meses de enero, febrero, marzo y abril (subfigura a) y a lo largo de Septiembre, Octubre y Noviembre (subfigura b). En ella se puede apreciar lo siguiente:

- Tenemos tres sensores que miden el consumo de calefacción total de las tres plantas de la zona piloto, de los cuales uno mide el consumo del área D1 y los otros dos el del área D5/2. De estos dos últimos uno mide el consumo del área D5/2 este (E) y otro el del área oeste (W), es decir, el área D5/2 está dividida en dos subáreas tal ya como se

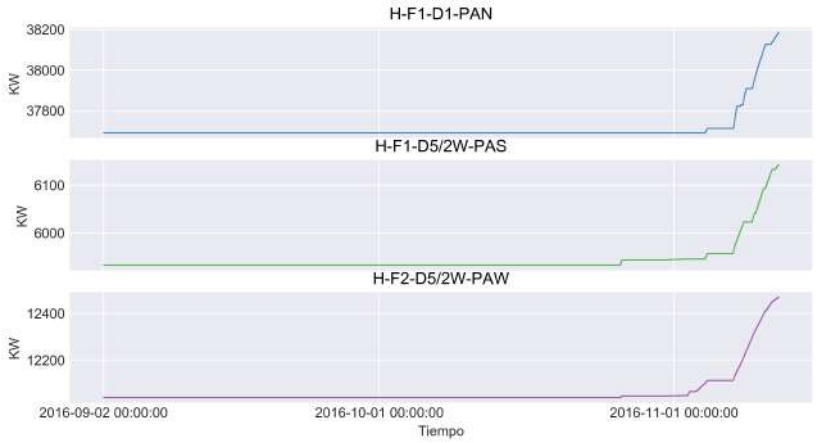
comentó en el capítulo 3. Por tanto, vemos que nos falta el sensor que mide el consumo de calefacción en el área D2 en las tres plantas, el cual experimentó fallos durante su funcionamiento a lo largo del año 2016 tal y como vimos también en el capítulo 3.

- Al igual que ocurría con el consumo eléctrico, el consumo de calefacción es mayor en el área D5/2.
- En las subfiguras a y b vemos que el consumo de calefacción crece en todas las áreas de forma lineal en los meses de Enero, Febrero, Marzo hasta que en Abril permanece constante hasta Noviembre. Esto nos indica que desde Abril hasta Noviembre no se ha utilizado la calefacción en el edificio.
- En la subfigura a se observa que estos sensores también tienen perdidos sus valores correspondientes a los días 26, 27, 28 y 29.

Ahora vamos tanto con los sensores que miden el consumo de calefacción en la primera planta como con los que lo hacen en la segunda.



(a) Enero, febrero, marzo y abril.



(b) Septiembre, octubre y noviembre.

Figura 4.5: Valores de los sensores que miden el consumo de calefacción en las plantas 1 y 2.

En la figura 4.5 vemos las gráficas de los sensores que miden el consumo de calefacción en distintas áreas de las plantas 1 y 2 de la zona piloto a lo largo de los meses de enero, febrero, marzo y abril (subfigura a) y de los meses de septiembre, octubre y noviembre (subfigura b). En ella se observa que:

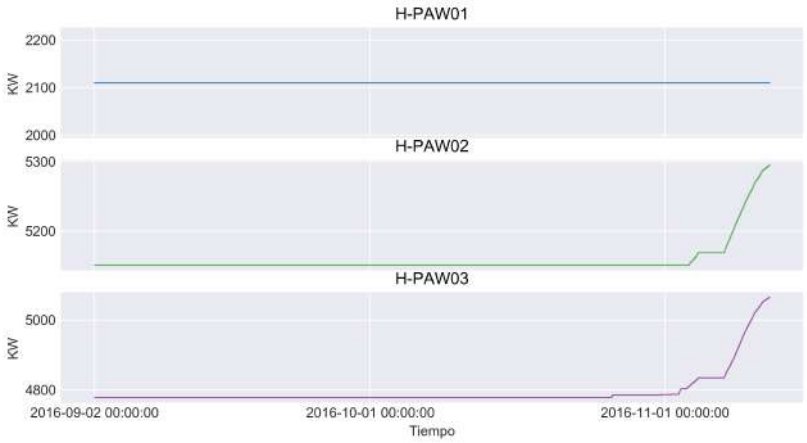
- Tenemos tres sensores de consumo de calefacción situados en las plantas 1 y 2. Más en concreto, dos se encuentran en la primera planta y el otro en la segunda. De los sensores de la planta 1, uno se encuentra en el subárea PAN del área D1 (H-F1-D1-PAN) y otro en el subárea PAS del área D5/2 oeste (H-F1-D5/2W-PAS). Mientras que el sensor

de la segunda planta se encuentra en el subárea PAW del área D5/2 oeste (H-F2-D5/2W-PAW).

- El consumo de calefacción medido por estos tres sensores está incluido en el consumo medido por los tres sensores que miden el consumo de calefacción total de cada área en las tres plantas. Así tenemos que el consumo registrado por el sensor H-F1-D1-PAN está incluido en el sensor H-F123-D1 y el consumo de los sensores H-F1-D5/2W-PAS y H-F1-D5/2W-PAW están incluidos en el sensor H-F123-D5/2W. Esto se verifica si comparamos las gráficas de cada uno de estos tres sensores con las de los sensores en los que están incluidos, pues vemos que crecen de la misma forma y que el consumo de estos es menor que el registrado por los sensores en los que están incluidos

Finalmente vamos a ver las gráficas de los sensores que miden el consumo de calefacción en el subárea PAW, la cual se encuentra dentro del área D5/2W-PAW.

(a) Enero, febrero, marzo y abril.



(b) Septiembre, Octubre y Noviembre.

Figura 4.6: Valores de los sensores que miden el consumo de calefacción en el subárea PAW.

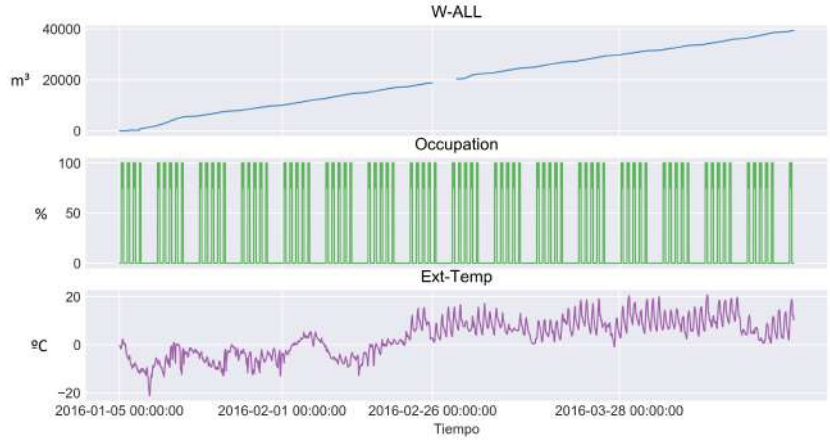
En la figura 4.6 vemos las gráficas de los sensores que miden el consumo de calefacción en el subárea PAW del área D5/2W de la zona piloto durante los meses de Enero, Febrero, Marzo y Abril (subfigura a) y los meses de Septiembre, Octubre y Noviembre (subfigura b). En ella podemos ver que:

- Tenemos tres sensores que miden el consumo de calefacción en el subárea PAW, cada uno en una zona concreta (1, 2 o 3).
- Dos de los sensores (H-PAW02 y H-PAW03) registran un crecimiento en el consumo de calefacción similar al registrado por los sensores de consumo de calefacción vistos con anterioridad. Mientras que el sensor

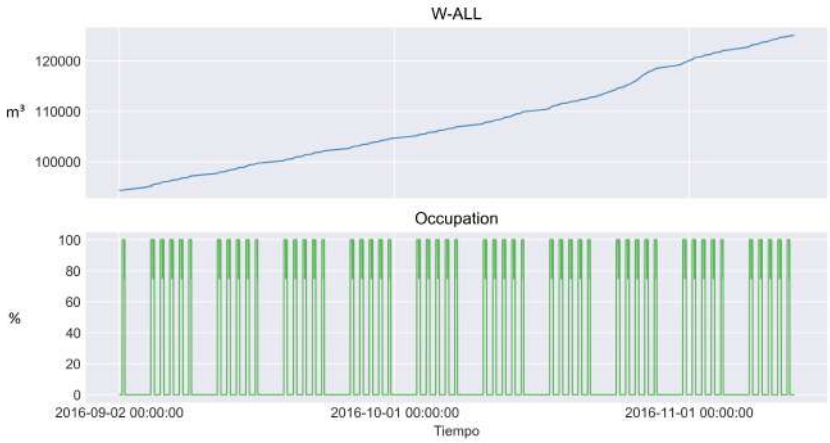
H-PAW01 no registra consumo alguno, pues permanece constante.

4.2.3.3. Sensor de consumo de agua y variables de ocupación y temperatura exterior

En esta última subsección vamos a visualizar la evolución en el tiempo de los valores del sensor de consumo de agua y de las variables de ocupación y temperatura exterior.



(a) Enero, febrero, marzo y abril.



(b) Septiembre, octubre y noviembre.

Figura 4.7: Valores del sensor de consumo de agua y de las variables de ocupación y temperatura externa.

En la figura 4.7 se encuentran los valores del sensor de consumo de agua y de las variables de ocupación y temperatura externa del edificio ICPE

durante los meses de enero, febrero, marzo y abril (subfigura a) y los meses de septiembre, octubre y noviembre (subfigura b). En ella vemos lo siguiente:

- En ambas subfiguras vemos que el consumo de agua registrado por el sensor W-ALL empieza en 0 el 5 de Enero y va creciendo linealmente hasta el 11 de Noviembre. Solo hay ciertos períodos de tiempo en los que permanece constante, que son las horas no laborales, días festivos, fines de semana, etc. Eso sí, en la subfigura a vemos que este sensor tampoco tiene los valores de consumo correspondientes a los días 26, 27, 28 y 29 de Febrero.

- La variable de ocupación presenta en ambas subfiguras unos patrones de crecimiento y decrecimiento que se repiten de forma exacta en el mismo período de tiempo. En este caso, dicho período es de una semana, y si nos fijamos bien en una semana vemos como la ocupación sube y baja 5 veces y después permanece a 0 hasta que se vuelve a repetir el patrón. Estas subidas y bajadas se producen en los días laborales, es decir, de lunes a viernes, de ahí que haya 5 subidas y bajadas en una semana. Más en concreto, en un día laboral, la ocupación sube de golpe hasta el 100 % a las 7:30, se mantiene entre el 75 % y el 100 % hasta las 16:00 y a las 16:30 desciende a 0 % hasta las 7:30 del día siguiente. Mientras, los mayores períodos de tiempo en los que la ocupación permanece a 0 % se corresponden con los fines de semana (Sábados y Domingos). En este caso, los patrones de ocupación se repiten de esta forma tan exacta porque los valores de esta variable son aproximados y se han obtenido mediante interpolación a partir de una distribución tal y como vimos en el capítulo 5.

- En la subfigura a vemos que las temperatura externa del edificio ICPE es fria, llegando incluso a bajo cero en los meses de Enero y Febrero, para acabar siendo más calurosas en Marzo y Abril ya que nos acercamos a la primavera. Mientras, en la subfigura b vemos como en los meses de Septiembre y Ocutbre de 2016 no tenemos valores de temperatura, debido a que no ha sido posible su obtención.

4.3. Preprocesamiento

Tras llevar a cabo el análisis exploratorio de los datos, a partir del cual hemos obtenido un conocimiento más profundo de los mismos, vamos a aplicar sobre los mismos un conjunto de técnicas de preprocesamiento basándonos en dicho conocimiento. Tal y como dijimos en la sección 2.3 estas técnicas serán las siguientes.

- Agregación, modificación y eliminación de datos (explicado en la sección 2.3.1).

- Visualización de los datos transformados (explicado en la sección 2.3.2).
- Tratamiento de outliers (explicado en la sección 2.3.3).
- Tratamiento de valores perdidos (explicado en la sección 2.3.4).
- Selección de variables (explicado en la sección 2.3.5).

Vamos a ver más detenidamente la aplicación de cada una de estas tareas sobre nuestros datos y sus resultados.

4.3.1. Agregación, modificación y eliminación de datos

En primer lugar vamos a realizar una serie de transformaciones sobre las variable (columnas) y las observaciones (filas) sobre nuestros dos conjuntos de datos, las cuales consisten en eliminar variables y agregar variables y modificar sus valores.

4.3.1.1. Eliminación y agregación de datos

En este apartado se han llevado a cabo las siguientes subtareas:

1. Como hemos visto en el análisis exploratorio de los datos, el consumo de calefacción es nulo en el edificio ICPE durante los meses de Septiembre, Octubre y Noviembre. Razón por la cual no podemos utilizar los valores de los sensores de dichos meses para predecir el consumo de calefacción. Por tanto, se ha descartado el conjunto de datos de septiembre, octubre y noviembre.

2. Centrándonos únicamente en el conjunto de datos de enero, febrero, marzo y abril, nos hemos percatado que en el mes de abril ya no hay consumo de calefacción debido al aumento de la temperatura local de Bucarest. Por tanto, hemos eliminado también los valores del mes de abril de todos y cada uno de los sensores.

3. Se ha eliminado la variable H-F123-D2, el cual, como vimos en la sección 3.3.2 experimentó fallos en su funcionamiento. Puesto que este sensor es una variable objetivo, ya no podemos predecir el consumo de calefacción en el área D2 en las tres plantas de la zona piloto del edificio ICPE. Por tanto, los valores de las variables E-F2-D2 y E-F3-D2 también se descartan ya que solo iban a ser candidatas para predecir la variable H-F123-D2.

4. También se descartan los valores de las variables H-PAW01 y E-F1-D5/2-C08 ya que como se vio en el apartado 6.2.3, estos no registran consumo alguno a lo largo del tiempo.

5. Por último se han sumado los valores de las variables H-F123-D5/2W y H-F123-D5/2E, dando lugar a la variable H-F123-D5/2 la cual contiene el consumo de calefacción en las tres plantas en el área D5/2. Así, resolvemos el problema de que a la hora de elegir las variables eléctricas para predecir las variables objetivo H-F123-D5/2W y H-F123-D5/E no sabíamos cuales pertenecían al área D5/2W y cuales al área D5/2E.

4.3.1.2. Obtención de los valores de consumo instantáneo

Como se ha mencionado en la sección 4.2, las variables de nuestro conjunto de datos (a excepción de las variables de ocupación y temperatura exterior) no nos ofrecen en cada instante de tiempo un valor de consumo instantáneo, sino el consumo total registrado hasta dicho instante de tiempo (contador). Sin embargo, en nuestro caso debemos tener los valores de consumo instantáneo en cada instante de tiempo debido principalmente a dos razones:

1. Nuestro principal objetivo es entrenar un modelo de red neuronal que reciba como entrada una serie temporal multivariable con n observaciones ordenadas temporalmente de más antigua a más reciente y nos proporcione como salida los valores instantáneos de consumo de calefacción en un área determinada en los siguientes k instantes de tiempo. Por tanto, si queremos predecir valores instantáneos de consumo, es conveniente entrenar con valores instantáneos de consumo.

2. A la hora de llevar a cabo la selección de variables para predecir la variable objetivo, si empleamos los valores de contador de los sensores tendremos que todos están altamente correlados, pues sus valores siempre crecen de forma lineal.

Por tanto, los valores de contador de cada variable de consumo se han convertido en valores de consumo instantáneo. Para ello, en cada sensor se ha calculado el consumo instantáneo en el instante de tiempo t como la diferencia entre el valor de contador de la variable de consumo en el instante de tiempo t y el registrado en el instante de tiempo t - 1. Esto se puede ver de una forma más genérica con la siguiente fórmula.

$$Cins_i(t) = Cc_i(t) - Cc_i(t-1) \tag{4.1}$$

Donde $\text{Cins}_\text{i}(\text{t})$ es el consumo instantáneo de la variable de consumo i en el instante de tiempo t y $\text{Cc}_\text{i}(\text{t})$ es el valor de contador del sensor i en el instante de tiempo t. Cabe destacar que esta operación hace que en cada variable perdamos los instantes de tiempo 2016-01-05 00:00:00 y 2016-03-01 00:00:00, pues el primero no tiene ningún instante de tiempo detrás (es el primero de la serie temporal) y el segundo tiene detrás un instante de tiempo

de valores perdidos. En cuanto al primer instante de tiempo no haremos nada, es decir, cada variable empezará en el instante 2016-01-05 00:15:00, mientras que el segundo pasará a ser un instante de tiempo perdido y lo trataremos a continuación junto a los demás instantes de tiempo perdidos.

4.3.2. Visualización de los datos transformados

Una vez hemos transformado nuestro conjunto de datos quedándonos con las variables y observaciones más útiles y convirtiendo los valores de contador de las variables de consumo en valores de consumo instantáneo, merece la pena volver a analizar visualmente las distintas variables. Aunque eso sí, esta vez no solo vamos a visualizar sus valores a lo largo del tiempo sino que además les aplicaremos las técnicas de descomposición y autocorrelación ya explicadas en la sección 2.1, con las cuales vamos a poder realizar un análisis y descripción más exahustivos de las mismas.

De esta manera, se ha tratado cada variable como una serie temporal univariable y se han aislado y analizado sus componentes de tendencia, estacionalidad e irregularidad, además de su autocorrelación para distintos lags de tiempo, extrayendo así conocimiento sobre estas variables que nos será útil de cara a las siguientes tareas de preprocesamiento.

El análisis de la descomposición y correlación de cada variable puede verse de forma detallada y gráfica en el apéndice A. Aquí exponemos las siguientes conclusiones generales extraídas de este análisis:

- En todas las variables (salvo la ocupación y temperatura exterior) hemos observado que hay valores puntuales anormalmente elevados sobre todo en Enero y Febrero. Estos valores serán tratados como outliers, los cuales tal y como se comentó en el apartado 2.3.3 de deben ser tratados adecuadamente ya que pueden entorpecer el entrenamiento de los modelos de predicción. Además, también hemos visto que todas las variables de consumo tienen valores perdidos correspondientes a los días 26, 27, 28 y 29 de Febrero, los cuales deben ser también tratados de una forma propicia.

- En general todas las variables presentan una cierta componente estacionaria, siendo las variables de consumo de calefacción las que mayor estacionalidad presentan y las de consumo eléctricas las que menos. Esta estacionalidad es un patrón que se repite en el período de una semana, en el cual los valores de consumo son más altos durante los días laborales (de lunes a viernes), pues es cuando más ocupado está el edificio, y son más bajos o nulos los fines de semana, es decir, cuando no hay nadie en el edificio.

- La mayoría de las variables tienden a mantener sus valores de media constantes salvo las de consumo de calefacción. Pues este desciende

conforme avanzamos de Enero a Marzo al ascender las temperaturas exteriores del edificio y reducirse la demanda de energía.

- Todas las variables salvo la ocupación presentan componentes irregulares, que se deben principalmente a comportamientos de consumo impredecibles o irregulares. Estas componentes pueden entorpecer procesos como la imputación de valores perdidos y la predicción. En este sentido las variables de consumo de calefacción son las que menos irregularidad presentan junto a la variable de ocupación, lo cual se debe principalmente a su mayor estacionalidad.
- Las variables con mayores niveles de autocorrelación son las que mayor componente de estacionalidad y menor irregularidad presentan (variables de consumo de calefacción, de temperatura y ocupación). Estas son las más predecibles, pues cada valor de las mismas está correlado con los valores anteriores. En particular, las variables H-F123-D1 y H-F123-D5/2 son las más predecibles, lo que es una buena noticia ya que serán las variables cuyos valores buscaremos predecir.

4.3.3. Tratamiento de outliers

En este apartado vamos a ver los métodos que hemos empleado sobre las variables de consumo para detectar y tratar sus outliers detectados en el apartado anterior, tras lo cual veremos los resultados sobre tres variables de consumo concretas.

4.3.3.1. Detección de outliers

En nuestro caso, hemos empleado un método para detectar outliers en cada una de nuestras variables que se puede encuadrar dentro del grupo de métodos de detección de outliers basados en umbral visto en el apartado 2.3.3.2. Este método de detección se ha aplicado sobre cada una de nuestras series temporales (variables) de forma individual y consta de los siguientes pasos:

1. En primer lugar se calculan los percentiles 1, 2, 3, 4 .. 98, 99, 100 de las observaciones de la variable.
2. En nuestras variables únicamente hay outliers positivos, ya que el valor mínimo en ellas es de 0. Además, como hemos visto en el apartado anterior hay muy pocos outliers (entorno a un 2 %-4 % de las observaciones de cada variable). Por tanto, nos fijamos en los valores de los percentiles 96, 97, 98, 99 y 100.
3. Entre dichos percentiles buscamos el mínimo percentil que tenga un valor anómalo (mucho mayor) en comparación con los anteriores percentiles. Dicho valor se denomina O.

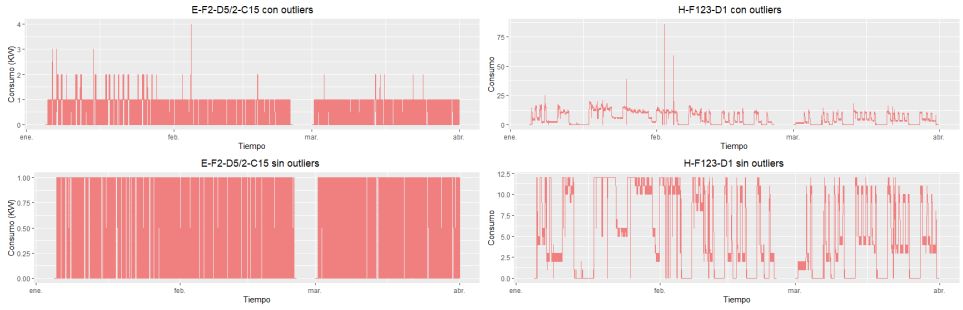
4. Todos los valores de la variable que superen o igualen el valor de O son considerados outliers.

Con esto conseguimos detectar en cada variable los valores anormalmente elevados en comparación con el resto. Se trata de un proceso que debe aplicarse de forma individual y manual a cada variable, pues depende directamente de la naturaleza y valores concretos de la misma.

4.3.3.2. Tratamiento de outliers

Tras detectar los outliers de cada variable se ha optado por sustituirlos por valores más regulares de la propia variable. Esta es una de las dos maneras de tratar los outliers que se comentó en el apartado 2.3.3.3, y se ha decidido aplicar sobre nuestras variables aunque tenga el principal inconveniente de que perdemos información, pues como dijimos en el apartado anterior en nuestras variables no hay muchos outliers, por lo que la pérdida de información no es tan significativa.

Dicho esto, los outliers de cada variable se han sustituido por los valores regulares más cercanos a estos. Más en concreto, todos los outliers se han sustituido por el mayor valor de percentil que no se considera outlier. Así, en la siguiente figura vemos tres variables antes y después de tratar sus outliers.



(a) E-F2-D5/2-C15.

(b) H-F123-D1.

(c) W-ALL.

Figura 4.8: Gráficas con y sin outliers de las variables E-F2-D5/2-C15, H-F123-D1 y W-ALL.

En la figura 4.8 podemos ver las gráficas con (arriba) y sin (abajo) outliers de las variables E-F2-D5/2-C15 (subfigura a), H-F123-D1 (subfigura b) y W-ALL (subfigura c). En este caso, vamos como tras eliminar los outliers de las variables, estas pasan a ser más regulares.

4.3.4. Tratamiento de valores perdidos

Tras llevar a cabo la detección y eliminación de outliers ya no estamos expuestos a los efectos perjudiciales de los mismos descritos en el apartado 2.3.3.1, por lo que se ha llevado a cabo adecuadamente el tratamiento de los valores perdidos de cada una de las variables de consumo.

En la sección 2.3.4 vimos qué eran los valores perdidos, su origen y las dos principales estrategias existentes para tratarlos: Eliminarlos e imputarlos (sustituirlos). En nuestro caso, hemos optado por la segunda estrategia ya que los valores perdidos rompen la continuidad de nuestras variables, lo cual es bastante problemático cuando tratamos con series temporales y además como se dijo en el apartado 4.3.2 no hay demasiados valores perdidos en cada variable.

En esta sección vamos en primer lugar a definir el problema de imputación de valores perdidos al que nos enfrentamos, para inmediatamente ver la experimentación que se ha llevado a cabo para tratar de imputar los va-

lores perdidos de cada una de las variables, la cual ha consistido en probar cuatro métodos de imputación multivariable y univariable recomendados por **[32]** para series temporales. Y finalmente expondremos las conclusiones de esta experimentación y veremos el método de imputación propio que finalmente hemos aplicado debido a que los demás métodos probados en la experimentación no nos han permitido realizar una imputación lo suficientemente buena.

4.3.4.1. Definición del problema de imputación

Como hemos visto en el apartado 4.3.2, todas y cada una de las variables de nuestro conjunto de datos, a excepción de las variables de ocupación y temperatura exterior, tienen perdidas todas las observaciones correspondientes a los días 26, 27, 28 y 29 de Febrero. Entonces, si cada día tiene 96 observaciones, esto significa que tenemos un total de 384 observaciones perdidas por cada variable.

Para solucionar este problema vamos a tratar de imputar los valores perdidos de cada variable para recuperar su continuidad y evitar la pérdida de 384 observaciones. Para ello, como hemos dicho previamente, vamos a llevar a cabo una experimentación con distintos métodos de imputación.

4.3.4.2. Experimentación

Para tratar de resolver el problema planteado llevaremos a cabo un proceso de experimentación consistente en aplicar sobre nuestras variables cuatro algoritmos de imputación multivariable y otros dos univariable recomendados por **[32]**, tras lo cual analizaremos los resultados. Más concretamente, por cada algoritmo a aplicar se seguirán los siguientes pasos:

1. Explicación breve del algoritmo de imputación.

2. Aplicación el mismo sobre el conjunto de datos.

3. Evaluación del algoritmo, contemplando los valores imputados de tres variables: Una de consumo eléctrico, otra de consumo de calefacción y la de consumo de agua. de esta forma evaluamos el algoritmo de imputación sobre cada tipo de variable de consumo, evitando tener que hacerlo sobre todas.

En lo que respecta a la evaluación de la imputación llevada a cabo por cada algoritmo, no tenemos una manera directa de realizarla. Pues no conocemos el verdadero valor de cada observación imputada. Sin embargo, podemos llevarla a cabo comprobando si los valores imputados por cada algoritmo cumplen o no con las siguientes condiciones.

1. El día 26 es Viernes y el 29 Lunes, por lo que los valores imputados para cada variable de consumo en esos días tienen que presentar valores de consumo propios de un día laboral. Por lo que de 7:30 a 16:00 deberían tener un consumo alto, mientras que el resto de horas debería ser bajo.

2. Los días 27 y 28 son Sábado y Domingo respectivamente, por lo que no trabaja nadie en el edificio. Así, los valores imputados para cada variable de consumo en estos días deben ser bajos.

3. Para cada variable, podemos obtener a partir de sus valores de contador, el consumo total registrado durante los días 26, 27, 28 y 29 calculando la diferencia entre el consumo total hasta el 1 de Marzo y el consumo total hasta el 25 de Febrero. Así, la suma de los valores imputados de cada variable deben estar próximos a dicho consumo total.

Dicho esto, vamos a aplicar los distintos métodos de imputación sobre las variables.

4.3.4.2.1 Amelia

Amélia [**38**] es un método de imputación de conjuntos de datos multivariable. Soporta dos formas distintas a la hora de imputar series temporales multivariable. Una consiste en imputar los valores perdidos de cada variable a partir de correlaciónes con versiones pasadas y futuras de la misma. La otra consiste en imputar los valores perdidos de cada variable a partir de su correlación con las demás y con versiones pasadas y futuras de ella misma. Se ha elegido la última solución, al ser más completa, dando lugar a los siguientes resultados.

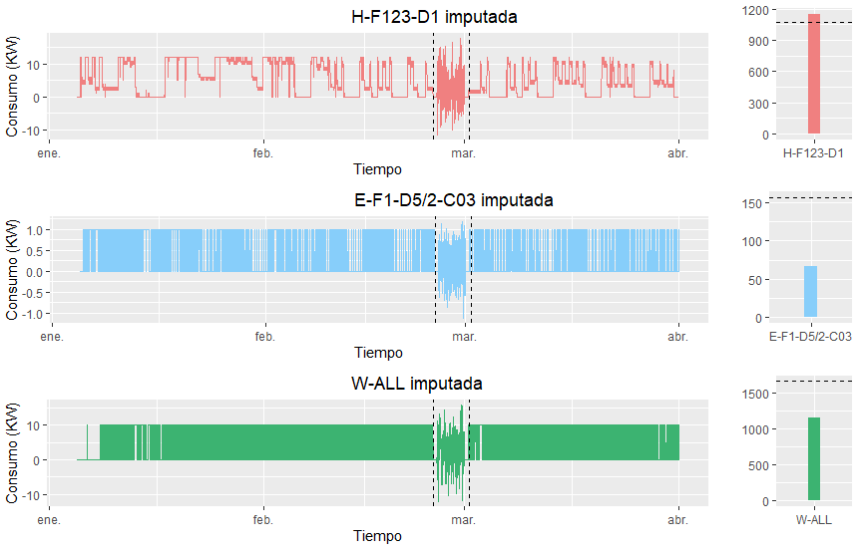


Figura 4.9: Gráficas de imputación con Amelia de los valores perdidos de las variables H-F123-D1, E-F1-D5/2-C03 y W-ALL.

En la figura 4.9 podemos ver las gráficas de las variables H-F123-D1 (rojo), E-F1-D5/2-C03 (Azul) y W-ALL (verde). Por cada variable tenemos dos gráficas, una (izquierda) en la que visualizamos los valores a lo largo del tiempo durante los meses de enero, febrero y marzo y otra (derecha) donde vemos una barra que nos indica la suma total de los valores imputados de dicha variable. Por último, la gráfica de la izquierda de cada variable presenta dos barras discontinuas verticales de color negro, las cuales indican el intervalo de los días 26, 27, 28 y 29 de Febrero, es decir, delimitan los valores imputados. Mientras, en la gráfica de la derecha vemos una línea discontinua horizontal de color negro que nos indica el consumo total verdadero registrado por la variable durante los días 26, 27, 28 y 29 de Febrero.

Se observa que los valores imputados mediante Amelia presentan oscilaciones muy bruscas y seguidas en las tres variables, lo que hace que el patrón de los valores imputados no se parezca a los patrones de consumo del resto de días. Además, en las tres variables vemos valores imputados negativos, lo cual es imposible desde el punto de vista del consumo eléctrico, de calefacción y de agua. Por todo esto, se puede concluir que no se cumplen las dos primeras condiciones propuestas previamente para la evaluación de los métodos de imputación. Tampoco se cumple la tercera condición, pues en las gráficas de la derecha de cada variable vemos que la suma total de los valores imputados está alejada (por arriba o por debajo) del consumo total verdadero.

4.3.4.2.2 Mtsdi

Mtsdi [**37**] es un paquete del lenguaje de programación R que proporciona un método de imputación de conjuntos de datos multivariable, el cual se puede aplicar sobre series temporales multivariable para las que la imputación de los valores perdidos de cada variable se realiza a partir de su correlación con el resto de variables y de un modelado de su estructura temporal.

De esta forma se obtienen los siguientes resultados al imputar los valores perdidos de las variables mediante Mtsdi.

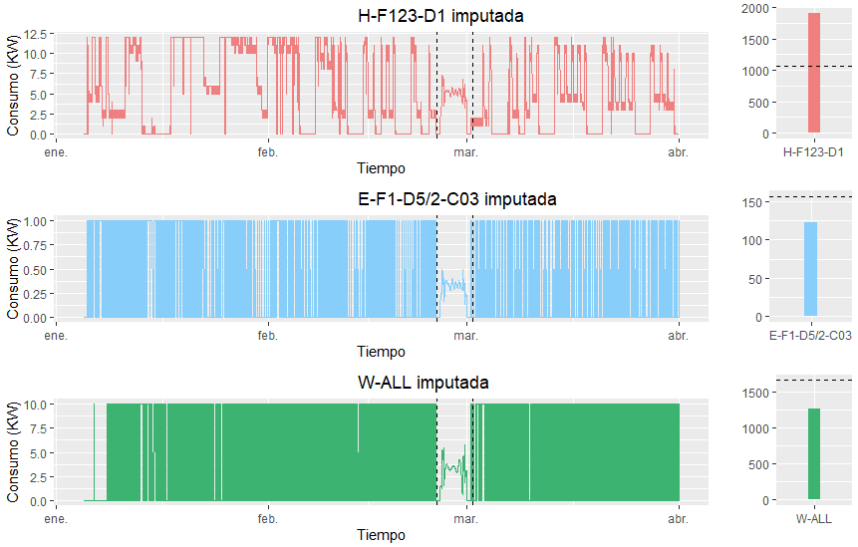


Figura 4.10: Gráficas de imputación con Mtsdi de los valores perdidos de las variables H-F123-D1, E-F1-D5/2-C03 y W-ALL.

En la figura 4.10 vemos que para las tres variables los patrones de consumo no oscilan tan bruscamente y no son tan negativos como con Amelia. Eso sí, vemos que el consumo se mantiene más o menos durante los días 26, 27, 28 y 29, lo que hace que no se cumplan las dos primeras condiciones. Por otra parte, en las gráficas de la derecha de cada variable se observa que el consumo total imputado se acerca bastante al consumo real en el caso de las variables E-F1-D5/2-C03 y W-ALL. Mientras que el consumo imputado para la variable H-F123-D1 se pasa bastante del que verdaderamente le corresponde.

4.3.4.2.3 Irmi

Irmi [**35**] es un método de imputación multivariable iterativo, el cual para imputar los valores perdidos de cada variable crea un modelo de regresión tomando la variable a imputar como objetivo y el resto como predictoras.

La imputación realizada con este método ofrece los siguientes resultados.

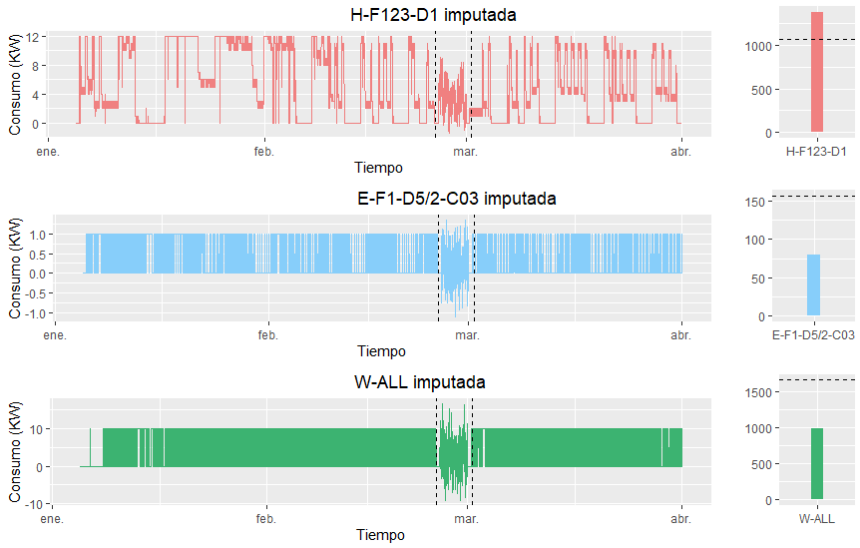


Figura 4.11: Gráficas de imputación con Irmi de los valores perdidos de las variables H-F123-D1, E-F1-D5/2-C03 y W-ALL.

En la figura 4.11 vemos en las gráficas de la izquierda que los valores imputados con Irmi son bastante parecidos a los imputados con Amelia (figura 4.9) para las tres variables, es decir, presentan oscilaciones muy bruscas además de valores negativos. Por tanto, este método tampoco cumple con las dos primeras condiciones. Por otro lado, en las gráficas de la derecha vemos que para cada variable, el consumo total imputado está alejado del verdadero consumo, lo que implica que este método de imputación tampoco cumple con la tercera condición.

4.3.4.2.4 MICE

MICE (Multivariate imputation by chained equations) **[36]** se trata de un método de imputación multivariable iterativo (al igual que Irmi), el cual imputa los valores perdidos de cada variable mediante un modelo (regresión lineal, regresión logística, regresión polinomial, etc) que tiene como objetivo la variable a imputar y el resto como variables predictoras. Los resultados de la imputación con este método son los siguientes.

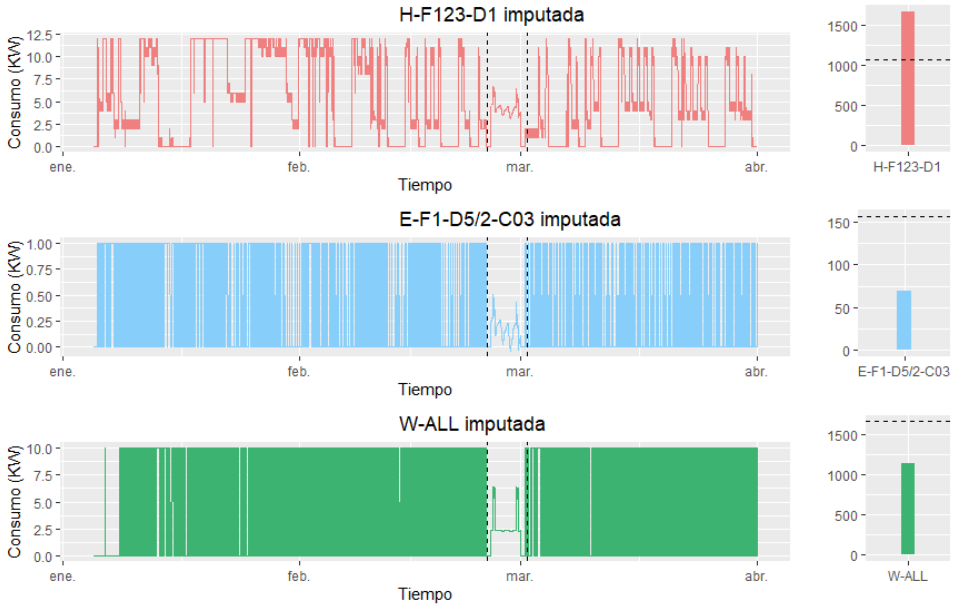


Figura 4.12: Gráficas de imputación con MICE de los valores perdidos de las variables H-F123-D1, E-F1-D5/2-C03 y W-ALL.

En la figura 4.12 vemos en las gráficas de la izquierda que para las tres variables los valores imputados no presentan oscilaciones bruscas o patrones negativos a diferencia de lo obtenido con Amelia (figura 4.9) o Irmi (4.11). Sin embargo, se observa que el consumo imputado permanece constante durante los días 26, 27, 28 y 29, por lo que no se cumplen las dos primeras condiciones. Por otra parte, en las gráficas de la derecha vemos que para las variables E-F1-D5/2-C03 y W-ALL el consumo total imputado es parecido al verdadero, al contrario de lo que ocurre con la variable H-F123-D1.

4.3.4.2.5 StructTS

StructTS **[33]** se trata de un método de imputación univariable, por lo que debe ser aplicado a cada una de nuestras variables de forma individual. Este método imputa los valores perdidos de una serie temporal aplicando un filtro estacional de Kalman. Al aplicar este método para imputar cada variable de nuestro conjunto de datos obtenemos lo siguiente.

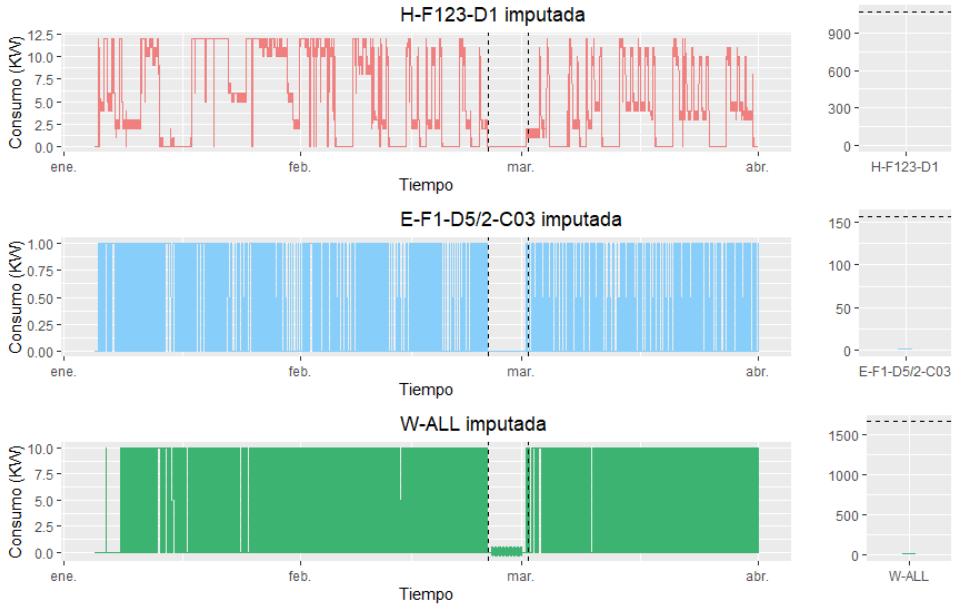


Figura 4.13: Gráficas de imputación con StructTS de los valores perdidos de las variables H-F123-D1, E-F1-D5/2-C03 y W-ALL.

En la figura 4.13 si observamos las gráficas de la izquierda vemos que los valores imputados para las tres variables son prácticamente nulos. Por lo tanto, el consumo total imputado también es 0 en las tres variables tal y como podemos observar en las gráficas de la derecha. así, tenemos que este método no cumple con ninguna de las tres condiciones. Aunque seo sí, hay que destacar que este método no se ha ejecutado en R con la frecuencia real de nuestras series temporales, la cual es de 672 (número de observaciones que conforman el período de estacionaliad, es decir, una semana) debido a que tarda demasiado en imputar una sola variable.

4.3.4.2.6 Interp

Interp [**74**] es un método de imputación univariable que aplica una interpolación lineal para imputar series temporales sin estacionalidad y una descomposición STL para imputar series temporales estacionales. Más en concreto, a la hora de imputar los valores perdidos de una serie temporal, este método quita la componente estacional y aplica una interpolación lineal sobre la componente restante para imputar sus valores perdidos y finalmente añade de nuevo la componente estacional. De esta manera al aplicar este método sobre nuestro conjunto de datos obtenemos los siguientes resultados.

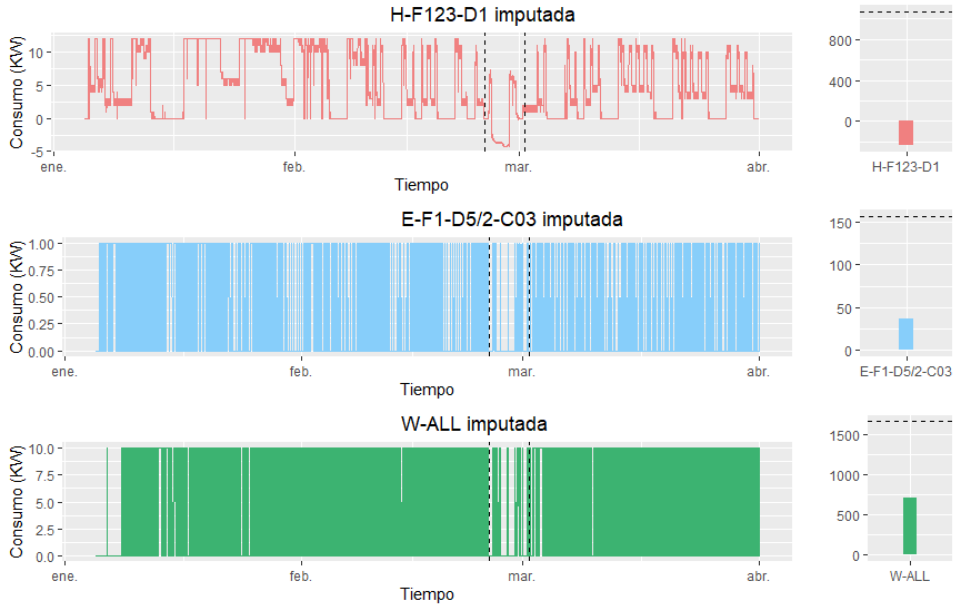


Figura 4.14: Gráficas de imputación con Interp de los valores perdidos de las variables H-F123-D1, E-F1-D5/2-C03 y W-ALL.

En la figura 4.14 se observa en las gráficas de la izquierda que los valores imputados mediante este método siguen un patrón más o menos parecido al de los demás valores de consumo. Esto indica una mejor captación del componente estacionario de cada variable por parte de este método. Por otra parte, también vemos que más o menos se cumplen las dos primeras condiciones, pues los días 27 y 28 hay poco consumo, mientras que en los días 26 y 29 el consumo sigue más o menos el patrón de un día laboral. Sin embargo, a pesar de todo esto, vemos como los valores imputados para los días 27 y 28 en la variable H-F123-D1 son negativos y además, en las gráficas de la derecha vemos que en las tres variables el consumo total imputado es bastante inferior al verdadero, por lo que este método no cumple con la tercera condición.

4.3.4.2.7 Conclusiones

Ante la experimentación llevada a cabo, podemos extraer las siguientes conclusiones:

- En lo que se refiere a los métodos de imputación multivariable hemos visto que ninguno cumple con las tres condiciones. Esto puede estar causado por el hecho de que estos métodos imputan una observación concreta de una variable a partir de sus valores pasados (solo algunos de los métodos multivariable) y de los valores del resto de variables, pero como dijimos en el apartado 4.3.4.1, el valor del resto de variables

también esta perdido (en el mismo instante de tiempo de la observación que tratamos de imputar) a excepción de las variables de ocupación y temperatura exterior. Por tanto, no podemos explotar al máximo la capacidad de estos métodos.

- En cuanto a los métodos univariable hemos comprobado que el método StructTS no funciona bien debido a que no podemos ejecutarlo con la frecuencia verdadera de nuestras series temporales (variables). Mientras que el método interp es el que mejor rendimiento ofrece, pero está lejos de cumplir con la tercera condición y ofrece valores negativos.

4.3.4.2.8 Imputación final

De acuerdo con las conclusiones extraídas en el apartado anterior, no hemos podido encontrar un método de imputación adecuado para nuestro problema particular. Por eso, finalmente se ha optado por imputar los valores perdidos de las variables de consumo mediante un método más simple basado en una hipótesis extraída a partir de todas las observaciones que hemos hecho hasta este momento sobre nuestros datos. Esta hipótesis consiste en que en los días similares de dos semanas adyacentes se registran patrones de consumo similares en cada variable. Esto es así, porque el clima suele ser similar de una semana para otra, la ocupación en el mismo día de dos semanas adyacentes suele ser la misma debido a que en el edificio están los mismos grupos trabajando en los mismos proyectos, etc. El método de imputación ha constado de los siguientes pasos:

1. Se obtuvo el consumo total registrado por cada variable durante los días 26, 27, 28 y 29 de febrero a partir de sus valores contador.

2. Este consumo de cada variable se comparó con su consumo total durante los mismos días de la anterior semana y de la siguiente, calculando la diferencia.

3. A partir de estas diferencias se obtuvieron las diferencias absolutas entre el consumo total de los días 26, 27, 28 y 29 y el consumo de los días homólogos de la semana pasada y la siguiente.

4. Así se obtuvo que la diferencia de consumo con la semana anterior era de 59 mientras que con la siguiente era de 2512. Por tanto, se determinó, que el patrón de consumo de las variables en los días 26, 27, 28 y 29 se parecía más al de los mismos días de la semana pasada.

5. Se sustituyeron pues los valores perdidos de cada variable por los valores de los días 19, 20, 21 y 22 de Febrero, es decir, por los valores de una semana antes.

Con este método de imputación se han obtenido los siguientes resultados.

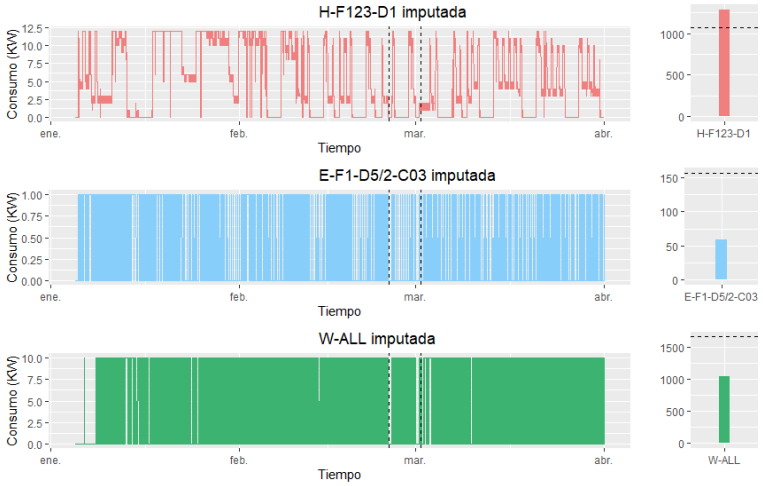


Figura 4.15: Gráficas de imputación mediante patrones similares de los valores perdidos de las variables H-F123-D1, E-F1-D5/2-C03 y W-ALL.

En la figura 4.15 vemos en las gráficas de la izquierda que los patrones de consumo imputados son ahora muy parecidos al resto, pues son los mismos que los la semana pasada y además puede verse como cumplen con las dos primeras condiciones. Por otra parte, en las gráficas de la derecha vemos que este método no cumple del todo con la tercera condición pero se acerca bastante.

4.3.5. Selección de variables

Finalmente, tras transformar, tratar los outliers e imputar los valores perdidos de cada una de las variables del conjunto de datos, vamos a seleccionar las variables más relevantes para predecir las variables objetivo H-F123-D1 y H-F123-D5/2. Estas dos variables se van a tratar de predecir por separado, por lo que la selección de variables de cada una se llevará a cabo también por separado, es decir, en primer lugar seleccionaremos las variables más relevantes para predecir H-F123-D1, para continuar después haciendo lo mismo con la variable H-F123-D5/2. Dicho esto, los pasos a seguir en esta sección son los siguientes:

1. En primer lugar calcularemos la correlación cruzada entre cada par de variables del conjunto de datos y analizaremos los niveles de correlación obtenidos mediante la visualización de un mapa de calor.

2. Tras esto, estudiaremos cuáles son las variables más relevantes para poder predecir cada variable objetivo haciendo uso del conocimiento

extraído del paso anterior y del algoritmo XGBoost mencionado en el apartado 2.3.5.1.2.

3. Para la variable H-F123-D1 se agruparán según su nivel de correlación las variables más relevantes para su predicción y de cada grupo de variables con una correlación elevada entre ellas nos quedaremos únicamente con una para eliminar información redundante.

4. Se repetirá el paso 3 para la variable H-F123-D5/2.

Vamos pues a empezar viendo la correlación entre cada par de variables del conjunto de datos.

4.3.5.1. Análisis de la correlación entre variables

Para llevar a cabo el análisis de la correlación entre variables se ha calculado entre cada par de variables A y B todas las correlaciones posibles rotando A hacia la derecha y quedándonos finalmente con el valor de correlación más fuerte (positivo o negativo) entre ambas, es decir, entre cada par de variables calculamos la correlación cruzada, la cual ya fue explicada en el apartado 2.3.5.1.1.2.

Así, se ha construido una matriz de correlaciones, y a partir de ella un mapa de calor, el cual vemos a continuación.

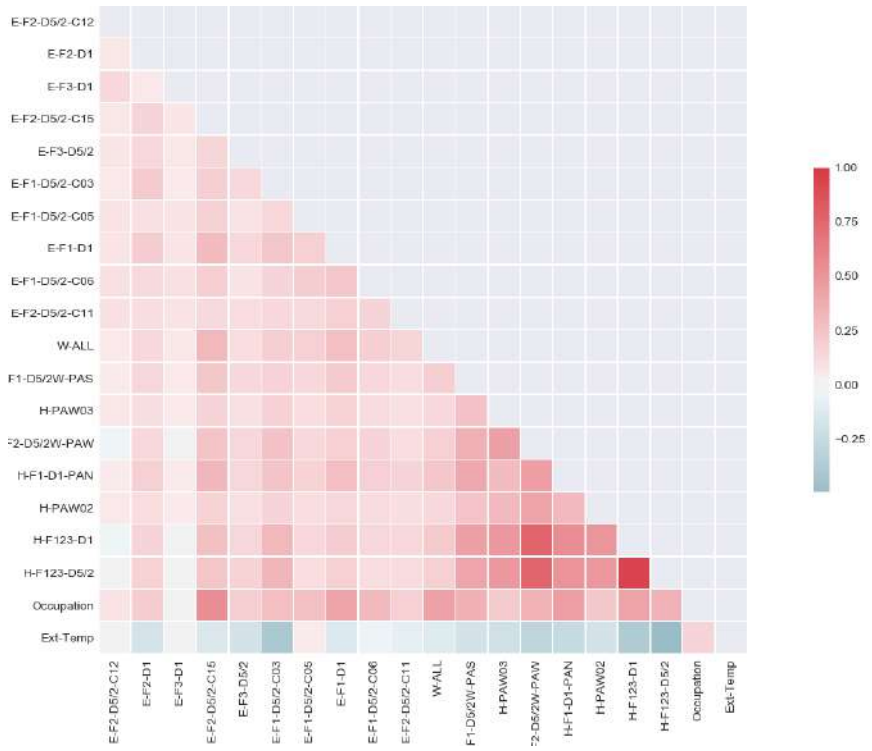


Figura 4.16: Mapa de calor que indica la correlación entre las variables.

En la figura 4.16 podemos ver el mapa de calor que nos indica de forma visual el nivel e correlación entre cada par de variables. Así, cuanto más roja sea una casilla correspondiente a dos variables más positiva será la correlación entre las mismas y cuanto más azul sea más negativa será la correlación indicando un color banco que ambas variables son independientes.

Tanto en la tabla como en el mapa de calor de correlaciones podemos observar lo siguiente:

1. No hay mucha dependencia entre los sensores eléctricos, lo cual puede deberse a que como ya hemos dicho anteriormente cada sensor eléctrico está colocado en una zona muy concreta, y de una zona a otra pueden variar el número de trabajadores, el equipamiento eléctronico, las horas de luz natural, etc, que son factores que influyen directamente en el consumo de electricidad de la zona. Además, como se puede ver vemos en el apéndice A.1 que el consumo registrado por los sensores eléctricos es bastante irregular, lo cual también propicia los bajos niveles de correlación entre variables eléctricas. Todo esto, también puede explicar la baja correlación de las variables eléctricas, en general, con el resto.

2. Hay una correlación elevada entre las variables de consumo de cale-

facción. Esto encaja con lo que se dijo en el apartado 4.3.2, es decir, que las variables de consumo de calefacción presentan unos patrones mucho más regulares y tienen una mayor autocorrelación. A parte de esto, los elevados niveles de correlación entre estas variables también se debe a que el consumo registrado por algunas de ellas está incluido dentro del consumo registrado por otras, tal y como se comentó en el apartado 3.3.2.

3. La variable de consumo de agua no está en general correlada con el resto de variables a excepción de la variable de ocupación con la cual tiene un nivel de correlación de 0.43. Esto tiene sentido, pues cuanta más alta sea la ocupación del edificio más agua se consumirá y viceversa.

4. La variable de ocupación es la que más correlada está en general con el resto de variables, teniendo en la mayoriá de los casos un nivel de correlación superior al 0.20. Esto es lógico, pues cuanta más ocupación haya en el edificio más consumo eléctrico, de calefacción y de agua habrá. sin embargo, estos niveles de correlación no son tan elevados como cabría esperar, lo cual puede deberse entre otras causas a que:

   - La variable occupation es una estimación de la ocupación global del edificio, por lo que la ocupación local verdadera de una zona concreta puede que sea distinta y vaya en disonancia con la ocupación global
   - Algunos sensores registran consumo incluso en horas en las que el edificio está vacío.
   - Durante la jornada laboral la ocupación tiene niveles elevados pero constantes al tiempo que las variables de consumo aumentan y disminuyen sus valores constantemente.

5. La variable Ext-Temp está correlada en sentido negativo con el resto de variables y en especial con las de consumo de calefacción, pues cuanto menor es la temperatura más calefacción se consume para mantener una temperatura agradable en el interior del edificio.

6. Las variables objetivo H-F123-D1 y H-F123-D5/2 están muy correladas entre sí. Tanto, que son prácticamente idénticas. Esto hace que tengan ambas niveles similares de correlación con el resto de variables, tal y como se aprecia en el mapa de calor.

Tras este análisis vamos a ver cuáles son las variables más relevantes a la hora de predecir H-F123-D1 y H-F123-D5/2.

4.3.5.2. Selección de variables relevantes para la predicción

Como hemos dicho, vamos a ver cuales son las variables más relevantes para predecir tanto H-F123-D1 como H-F123-D5/2. Para ello, haremos por cada una de ellas lo siguiente:

1. Visualizaremos en una gráfica de barras su correlación con cada variable predictora, para así ver con cuales tiene mayor relación.

2. Con el algoritmo XGBoost entrenaremos un modelo a partir de todo nuestro conjunto de datos que sea capaz de predecir la variable objetivo (H-F123-D1 o H-F123-D5/2) en función del resto de variables, tras lo cual visualizaremos mediante otro gráfico de barras cuales han sido las variables predictoras más relevantes en dicho modelo. En este caso, cabe destacar que para ambas variables objetivo, el modelo de XGBoost se entrena generando 10000 árboles de decisión, cada uno de ellos con una profundidad máxima de 6, que son los parámetros recomendados por el propio algoritmo.

3. Con lo observado en los pasos 1 y 2 determinaremos cuáles son las variables más relevantes para predecir cada variable objetivo.

Vamos a llevar a cabo estos tres pasos en primer lugar con la variable H-F123-D1.

4.3.5.2.1 Variable H-F123-D1

En el caso de la variable H-F123-D1 vemos en la siguiente gráfica de barras su correlación con el resto de variables además de con ella misma. Esta última correlación también la visualizamos porque venimos diciendo a lo largo de este trabajo, en nuestro problema de predicción queremos predecir los valores futuros de la variable objetivo a partir de sus propios valores pasados además de los del resto de variables. También cabe destacar que las correlaciones están en valor absoluto para simplificar su visualización.

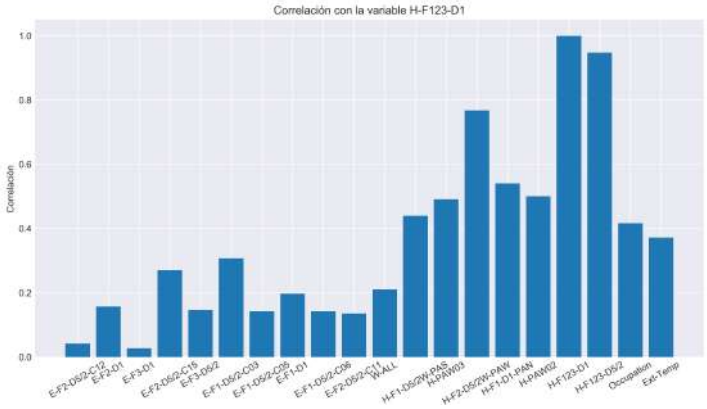


Figura 4.17: Correlación de la variable H-F123-D1 con el resto.

En la gráfica de la figura 4.17 podemos ver que la variable H-F123-D1 se encuentra correlada con las variables de consumo de calefacción y con las de ocupación y temperatura, exhibiendo un nivel de correlación superior a 0.4 con todas ellas. En este caso, si consideramos que aquellas variables que tienen un nivel de correlación con la variable H-F123-D1 superior a 0.3 son relevantes para predecirla, tendríamos que además de las anteriores, también son relevantes las variables E-F2-D5/2-C15 y E-F1-D5/2-C03.

Dicho esto, vamos a ver cuáles son las variables más relevantes para predecir H-F123-D1 según XGBoost.

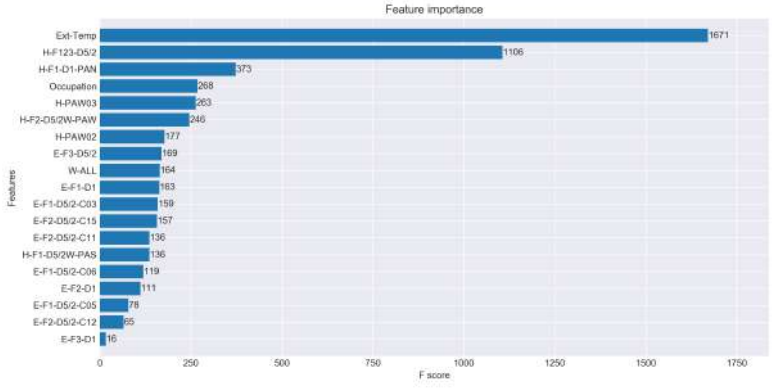


Figura 4.18: Variables más importantes para predecir H-F123-D1 según XGBoost.

En la figura 4.18 podemos ver una gráfica horizontal de barras con la importancia de cada variable para predecir H-F123-D1 según el algoritmo XGBoost. Esta importancia se mide como el número total de veces que cada

variable ha sido escogida para separar los datos en todas las iteraciones de boosting (árboles) durante el entrenamiento del modelo.

Vemos en la gráfica que las variables más relevantes para predecir H-F123-D1 son Occupation, Ext-Temp y las variables de consumo de calefacción, lo que coincide con lo visto en la figura 4.17. Dicho esto, llama la atención la enorme importancia que de la variable Ext-Temp, la cual es la más importante según XGBoost a pesar de no tener un gran nivel de correlación con H-F123-D1.

También vemos que ahora para XGboost las variables H-D2-D5/2W-PAS, E-F1-D5/2-C03 y E-F2-D5/2-C15 no son tan relevantes como sugieren los niveles de correlación.

Por lo tanto, para predecir la variables H-F123-D1 vamos a:

- Descartar todas las variables de consumo eléctrico, pues la mayoría tiene muy poca correlación con H-F123-D1 y las que tienen un nivel superior a 0.3 no son tan importantes según XGBoost.
- Quedarnos con las variables de consumo de calefacción a excepción de H-F1-D5/2W-PAS ya que no es tan importante según XGBoost a pesar de tener un nivel de correlación de 0.4 con H-F123-D1. Cabe destacar que de las variables de consumo de calefacción también nos quedamos con la propia variable H-F123-D1 y con H-F123-D5/2, la cual es la otra variable objetivo.
- Quedarnos con las variables de ocupación (Occupation) y temperatura exterior (Ext-Temp), pues como hemos visto son muy relevantes.

4.3.5.2.2 Variable H-F123-D5/2

En el caso de la variable H-F123-D5/2 tenemos que los niveles de correlación con el resto de variables y con ella misma son los que vemos en la siguiente gráfica.

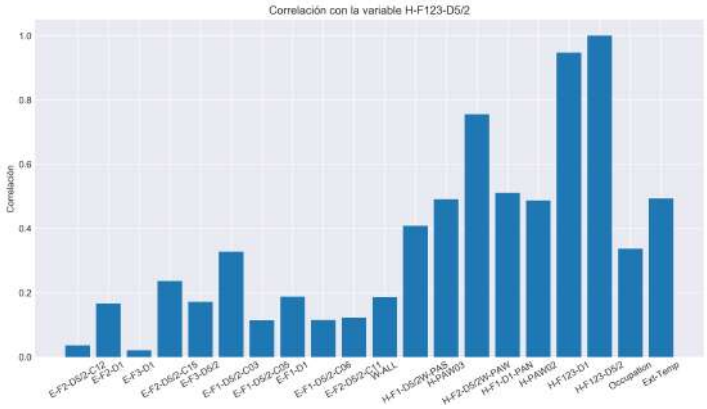


Figura 4.19: Correlación de la variable H-F123-D5/2 con el resto.

En la figura 4.19 vemos que la variable H-F123-D5/2 tiene los mismos niveles de correlación con el resto de variables que los que tenía la variable H-F123-D1. Esto es así, porque HF123-D1 y H-F123-D5/2 son prácticamente idénticas al tener un nivel de correlación de 0.95.

Ahora vamos a ver cuales son las variables más importantes para predecir H-F123-D5/2 según XGBoost.

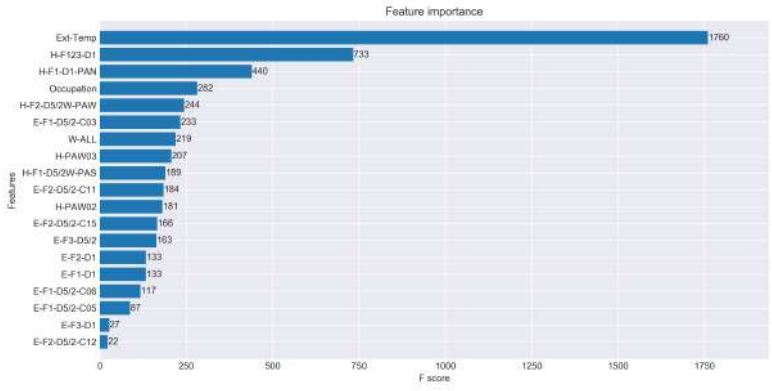


Figura 4.20: Variables más importantes para predecir H-F123-D5/2 según XGBoost.

En la figura 4.20 vemos que obtenemos unos resultados muy parecidos a los obtenidos para la variable H-F123-D1, salvo que en este caso XGBoost le da una mayor importancia a las variables E-F1-D5/2-C03 y W-ALL y menos a la variable H-PAW02.

Por lo tanto, para predecir la variable H-F123-D5/2 vamos a:

- Quedarnos con las variables de consumo de calefacción, pues tienen

una alta correlación con H-F123-D5/2, a excepción de H-PAW02 y H-F1-D5/2W-PAS las cuales no son tan importantes según XGBoost. Como ocurría con H-F123-D1 aquí incluimos las variables H-F123-D1 y H-F123-D5/2.

- Descartar todas las variables eléctricas, pues tienen baja correlación con H-F123-D5/2 y poca importancia según XGBoost, a excepción de la variable E-F1-D5/2-C03 que tiene un nivel de correlación superior a 0.3 y una importancia más elevada.
- Descartar la variable W-ALL, pues aunque se importante de acuerdo con XGBoost, su correlación con H-F123-D5/2 es menor que 0.3.
- Quedarnos con las variables Ext-Temp y Occupation, al poseer estas niveles de correlación superiores a 0.3 con H-F123-D5/2 y estar ambas entre las cuatro variables más relevantes según XGBoost.

4.3.5.2.3 Resumen

Finalmente vemos las variables más relevantes para predecir H-F123-D1 y H-F123-D5/2 en la siguientes tablas.

Variable	*Correlación con H-F123-D1*	*Importancia según XGBoost*
H-F123-D1	1	Máxima
H-F123-D5/2	0.95	1106
Ext-Temp	-0.37	1671
H-F1-D1-PAN	0.54	373
Occupation	0.42	268
H-PAW03	0.49	263
H-F2-D5/2W-PAW	0.77	246
H-PAW02	0.5	177

Tabla 4.7: Variables más relevantes para predecir H-F123-D1.

Variable	*Correlación con H-F123-D5/2*	*Importancia según XGBoost*
H-F123-D5/2	1	Máxima
H-F123-D1	0.95	733
Ext-Temp	-0.49	1770
H-F1-D1-PAN	0.51	440
H-F2-D5/2W-PAW	0.76	244
E-F1-D5/2-C03	0.33	233
H-PAW03	0.49	207
Occupation	0.34	282

Tabla 4.8: Variables más relevantes para predecir H-F123-D5/2.

En las tablas 4.7 y 4.8 vemos las variables más relevantes para predecir H-F123-D1 y H-F123-D5/2 respectivamente. En ambas se observa en cada fila una variable relevante junto a su nivel de correlación con la variable objetivo correspondiente y a su nivel de importancia para la predicción de la misma según XGBoost.

Se puede observar que se han elegido las mismas variables para predecir H-F123-D1 y H-F123-D5/2, salvo una diferencia. Para predecir H-F123-D1 se ha elegido la variable H-PAW02, mientras que para predecir H-F123-D5/2 se ha elegido E-F1-D5/2-C03 en su lugar.

4.3.5.3. Eliminación de variables redundantes

Tras seleccionar las variables predictoras más relevantes de H-F123-D1 y H-F123-D5/2, vamos a ver la correlación entre dichas variables predictoras con el propósito de eliminar aquellas que posean información muy similar, es decir, información redundante. Así, consultando la matriz de correlaciones vemos lo siguiente.

- En el caso de la variable H-F123-D1, H-F2-D5/2W-PAW y H-F123-D5/2 son las que más correladas están (tiene sentido, pues el consumo de la primera está incluido en la segunda) con un nivel de 0.76. Por tanto, nos aportan información muy similar para la predicción, asi que nos vamos a quedar con H-F123-D5/2 ya que posee una mayor correlación con H-F123-D1.

  En cuanto al resto de variables no se han encontrado grupos que estén correlados con un nivel superior a 0.5, por lo que no se puede considerar que contengan información redundante.

- En cuanto a la variable H-F123-D5/2 ocurre algo parecido que con H-F123-D1, es decir, entre las variables predictoras H-F2-D5/2W-PAW y H-F123-D1 tienen una correlación de 0.77, razón por la cual nos quedamos con H-F123-D1 y descartamos H-F2-D5/2W-PAW al tener la primera una mayor correlación con H-F123-D5/2. Para el resto de variables, también sucede que no hay ningún grupo con un nivel de correlación superior a 0.5.

Así, finalmente vemos en las siguientes dos tablas la selección definitiva de variables predictoras de H-F123-D1 y H-F123-D5/2.

Variable	Correlación con H-F123-D1	Importancia según XGBoost
H-F123-D1	1	Máxima
H-F123-D5/2	0.95	1106
Ext-Temp	-0.37	1671
H-F1-D1-PAN	0.54	373
Occupation	0.42	268
H-PAW03	0.49	263
H-PAW02	0.5	177

Tabla 4.9: Variables seleccionadas para predecir H-F123-D1.

Variable	Correlación con H-F123-D5/2	Importancia según XGBoost
H-F123-D5/2	1	Máxima
H-F123-D1	0.95	733
Ext-Temp	-0.49	1770
H-F1-D1-PAN	0.51	440
E-F1-D5/2-C03	0.33	233
H-PAW03	0.49	207
Occupation	0.34	282

Tabla 4.10: Variables seleccionadas para predecir H-F123-D5/2.

Capítulo 5

Modelos de predicción

En este capítulo se describe el problema de predicción a resolver y su división en tres subproblemas distintos, tras lo cual veremos el proceso de experimentación llevado a cabo para la obtención de distintos modelos de predicción para abordar cada problema y sus resultados.

En el capítulo anterior (capítulo 4) hemos visto el proceso de preprocesamiento de los datos llevado a cabo. Como resultado final de este proceso tenemos una serie temporal multivariable con los valores en los meses de Enero, Febrero y Marzo de 2016 de 20 variables, de las cuales tenemos a su vez dos subconjuntos, uno para predecir la variable H-F123-D1 y otro para la variable H-F123-D5/2. De esta manera, en este capítulo vamos a describir cómo se ha abordado el problema de predecir ambas variables y los resultados obtenidos, para lo cual se ha dividido el capítulo en tres secciones principales: En la sección 7.1 definiremos de una manera más detallada el problema de predicción de cada variable y cómo este se ha dividido a su vez en tres subproblemas más pequeños de predicción. En la sección 7.2 veremos los pasos necesarios para poder entrenar un modelo con la finalidad de abordar un problema concreto de predicción. En la sección 7.3 describiremos de forma genérica el proceso llevado a cabo para obtener el mejor modelo de cada arquitectura comentada en el apartado 4.4.3 y de XGBoost para resolver cada subproblema de predicción para finalmente ver el rendimiento que han ofrecido estos mejores modelos a la hora de predecir en cada subproblema las variables H-F123-D1 y H-F123-D5/2.

5.1. Definición del problema

Como se comentó en el capítulo 1, el propósito de este trabajo es ver si mediante técnicas de aprendizaje profundo se puede o no predecir el consumo

futuro de calefacción del edificio ICPE a partir de valores históricos de varias variables que pudieran influir en el mismo (como los valores pasados del propio consumo de calefacción). Así, nuestro problema es ahora ver si se puede o no predecir el consumo de calefacción futuro en las áreas D1 y D5/2 del edificio ICPE en las tres plantas, es decir, los valores de las variables H-F123-D1 y H-F123-D5/2. Para ello, en el anterior capítulo (capítulo 6) se seleccionaron las variables más relevantes (tablas 4.9 y 4.10) cuyos valores pasados nos van a permitir predecir mejor los valores futuros de ambas variables. Por tanto, disponemos de dos series temporales, una con los valores de Enero, Febrero y Marzo de las variables predictoras de H-F123-D1 y otra con los valores de Enero, Febrero y Marzo de las variables predictoras de H-F123-D5/2. Entonces tenemos dos problemas de predicción que vamos a tratar de forma independiente.

Cada uno de estos dos problemas va a consistir en predecir los valores futuros de la variable objetivo (H-F123-D1 y H-F123-D5/2) en las siguientes 12 horas (medio dia) cada 15 minutos a partir de los valores de los anteriores 4 días de sus variables predictoras. Más específicamente, queremos obtener a partir de una serie temporal con 384 observaciones (4 días) que contienen en orden cronológico los valores de las variables predictoras, los valores de la variable objetivo en los siguientes 48 instantes de tiempo (12 horas). En este caso, la cantidad de observaciones que utilizamos o que miramos hacia el pasado y hacia el futuro se denominan lookback y delay respectivamente. Empleamos un lookback de 384 instantes de tiempo o de 4 días porque con lo visto en las gráficas de autocorrelación de ambas variables objetivo (sección 4.3.2) y mediante las distintas experimentaciones, se ha determinado que los valores futuros están bastante correlados con los valores pasados de hasta 4 días, además este lookback nos permite «saltarnos» los fines de semana en el caso de que estemos prediciendo los valores de consumo de calefacción de un lunes o un martes. Puesto que si por ejemplo, utilizásemos un lookback de 2 días y estuviéramos prediciendo el consumo de calefacción en un lunes, los valores de consumo pasados serían nulos al pertenecer al sábado y al domingo por lo que el modelo tendería a predecir que el consumo de calefacción del lunes también es nulo. Mientras que se ha elegido un delay de 48 instantes de tiempo o de 12 horas porque se cree que es un horizonte suficiente como para que las predicciones permitan operar el edificio de manera eficiente.

Dicho esto, de acuerdo con [**45**], para entrenar modelos de XGBoost y RNA para predecir tanto H-F123-D1 como H-F123-D5/2 es necesario tomar de nuestra serie temporal las primeras k observaciones para entrenar y los restantes para validar respetando el orden temporal. Así por ejemplo, podríamos tomar los instantes de tiempo de Enero y Febrero para entrenar los modelos y los de Marzo para validar. Sin embargo, esta división no nos va a proporcionar modelos con demasiado rendimiento, pues como se ve en las gráficas de las variables H-F123-D1 y H-F123-D5/2 (apéndice A.2, figura A.11) sus valores en los meses de Enero y Febrero difieren bastante de sus

valores en el mes de Marzo, por lo que el modelo que obtendríamos sabría predecir sus valores en los meses de Enero y Febrero, pero no en Marzo. Esto se podría solucionar entrenando con datos de los meses de Enero, Febrero y Marzo de otros años, pero como sabemos solo disponemos de datos del año 2016. Por tanto, para tratar de solucionar este inconveniente se ha divido el problema en tres subproblemas más pequeños:

- Problema de Enero: Para este problema se han cogido las observaciones que van desde el 2016-01-05 00:15:00 hasta el 2016-01-21 23:45:00 para entrenar y el resto hasta el 2016-01-31 23:45:00 para validar.
- Problema de Febrero: Para este problema se han cogido las observaciones que van desde el 2016-02-01 00:00:00 hasta el 2016-02-21 23:45:00 para entrenar y el resto hasta el 2016-02-29 23:45:00 para validar.
- Problema de Marzo: Para este problema se han cogido las observaciones que van desde el 2016-03-01 00:00:00 hasta el 2016-03-21 23:45:00 para entrenar y el resto hasta el 2016-03-31 23:45:00 para validar.

De esta forma, hemos convertido cada uno de los dos problemas de predicción originales en tres problemas más pequeños, cada uno encuadrado en un mes concreto. Así, apaciguamos el inconveniente anteriormente comentado, aunque este no se resolverá del todo ya que incluso en un mismo mes hay diferencias entre los primeros y últimos valores de las variables H-F123-D1 y H-F123-D5/2. Por otra parte, vemos que en cada mes entrenamos con las observaciones de los primeros 21 días (excepto en enero que como sabemos le faltan los 5 primeros) y validamos con las observaciones restantes de ese mismo mes. De esta forma utilizamos aproximadamente para entrenar y validar el 70 % y el 30 % (proporciones habituales) respectivamente de las observaciones de cada mes.

5.2. Proceso de entrenamiento y validación

Una vez definidos los diferentes problemas de predicción de las variables H-F123-D1 y H-F123-D5/2, vamos a describir las distintas tareas necesarias para llevar a cabo el proceso de entrenamiento y validación tanto de un modelo XGBoost como de cualquiera de las arquitecturas de RNA que dijimos en el apartado 2.4.3 que íbamos a utilizar, con el propósito de resolver cualquiera de los tres problemas de predicción (Enero, Febrero o Marzo) de una de las dos variables objetivo (H-F123-D1 o H-F123-D5/2). Estas tareas son:

1. Normalización de los datos.
2. Generación de muestras de entrenamiento y validación.
3. Definición del modelo.

4. Elección del algoritmo de optimización (en el caso de las RNA) y de una función de error.

5. Entrenamiento y validación.

Vamos a ver con más detalle estas tareas en las siguientes subsecciones.

5.2.1. Normalización de los datos

La normalización de un conjunto de datos es un proceso consistente en centrar todos los valores da cada variable de dicho conjunto de manera que tengan media 0 y desviación típica 1. Esto se consigue calculando la media y la desviación típica de cada variable y a continuación restando a todos sus valores la media y dividiéndolos entre la desviación típica. Esta es una operación muy común a la hora de entrenar modelos de predicción y se aplica sobre conjuntos de datos en los que hay variables con distintas escalas de valores (como en nuestra serie temporal), pues los modelos de predicción suelen darle una mayor importancia o peso durante el entrenamiento a las variables con una escala mayor. De esta forma, si normalizamos el conjunto de datos le damos la misma importancia en principio (luego será el modelo el que le de más peso a una o a otras) a todas las variables.

En nuestro caso, a la hora de predecir o bien H-F123-D1 o bien H-F123-D5/2 en un mes concreto hacemos lo siguiente.

1. Del conjunto de datos o serie temporal para predecir la variable objetivo correspondiente, nos quedamos únicamente con las observaciones de todas las variables que pertenezcan al mes en el cual la queremos predecir.

2. Cogemos las observaciones de entrenamiento (primeras k) de todas las variables y con ellas calculamos la media y la desviación típica de cada variable.

3. Normalizamos todos los valores de cada variable con su media y desviación típica obtenidas con sus valores de entrenamiento.

Cabe señalar que se normalizan los valores de cada variable empleando la media y desviación típica solo de sus valores de entrenamiento para evitar que la información de los valores de validación se infiltre en los valores de entrenamiento a través de la media y la desviación típica.

5.2.2. Muestras de entrenamiento y validación

Tras normalizar los valores de las variables del mes concreto en el que vamos a predecir la variable objetivo, tenemos que generar a partir de las

observaciones de entrenamiento y de validación las series temporales o muestras de entrenamiento, con las cuales nuestro modelo aprenderá a predecir, y de validación, con las que evaluaremos el rendimiento del mismo. Así tenemos que:

- Cada muestra de entrenamiento se genera a partir de una observación o_t de entrenamiento concreta tomando las lookback observaciones anteriores a ella (incluida), obteniendo una muestra de ejemplo que le pasaremos al modelo como entrada para que aprenda. Además también se toman los valores de la variable objetivo situados en las siguientes delay observaciones después de o_t (sin incluir), estos valores son los que queremos que idealmente nos de como salida el modelo al pasarle como entrada la muestra de entrenamiento generada.
- Cada observación de validación también da lugar a una muestra de validación de la misma manera que sucede con las observaciones de entrenamiento.

Ahora bien en el mes concreto en el que estamos tratando el problema de predicción no se pueden generar muestras con todas las observaciones de entrenamiento y de validación, pues:

- Para generar una muestra de entrenamiento a partir de una observación o_t, esta tiene que tener al menos lookback observaciones que le precedan y que pertenezcan al mismo mes. Esto se hace así para que todas las observaciones de todas las muestras de entrenamiento pertenezcan al mismo mes en el cual estamos afrontando el problema de predicción. Además de esto, el o_t también debe tener por delante al menos delay observaciones de entrenamiento para sí evitar aprender a predecir valores de validación.
- A la hora de generar una muestra de validación a partir de una observación o_t, esta únicamente tiene que tener por delante delay observaciones que pertenezcan al mismo mes y que sean de validación.

Estas restricciones en la creación de muestras de entrenamiento y validación van encaminadas a evitar mezclar observaciones de entrenamiento con observaciones de validación y observaciones del mes actual con observaciones del mes anterior o el siguiente, para de esta forma asegurarnos de que el proceso entrenamiento y validación de cada modelo de predicción es lo más fiable posible y de que estamos resolviendo el problema en el mes correspondiente.

5.2.3. Definición del modelo

Una vez se han normalizado los valores de las variables en el mes y se han generado las muestras de entrenamiento y validación, tenemos que

definir el modelo. En este caso, distinguimos dos escenarios dependiendo de si empleamos un modelo de XGBoost o de RNA.

- Si utilizamos un modelo de XGBoost debemos establecer el número de árboles, su profundidad y el ratio de aprendizaje que tendrán todos los delay modelos de XGBoost (recordemos que con XGBoost tenemos que entrenar un modelo independiente para predecir el valor de la variable objetivo en el instante de tiempo i futuro. Sección 4.5).
- En caso de emplear un modelo de RNA, debemos decidir el tipo de arquitectura concreta a emplear (MLP, RNN, CNN o Seq2Seq. Sección 2.4.3). Tras lo cual tenemos que establecer el número de capas ocultas que tendrá la red y los parámetros de cada una de ellas, como la función de activación, número de neuronas, etc.

5.2.4. Elección del algoritmo de optimización y de la función de error

Una vez tenemos las muestras de entrenamiento y validación y hemos definido los hiperparámetros del modelo de predicción tenemos que decidir la función de error que pretendemos minimizar mediante el entrenamiento y el algoritmo de optimización (si estamos empleando una RNA) para minimizarla.

La función de error, en nuestro caso (problema de regresión) permite calcular cual es la diferencia entre los valores predichos por el modelo y los verdaderos valores y es la que el modelo va a tratar de minimizar durante el entrenamiento para aprender a predecir. En el caso de XGBoost la función de error a minimizar durante el entrenamiento es el error cuadrático medio (MSE), mientras que en el caso de las redes neuronales podemos hacer uso de múltiples funciones de error (MAE, MSE, MAPE, etc), siendo en nuestro caso la función elegida el error medio absoluto o MAE. De esta forma, los modelos de XGBoost se entrenaran minimizando el MSE mientras que las RNA lo harán empleando el MAE pero para evaluar el rendimiento tanto de los modelos de XGBoost como de RNA mediremos su error sobre los datos de validación con la función MAE, la cual tiene la siguiente fórmula para nuestro caso particular.

$$MAE(m) = \frac{\sum_{i=0}^{delay-1} |y_i - \hat{y}_i|}{delay} \tag{5.1}$$

En la ecuación 5.1 podemos ver la función que permite calcular el error que comete el modelo al predecir a partir de una muestra concreta m los siguientes delay valores de la variable objetivo. Para ello vemos que a cada valor predicho $\hat{y}_i$ le resta el valor deseado y_i y se queda con el valor absoluto. Finalmente se suman todas las diferencias entre valores predichos y verdaderos y se calcula el promedio.

Este es el MAE que comete el modelo a la hora de predecir una muestra concreta, pero podemos calcular el MAE que comete al predecir un lote de N muestras sumando el MAE de todas las muestras y haciendo el promedio tal y como se ve en la ecuación 5.2.

$$MAE = \frac{\sum_{m=0}^{N-1} MAE(m)}{N} \tag{5.2}$$

Empleamos esta función de error debido a que en los experimentos nos ha ofrecido un mejor aprendizaje en las RNA y a que es más interpretativa, pues nos indica cuanto se desvian en media las predicciones del modelo con respecto a los valores verdaderos. Dicho esto, tenemos que tener en cuenta que estamos trabajando con los datos normalizados, por lo que cuando obtengamos el MAE de un modelo sobre los conjuntos de entrenamiento y de validación, este será un MAE normalizado o NMAE. Para expresarlo en consumo de calefacción (KW) tendremos que multiplicarlo por la desviación típica de los valores de la variable objetivo de entrenamiento en el mes actual, de acuerdo con [**45**].

Por último, en el caso de los modelos de RNA debemos decidir qué algoritmo de optimización emplear, el cual, como se comentó en la sección 2.4.4.2 es el encargado de establecer como se modifican los pesos de la red a la hora de entrenar para minimizar la función de error. Como veremos más adelante se ha experimentado con tres algoritmos de optimización distintos.

5.2.5. Entrenamiento y validación

Finalmente, tras obtener las muestras de entrenamiento y validación normalizadas, definir el modelo concreto de XGBoost o RNA y elegir la función de error a minimizar y el algoritmo de optimización, tenemos que entrenar el modelo con las muestras de entrenamiento, tras lo cual evaluaremos si este es suficientemente bueno o no para llevar a cabo la tarea de predicción midiendo el NMAE que comete al predecir las muestras de validación. En este caso, cuanto menor sea este error mejor será el modelo, pues menos se desviará en media al predecir los valores de la variable objetivo y por tanto mejor generalizará. Sin embargo, no solo emplearemos el NMAE sobre el conjunto de muestras de validación para ver si el modelo es bueno o no, pues el NMAE es una media y podría llevarnos a engaño. Por tanto, para comprobar si el modelo obtenido tiene la calidad suficiente también vamos a visualizar los valores de validación de la variable objetivo en una gráfica a lo largo del tiempo junto con los valores verdaderos. Esto se hará de la siguiente forma:

1. Seleccionamos una de cada delay observaciones de validación.

2. Con cada observación seleccionada generamos una muestra de validación.

3. Predecimos con el modelo cada una de las muestras y pintamos todos los valores predichos en una gráfica en orden cronológico junto a los valores verdaderos.

Con esto lo que estamos haciendo es pedirle al modelo que nos prediga los valores de validación de la variable objetivo de delay en delay, que es para lo que está entrenado y los pintamos junto a los valores verdaderos. Así podremos ver de forma visual si la calidad de las predicciones del modelo es realmente buena.

5.3. Experimentación y discusión de resultados

Finalmente, para cada uno de los problemas (Enero, Febrero y Marzo) vamos a ver en esta última sección lo siguiente:

1. Descripción del proceso llevado a cabo para seleccionar el mejor modelo de XGBoost, MLP, RNN, CNN y Seq2Seq.
2. Ver la arquitectura (en el caso de las RNA) y los parámetros del mejor modelo de cada tipo.
3. Comparar y discutir el rendimiento de los mejores modelos de XGBoost, MLP, RNN, CNN y Seq2Seq.

5.3.1. Selección del mejor modelo de cada tipo

En primer lugar se ha llevado a cabo un proceso de experimentación con el objetivo de encontrar para los problemas de Enero, Febrero y Marzo un buen modelo de XGBoost, MLP, RNN, CNN y Seq2Seq. Para ello, se ha hecho por cada problema lo siguiente:

1. Por cada tipo de modelo (XGBoost, MLP, RNN, CNN y Seq2Seq) se han entrenado varios modelos con distintos parámetros.
2. Se ha evaluado con la función de error NMAE el rendimiento de cada modelo sobre el conjunto de muestras de validación.
3. Se comparan los distintos modelos de cada tipo en función de su valor de NMAE para así obtener el mejor.

Cabe destacar que como la inicialización de los pesos de los modelos de redes neuronales se realiza de forma aleatoria se ha ejecutado varias veces el entrenamiento de cada modelo específico quedándonos con su NMAE sobre el conjunto de validación promedio.

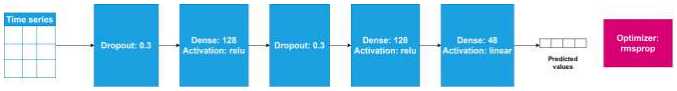
Estos pasos se han llevado a cabo tratando de predecir únicamente H-F123-D1, pues es muy parecida a H-F123-D5/2 (como vimos en el apartado

4.3.5) y tienen casi (a diferencia de una) las mismas variables predictoras, por lo que los mejores modelos para predecir H-F123-D1 es muy probable que también sean los mejores para predecir H-F123-D5/2 y de esta forma nos ahorramos la mitad de los experimentos.

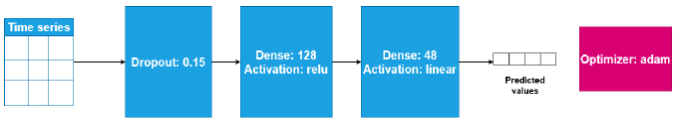
En las siguiente tabla (XGBoost) y 4 figuras (modelos de RNA) se puede ver con detalle las arquitecturas e hiperparámetros de los mejores modelos de cada tipo para cada problema. En el apéndice B.1 se puede consultar una discusión un poco más profunda acerca de los experimentos llevados a cabo con cada tipo de modelo y de sus arquitecturas.

Problema	N. Árboles	Prof. Máxima	Tasa aprendizaje
Enero	50	51	0.05
Febrero	100	100	0.001
Marzo	50	51	0.05

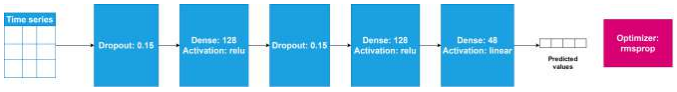
Tabla 5.1: Mejores modelos de XGBoost para enero, febrero y marzo.



(a) Enero.

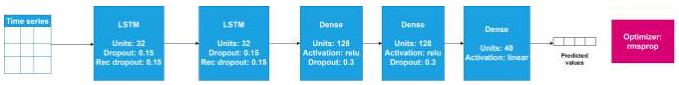


(b) Febrero.

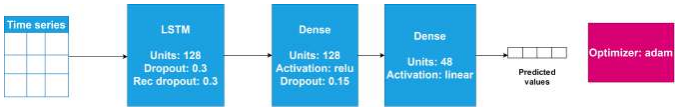


(c) Marzo.

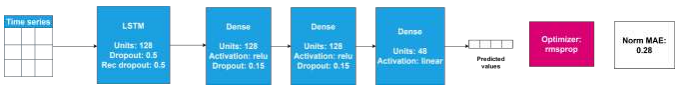
Figura 5.1: Mejores modelos de MLP.



(a) Enero.



(b) Febrero.



(c) Marzo.

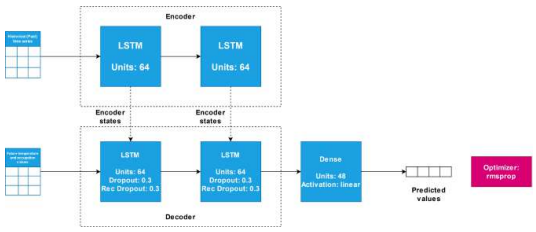
Figura 5.2: Mejores modelos de RNN.

(a) Enero.

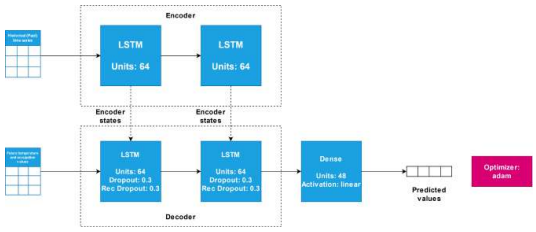
(b) Febrero.

(c) Marzo.

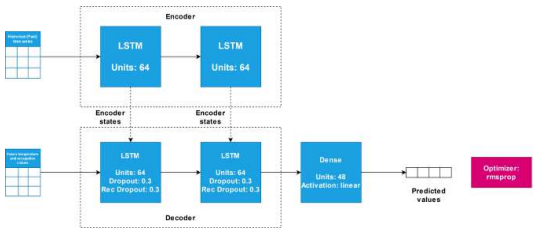
Figura 5.3: Mejores modelos de CNN.



(a) Enero.



(b) Febrero.



(c) Marzo.

Figura 5.4: Mejores modelos de Seq2Seq.

5.3.2. Elección del mejor modelo para cada problema

En la sección anterior hemos visto el proceso de experimentación general llevado a cabo para obtener el mejor modelo de XGBoost, MLP, RNN, CNN y Seq2Seq para predecir H-F123-D1 en los problemas de enero, febrero y marzo y los diagramas de dichos modelos. Pues bien, finalmente en esta sección vamos a ver con más detalle cual ha sido el rendimiento de los mejores modelos de cada tipo a la hora de predecir H-F123-D1 y H-F123-D5/2 en los problemas de Enero, Febrero y Marzo y los compararemos entre ellos para elegir el mejor para cada problema. Para ello, seguiremos el siguiente proceso por cada problema:

1. Los mejores modelos de XGBoost, MLP, RNN, CNN y Seq2Seq obtenidos en el proceso de experimentación anterior van a ser entrenados para predecir H-F123-D1 y H-F123-D5/2. Cabe destacar que todos los modelos de RNA (MLP, RNN, CNN y seq2seq) han sido entrenados

empleando los mismos parámetros de entrenamiento especificados en el apéndice B.1, salvo por el número de épocas que en este caso ha sido de 100, para darle más tiempo a algunos modelos a que convergan.

2. A continuación recogeremos en una tabla el NMAE sobre el conjunto de validación que obtiene el mejor modelo de cada tipo a la hora de predecir H-F123-D1 y H-F123-D5/2 y compararemos los rendimientos de los distintos modelos.

3. Finalmente para cada variable objetivo (H-F123-D1 y H-F123-D5/2) visualizaremos en una gráfica sus valores de validación verdaderos junto a los predichos por el modelo que obtiene un mejor rendimiento sobre ella.

5.3.2.1. Problema de enero

En primer lugar analizaremos el rendimiento del mejor modelo de XGBoost, MLP, RNN, CNN y Seq2Seq a la hora de predecir H-F123-D1 y H-F123-D5/2 en enero. En el apéndice B.2.1 podemos ver un análisis aun más detallado de cada modelo para abordar el problema de enero.

Modelo	NMAE	
	H-F123-D1	H-F123-D5/2
XGBoost	0.48	0.56
MLP	0.38	0.31
RNN	0.27	0.30
CNN	0.34	0.36
Seq2Seq	**0.21**	**0.20**

Tabla 5.2: Rendimiento de cada modelo para predecir H-F123-D1 y H-F123-D5/2 en enero.

En la tabla 5.2 podemos ver en cada fila el NMAE de validación que ha obtenido un modelo concreto a la hora de predecir H-F123-D1 y H-F123-D5/2 en Enero. De esta tabla podemos destacar lo siguiente:

- El modelo XGBoost (sección 5.3.1, tabla 5.1 (fila 1)) es el que obtiene unos valores más elevados de NMAE de validación tanto para H-F123-D1 como para H-F123-D5/2, por lo que es el que peor rendimiento ofrece. Esto se debe a que el modelo XGBoost está en realidad formado por 48 modelos cada uno entrenado de forma independiente para predecir el valor de la variable objetivo en un timestep concreto a partir de una serie temporal de entrada con 384 observaciones, por lo que no es capaz de establecer relaciones de dependencia entre los

valores predichos. Además, cada uno de los 48 modelos procesan la serie temporal de entrada de una sola vez, por lo que no tienen en cuenta las dependencias temporales entre los distintas observaciones de la misma.

- El modelo MLP (sección 5.3.1, figura 5.1 (a)) reduce bastante el NMAE de validación en comparación con XGBoost, lo cual se debe a que el modelo MLP sí es capaz de tener en cuenta dependencias entre los datos de salida, pues sus capas están totalmente conectadas.

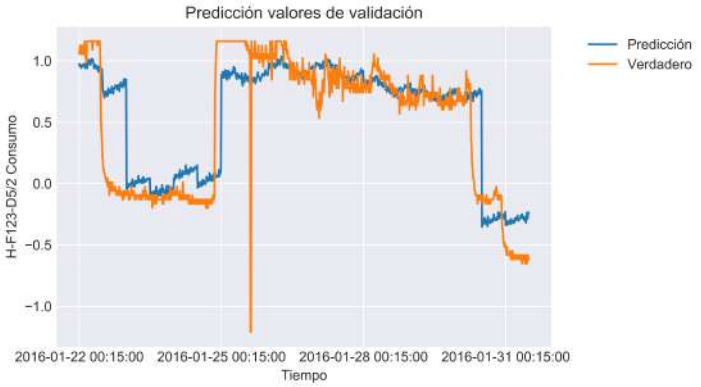
- El modelo RNN (sección 5.3.1, figura 5.2 (a)) mejora un poco el rendimiento del modelo MLP, pues a diferencia de este último procesa cada serie temporal de entrada teniendo en cuenta el orden cronológico de sus observaciones y las dependencias temporales entre las mismos.

- El modelo CNN (sección 5.3.1, figura 5.3 (a)) obtiene unos valores de NMAE de validación un poco mayores que los obtenidos con el modelo RNN, por lo que parece que las características temporales extraídas por las capas convolucionales 1D no ayudan tanto a la predicción en este problema como las características que son capaces de extraer las capas LSTM.

- El modelo Seq2Seq (sección 5.3.1, figura 5.4 (a)) es el que obtiene con diferencia unos valores de NMAE más bajos al predecir los valores de validación de H-F123-D1 y H-F123-D5/2, por ello vemos que sus resultados están resaltados en negrita. Por tanto, para el problema de enero los datos futuros de temperatura y ocupación que emplea el modelo Seq2Seq ayudan a la predicción.

Puesto que el modelo Seq2Seq es el que mejor rendimiento obtiene a la hora de predecir H-F123-D1 y H-F123-D5/2 vamos a ver gráficamente cómo predice los valores de validación de ambas variables. En el apéndice B.2.1 se puede consultar cómo predicen el resto de modelos.

(a) H-F123-D1



(b) H-F123-D5/2.

Figura 5.5: Predicción de los valores de validación en enero mediante Seq2Seq.

En la figura 5.5 vemos dos gráficas con los valores de validación verdaderos (naranja) junto a los predichos (azul) por el mejor modelo de Seq2Seq del problema de Enero (eje y) a lo largo del tiempo (eje x) para H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b). Estos valores de validación van desde el instante de tiempo 2016-01-22 00:15:00 hasta el 2016-01-31 23:45:00 que como dijimos en la sección 5.1, son los instantes de tiempo de validación del problema del mes de enero. Vemos que predice bastante bien las subidas y bajadas en el consumo de calefacción de H-F123-D1 y H-F123-D5/2, tal y como imaginábamos de acuerdo sus bajos valores de NMAE. Aunque eso sí, vemos un desfase pequeño entre los valores predichos y los verdaderos, pareciendo predecir las bajadas y las subidas de consumo un poco más tarde. Esto se puede deber principalmente a la diferencia en los instantes de tiempo de entrenamiento y de validación, pues las subidas y bajadas de consumo de calefacción en los primeros 21 días de Enero (instante de tiempo de entrenamiento) tienen lugar en momentos ligeramente distintos en comparación con las subidas y bajadas en los últimos 10 días (instantes de tiempo de

validación).

5.3.2.2. Problema de febrero

Ahora analizaremos el rendimiento del mejor modelo de XGBoost, MLP, RNN, CNN y Seq2Seq a la hora de predecir H-F123-D1 y H-F123-D5/2 en febrero. En el apéndice B.2.2 podemos ver un análisis aun más detallado de los mejores modelos del problema de Febrero.

Modelo	NMAE	
	H-F123-D1	H-F123-D5/2
XGBoost	1.17	1.10
MLP	0.82	0.61
RNN	0.43	0.31
CNN	**0.43**	**0.30**
Seq2Seq	0.43	0.36

Tabla 5.3: Rendimiento de cada modelo para predecir H-F123-D1 y H-F123-D5/2 en febrero.

En la tabla 5.3 podemos ver en cada fila el NMAE de validación que ha obtenido un modelo concreto a la hora de predecir H-F123-D1 y H-F123-D5/2 en Febrero. De esta tabla podemos destacar lo siguiente:

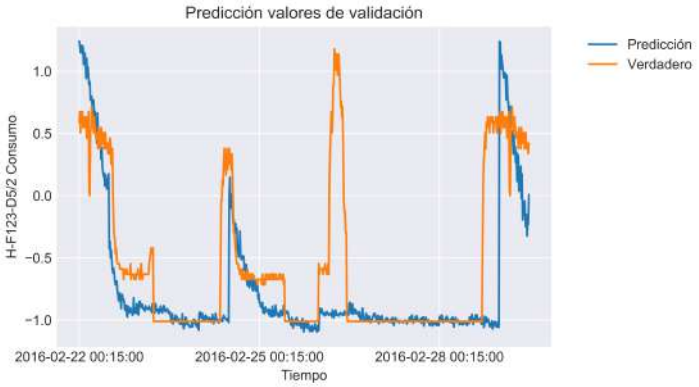
- En general los valores de NMAE de cada modelo a la hora de predecir los valores de validación de H-F123-D1 y H-F123-D5/2 son bastante mayores que los obtenidos en el problema de febrero, lo que nos indica que este problema es más difícil de abordar.
- El modelo XGBoost (sección 5.3.1, tabla 5.1, (fila 2)) es, al igual que en el problema de Enero, el modelo que mayor NMAE de validación obtiene al predecir H-F123-D1 y H-F123-D5/2. Esto confirma aún más las carencias de este modelo para el procesamiento de series temporales expuestas en el problema de enero.
- El modelo MLP (sección 5.3.1, figura 5.1 (b)), con su capacidad para establecer dependencias entre los valores de consumo de calefacción predichos vuelve a mejorar el rendimiento ofrecido por el modelo XGBoost, al igual que en el problema de enero.
- El modelo RNN (sección 5.3.1, figura 5.2 (b)) reduce aproximadamente hasta la mitad el NMAE de validación sobre H-F123-D1 y H-F123-D5/2, confirmando aun más la capacidad de estos modelos para extraer características temporales que facilitan enormemente la predicción de series temporales.

- El modelo CNN (sección 5.3.1, figura 5.3 (b)) obtiene el mismo NMAE de validación que el modelo RNN al predecir H-F123-D1 y ligeramente inferior al predecir H-F123-D5/2, por este motivo sus resultados se han resaltado en negrita indicando que este es el mejor modelo de febrero. Por tanto, el hecho de añadir capas convolucionales antes de las capas recurrentes LSTM viene un poco mejor en este problema de marzo.

- El modelo Seq2Seq (sección 5.3.1, figura 5.4 (b)) obtiene el mismo NMAE de validación que los modelos RNN y CNN a la hora de predecir H-F123-D1 y obtiene un NMAE de validación mayor al predecir H-F123-D5/2, por lo que a diferencia de lo que ocurría en enero. Por tanto, parece que la información contextual o futura sobre temperatura y ocupación no es de demasiada ayuda para el problema de febrero.

A continuación vamos a ver gráficamente como el mejor modelo del problema de febrero, es decir, el modelo CNN, predice los valores de validación de H-F123-D1 y H-F123-D5/2.

(a) H-F123-D1



(b) H-F123-D5/2.

Figura 5.6: Predicción de los valores de validación en Febrero mediante CNN.

En la figura 5.6 podemos ver como para predecir los valores de validación de H-F123-D1 (subfigura a) y H-F123-D52 (subfigura b) en el mes de eebrero (desde el instante de tiempo 2016-02-22 00:15:00 hasta el 2016-02-29 23:45:00) el modelo CNN es capaz de aproximar todas las subidas y bajadas de sus valores verdaderos. Sin embargo, vemos que en torno al día 26 de Febrero hay en ambas variables una subida y bajada que el modelo no es capaz de predecir correctamente, la cual si recordamos es originada por nuestro método de imputación (sección 4.3.4) con el que imputamos todos los valores perdidos de los días 26, 27, 28 y 29 de febrero. Por lo que parece ser que los valores imputados no se corresponden con los valores verdaderos, siendo la subida y bajada de consumo del día 26 anómala o inesperada e imposible de predecir por el modelo. Esto vemos que contribuye bastante a aumentar el NMAE de validación para H-F123-D1 y H-F123-D5/2 y puede ser una de las causas de la mayor dificultad de este problema.

5.3.2.3. Problema de marzo

Finalmente vamos a analizar el rendimiento del mejor modelo de XGBoost, MLP, RNN, CNN y Seq2Seq prediciendo H-F123-D1 y H-F123-D5/2 en marzo. En el apéndice B.2.3 puede verse un análisis aun más detallado de cada uno de los mejores modelos de marzo.

Modelo	NMAE	
	H-F123-D1	H-F123-D5/2
XGBoost	0.56	0.60
MLP	0.53	0.64
RNN	**0.29**	**0.29**
CNN	0.34	0.33
Seq2Seq	0.39	0.37

Tabla 5.4: Rendimiento de cada modelo para predecir H-F123-D1 y H-F123-D5/2 en marzo.

En la tabla 5.4 podemos ver en cada fila el NMAE de validación que ha obtenido un modelo concreto a la hora de predecir H-F123-D1 y H-F123-D5/2 en Febrero. De esta tabla podemos destacar lo siguiente:

- El modelo XGBoost (sección 5.3.1, tabla 5.1 (fila 3)) obtiene, al igual que en los problemas de enero y febrero valores de NMAE de validación bastante elevados a la hora de predecir H-F123-D1 y H-F123-D5/2, confirmando de manera definitiva el pobre rendimiento que ofrecen estos modelos a la hora de predecir series temporales en comparación con las técnicas de aprendizaje profundo.

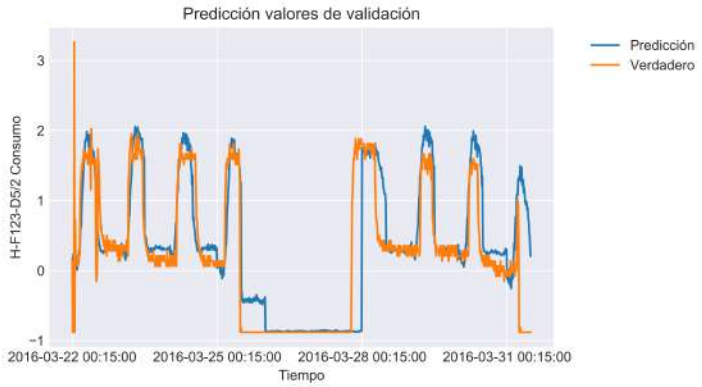
- El modelo MLP (sección 5.3.1, figura 5.1 (c)) obtiene un rendimiento similar al modelo XGBoost para predecir H-F123-D1 y H-F123-D5/2, siendo incluso peor a la hora de predecir H-F123-D5/2. Esto, nos muestra las carencias de las RNA tradicionales a la hora de manejar datos de dependientes del tiempo.

- El modelo RNN (sección 5.3.1, figura 5.2 (c)) consigue reducir en aproximadamente la mitad los NMAE de valdidación conseguidos con los modelos XGBoost y MLP, siendo los valores más bajos conseguidos en este problema de marzo, de ahí que esten resaltados en negrita. Por tanto, al igual que en los problemas de enero y febrero volvemos a ver el buen desempeño de las capas recurrentes LSTM a la hora de tratar con series temporales.

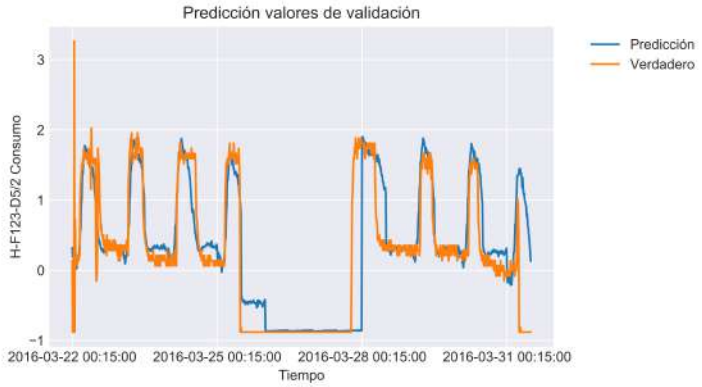
- El modelo CNN (sección 5.3.1, figura 5.3 (c)) obtiene unos valores de NMAE de validación mejores que los de XGBoost y MLP pero ligeramente superiores a los del modelo RNN, por lo que para este problema de marzo no se mejora el rendimiento añadiendo capas convolucionales 1D antes de las capas recurrentes LSTM.

- El modelo Seq2Seq (sección 5.3.1, figura 5.4 (c)) obtiene también un mejor rendimiento que los modelos XGBoost y MLP pero tiene un rendimiento ligeramente peor que el modelo CNN. Por tanto, al igual que ocurría en el problema de febrero la información futura acerca de la temperatura y ocupación del edificio ICPE no ayudan demasiado a la predicción del consumo de calefacción.

Finalmente veamos como el mejor modelo del problema de marzo, es decir, el modelo RNN, predice los valores de validación de H-F123-D1 y H-F123-D5/2.



(a) H-F123-D1



(b) H-F123-D5/2.

Figura 5.7: Predicción de los valores de validación en Marzo mediante RNN.

En la figura 5.7 podemos ver como al predecir los valores de validación de H-F123-D1 (subfigura a) y H-F123-D52 (subfigura b) en el mes de marzo (desde el instante de tiempo 2016-03-22 00:15:00 hasta el 2016-03-31 23:45:00) el modelo RNN ha conseguido predicciones más parecidas a los valores verdaderos que los conseguidos mediante los mejores modelos correspondientes en los problemas de enero y febrero, lo cual se debe principalmente a que en Marzo los datos son más estacionales y por tanto fáciles de predecir, pues se repiten una y otra vez las mismas oscilaciones semanales en el consumo.

5.3.3. Resumen

Después de ver con detalle para cada problema el rendimiento que proporciona cada tipo de modelo a la hora de predecir las variables H-F123-D1 y H-F123-D5/2, se recogen todos estos rendimientos en las siguientes dos tablas a modo de resumen.

Modelo	*Enero*		*Febrero*		*Marzo*	
	NMAE	*MAE (KW)*	*NMAE*	*MAE (KW)*	*NMAE*	*MAE (KW)*
XGBoost	0.48	2.5	1.17	5.88	0.56	2.24
MLP	0.38	1.97	0.82	4.12	0.53	2.12
RNN	0.27	1.40	0.43	2.16	**0.29**	**1.16**
CNN	0.34	1.77	**0.43**	**2.16**	0.34	1.36
Seq2Seq	**0.21**	**1.09**	0.43	2.16	0.39	1.56

Tabla 5.5: Rendimiento de cada modelo para predecir H-F123-D1 en cada problema.

Modelo	*Enero*		*Febrero*		*Marzo*	
	NMAE	*MAE (KW)*	*NMAE*	*MAE (KW)*	*NMAE*	*MAE (KW)*
XGBoost	0.56	16.05	1.10	26.125	0.60	8.25
MLP	0.31	8.88	0.61	14.48	0.64	8.8
RNN	0.30	8.6	0.31	7.36	**0.29**	**3.98**
CNN	0.36	10.32	**0.30**	**7.12**	0.33	4.53
Seq2Seq	**0.20**	**5.73**	0.36	8.55	0.37	5.08

Tabla 5.6: Rendimiento de cada modelo para predecir H-F123-D5/2 en cada problema.

En las tablas 5.5 y 5.6 vemos el rendimiento de cada modelo a la hora de predecir los valores de validación de H-F123-D1 y H-F123-D5/2 respectivamente para cada problema. Más en concreto en cada fila de ambas tablas tenemos, por cada modelo, tres columnas, en cada una de las cuales encontramos el NMAE de validación que se obtiene con dicho modelo en un problema concreto junto a dicho NMAE convertido en KW (multiplicando el NMAE por la desviación típica de los valores de los timesteps de entrenamiento de la variable objetivo en el mes o problema correspondiente), que es la unidad original de las variables H-F123-D1 y H-F123-D5/2. En ambas tablas vemos lo siguiente.

- Los valores de NMAE de cada modelo a la hora de predecir H-F123-D1 y H-F123-D5/2 en cada problema son similares, pero los valores de MAE de H-F123-D5/2 son mayores, por lo que en media al predecir esta variable con cualquier modelo nos desviamos una mayor cantidad de KW. Esto se debe pricipalmente a que los valores de la varaible H-F123-D5/2 se mueven en un rango de 0 a 70 KW, mientras que la variable H-F123-D1 lo hace en un rango de 0 a 12 KW, por lo que la variable H-F123-D5/2 presenta una mayor desviación típica en sus valores, haciendo que ante un NMAE similar para predecir H-F123-D1 y H-F123-D5/2, obtengamos para esta última un MAE mayor.

- Por cada columna o problema vemos unos valores de NMAE y MAE resaltados en negrita, los cuales son los mínimos alcanzados para ese

problema. Así, para para ambas variables objetivo se tiene que el modelo Seq2Seq es el mejor para el problema de enero, mientras que los modelos CNN y RNN los son para los problemas de febrero y marzo respectivamente.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se exponen las conclusiones finales en relación con el principal objetivo planteado en este proyecto, extraídas tras la realización de las tareas de adquisición de datos, preprocesamiento y experimentación con distintos modelos de predicción. Tras lo cual se comentan las posibles vías de investigación y de trabajo futuro que se pueden seguir para tratar de profundizar y mejorar en distintos aspectos de este proyecto.

6.1. Conclusiones

En este proyecto se ha planteado el objetivo de predecir el consumo energía eléctrica destinada al funcionamiento de los sistemas calefacción en el futuro en el edificio ICPE haciendo uso de modelos de aprendizaje automático y en especial de aprendizaje profundo. Para alcanzar este objetivo se han llevado a cabo las siguientes tareas:

1. Descripción del edificio ICPE para obtener un mayor conocimiento del mismo y de sus sensores.

2. Obtención en forma de serie temporal multivariable de los valores de los sensores con los que resolver el problema.

3. Análisis exploratorio de los sensores para obtener un mayor conocimiento acerca de los mismos.

4. Preprocesamiento de los datos con el objetivo de poder aplicarles técnicas de aprendizaje automático.

Estas cuatro tareas nos han permitido definir el problema de predicción concreto a resolver, el cual ha consistido en predecir por separado los valores de las variables de consumo de calefacción H-F123-D1 y H-F123-D5/2 a partir de sus propios valores pasados y de los de 6 variables predictoras en los meses de enero, febrero y marzo. Para resolver cada uno de estos problemas de predicción se ha experimentado con 5 técnicas o modelos de aprendizaje automático: XGBoost, MLP, RNN, CNN y Seq2Seq, de las cuales las 4 últimas son de aprendizaje profundo. De esta forma se han obtenido las siguientes conclusiones como consecuencia de esta experimentación:

- El primer modelo con el que hemos abordado los problemas ha sido XGBoost, un técnica de aprendizaje automático tradicional, la cual se pretendía que nos sirviese como caso base o punto de partida cuyo rendimiento se pretendía mejorar con los modelos de aprendizaje profundo. Cosa que hemos conseguido, pues este modelo es el que peor rendimiento ha obtenido en todos los problemas en términos de NMAE de validación. Este peor rendimiento se debe a que el modelo de XGBoost no es capaz de extraer dependencias temporales de la serie temporal que recibe como entrada ni de establecer dependencias entre los valores de consumo de calefacción predichos. Lo cual se debe a que para poder emplear la técnica XGBoost para llevar a cabo tareas de predicción multistep de series temporales multivariable hemos tenido que entrenar varios modelos de XGBoost independientes, cada uno encargado de de predecir el i-ésimo valor de consumo de calefacción en el futuro. De esta forma, ha quedado patente la superioridad que ofrecen, al menos en estos problemas de predicción de series temporales, las técnicas de aprendizaje profundo frente a otras técnicas de aprendizaje automático más tradicionales y bastante contrastadas como XGBoost.

- El segundo modelo con el que hemos probado a resolver todos los problemas ha sido el modelo MLP, que es una RNA clásica con varias capas ocultas totalmente conectadas. Este modelo, debido a que es capaz de proporcionar como salida varios valores, es decir, de predecir varios valores de consumo de calefacción, sí puede establecer relaciones entre los valores de consumo de calefacción predichos. De ahí que obtenga un mejor rendimiento que el modelo XGBoost. Sin embargo, no es mejor que los siguientes modelos que hemos probado, pues al igual que XGBoost este modelo sigue procesando cada serie temporal de una sola vez sin tener en cuenta el orden cronológico de las observaciones ni las dependencias temporales entre las mismas, algo muy importante a la hora de predecir series temporales.

- Tras probar el modelo MLP, se tomaron capas recurrentes LSTM y se añadieron como primeras capas ocultas a los mejores modelos MLP de

cada problema dando así lugar a los modelos RNN. Esto ha reducido el NMAE de validación cometido por los modelos MLP en algunos problemas hasta la mitad. Esto se debe a que las primeras capas LSTM procesan la serie temporal de entrada observación a observación de forma recurrente teniendo en cuenta o recordando la la información obtenida al procesar las anteriores observaciones, le permite tener en cuenta el orden cronológico de las observaciones y extraer dependencias temporales entre las mismas que facilitan la predicción a las últimas capas totalmente conectadas de la red. Estos modelos RNN son los que mejor rendimiento han ofrecido en global para predecir H-F123-D1 y H-F123-D5/2 en enero, febrero y marzo.

- Los modelos CNN han surgido al añadir capas convolucionales antes de las capas LSTM de los mejores modelos RNN. Así, se esperaba que las capas convolucionales 1D extrajesen de forma automática características temporales de la serie temporal de entrada, lo cual no ha servido de mucho ya que hemos conseguido un rendimiento próximo al de los modelos RNN pero ligeramente inferior en global. Aunque eso sí, es el modelo que mejor rendimiento ha obtenido en el mes de Febrero a la hora de predecir tanto H-F123-D1 como H-F123-D5/2.

- Los modelos Seq2Seq son un tipo de RNA con una arquitectura un poco más compleja que el resto, los cuales han utilizado una información adicional a la hora de predecir el consumo de calefacción. Esta información consiste en valores de temperatura y de ocupación que habrán en los instantes de tiempo futuros en los que se va a predecir el consumo de calefacción. Con esta información extra y con el uso de capas LSTM se esperaba que los modelos Seq2Seq mejorasen el rendimiento del resto de modelos. Sin embargo, estos solo ha ocurrido en enero a la hora de predecir H-F123-D1 y H-F123-D5/2, mientras que en febrero y marzo su rendimiento es ligeramente peor que el de los modelos RNN y CNN. Esto puede deberse a que en enero los valores futuros de ocupación y temperatura están más correlados con los valores a predecir de H-F123-D1 y H-F123-D5/2.

Dicho esto, a la hora de entrenar los distintos tipos de modelos para aprender a predecir H-F123-D1 y H-F123-D5/2 en enero, febrero y marzo hemos encontrado los siguientes inconvenientes:

- En cada mes disponemos de muy pocos datos para entrenar y validar. Pues entrenamos solo con las observaciones de los primeros 21 días del mes y validamos con las observaciones del resto de días. Esto provoca que haya falta de variedad en los datos y que por tanto el modelo no aprenda a predecir el consumo de calefacción en un amplio abanico de circunstancias.

- Los valores de entrenamiento y validación de todas las variables en cada mes difieren un poco debido al hecho de entrenar con los primeros 21 días y validar con el resto. Esto hace que hayan pocos patrones útiles que cada modelo pueda aprender del conjunto de entrenamiento para predecir el conjunto de validación.

Estos dos inconvenientes, por un lado son los principales causantes de que los modelos de RNA, por muy simples que sean, casi siempre sobreajusten el conjunto de entrenamiento sin llegar a converger tanto como nos gustaría. Por otro, hacen que tal como hemos observado en la evaluación de los distintos tipos de modelos para cada problema, estos modelos predigan los valores de validación con cierto desfase con respecto a los valores verdaderos.

Sin embargo, a pesar de todos estos problemas y sus consecuencias, los modelos que emplean capas recurrentes LSTM, es decir, RNN, CNN y Seq2Seq nos aportan predicciones de los valores de validación de H-F123-D1 y H-F123-D5/2 bastante cercanas visualmente a los valores verdaderos de validación y con un error medio absoluto o MAE de solo 1 o 5 KW.

Por tanto, en este proyecto hemos podido comprobar que las técnicas de aprendizaje profundo especializadas en el tratamiento de series temporales nos ofrecen modelos con una gran capacidad de aprendizaje y de generalización a la hora de resolver un problema real para el cual disponemos de escasos datos y que tiene una gran cantidad de inconvenientes.

6.2. Trabajo futuro

Finalmente vamos a comentar algunas de la posibles vías de trabajo futuro que merecería la pena seguir para tratar de mejorar los resultados obtenidos y que no han podido llevarse a cabo debido a la delimitación del objetivo, a que se han intentado pero de forma superficial y sin ofrecer resultados significativos o a que no se han podido llevar a cabo.

- Uno de los problemas que hemos tenido en el proyecto ha sido la falta de datos, pues solo hemos trabajado con los valores de los sensores del edificio ICPE de los meses de enero, febrero y marzo del 2016. Para solucionarlo existen varias vías entre las que destacan las tres siguientes:

    - Tomar datos de los sensores del edificio ICPE de otros años y entrenar con ellos. De esta forma podríamos utilizar los datos de enero, febrero y marzo de una serie de años y entrenar con todos ellos dejando para validar los datos de enero, febrero y marzo de un año. Así conseguimos una mayor variedad en los datos de entrenamiento y solucionamos el problema de que no se

parecen los datos de entrenamiento y validación, pues los valores de los sensores de un mismo mes en distintos años son parecidos (estacionales).

  - Si no se pueden obtener los datos de los sensores del edificio ICPE de otros años podemos generarlos de manera artificial mediante algún tipo de modelo de simulación fiable del edificio, el cual nos va a permitir generar bajo diferentes circunstancias reales simuladas valores aproximados de todas las variables.
  - No solo está la posibilidad de trabajar con más datos, sino que también se podría averiguar qué miden algunos de los sensores que no hemos empleado en este proyecto y utilizar sus valores para predecir el consumo de calefacción.

- Se podría investigar la utilización de técnicas de eliminación del ruido de los valores de las variables, para de esta forma suavizar sus oscilaciones bruscas a lo largo del tiempo para ver si mejora el rendimiento de los sensores. Esto es algo que se ha intentado en este proyecto pero solo empleando un filtro de Savitzky-Golay **[75]** y sin obtener una mejora en el rendimiento de los modelos.

- Una solución para intentar evitar el problema de que no hay parecido entre los datos de entrenamiento y validación es entrenar con datos distribuidos en vez de continuos en el tiempo. Esto consiste en seleccionar las observaciones de entrenamiento y validación de un problema de forma aleatoria. Lo hemos intentado probar en este proyecto pero presenta el inconveniente de que cuando generamos a partir de una observación un muestra de validación, todos las observaciones que pertenezcan a ella ya no podrán ser utilizadas para entrenar, para así evitar utilizar las observaciones de entrenamiento para validación, lo que provoca que cuando no tengamos muchos datos nos quedemos rápido sin observaciones si tenemos un lookback elevado como es nuestro caso.

- Por último, también se puede intentar emplear modelos de redes neuronales recurrentes que empleen un mecanismo de atención (attention) **[76]** que es una técnica del estado del arte del procesamiento del lenguaje natural o NLP.

Bibliografía

[1] Daniel L. Marino, Kasun Amarasinghe y Milos Manic. «Building energy load forecasting using Deep Neural Networks». En: (oct. de 2016), págs. 7046-7051. DOI: 10.1109/IECON.2016.7793413. arXiv: 1610.09460. URL: http://ieeexplore.ieee.org/document/7793413/.

[2] Mischa Schmidt y Christer Åhlund. «Smart buildings as Cyber-Physical Systems: Data-driven predictive control strategies for energy efficiency». En: *Renewable and Sustainable Energy Reviews* 90.March 2017 (2018), págs. 742-756. ISSN: 18790690. DOI: 10.1016/j.rser.2018.04.013. arXiv: 1807.06084. URL: https://doi.org/10.1016/j.rser.2018.04.013.

[3] C. K. Chau, T. M. Leung y W. Y. Ng. «A review on life cycle assessment, life cycle energy assessment and life cycle carbon emissions assessment on buildings». En: *Applied Energy* 143.1 (2015), págs. 395-413. ISSN: 03062619. DOI: 10.1016/j.apenergy.2015.01.023.

[4] Miriam Benedetti y col. «A proposal for energy services' classification including a product service systems perspective». En: *Procedia CIRP* 30 (2015), págs. 251-256. ISSN: 22128271. DOI: 10.1016/j.procir.2015.02.121. URL: http://dx.doi.org/10.1016/j.procir.2015.02.121.

[5] Katharina Bunse y col. «Integrating energy efficiency performance in production management - Gap analysis between industrial needs and scientific literature». En: *Journal of Cleaner Production* 19.6-7 (2011), págs. 667-679. ISSN: 09596526. DOI: 10.1016/j.jclepro.2010.11.011. URL: http://dx.doi.org/10.1016/j.jclepro.2010.11.011.

[6] Andrea Conserva y col. «Energy IN TIME project: Summary of final results». En: *Proceedings of the 12th Conference on Sustainable Development of Energy, Water and Environment Systems* (2017), págs. 1-16. URL: https://decsai.ugr.es/~jgomez/bigfuse/publications.html.

[7] Michael Page Consultoría. *Tendencias del mercado laboral*. URL: https://www.michaelpage.es/sites/michaelpage.es/files/PG_ER_IT.pdf. (accessed: 03.07.2020).

[8] Periódico ElPaís. Sección Economía. *Herramientas financieras: Calculadora sueldo neto*. URL: https://cincodias.elpais.com/herramientas/calculadora-sueldo-neto/#tabla_resultados. (accessed: 03.07.2020).

[9] George Hyndman Rob J; Athanasopoulos. *Forecasting: principles and practice. 8.9 Seasonal ARIMA models*. URL: https://otexts.com/fpp2/seasonal-arima.html. (accessed: 03.07.2020).

[10] Corinna Cortes y Vladimir Vapnik. «Support-vector networks». En: *Machine Learning* 20.3 (sep. de 1995), págs. 273-297. ISSN: 0885-6125. DOI: 10.1007/BF00994018. URL: http://link.springer.com/10.1007/BF00994018.

[11] Tianqi Chen y Carlos Guestrin. «XGBoost». En: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, ago. de 2016, págs. 785-794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: https://dl.acm.org/doi/10.1145/2939672.2939785.

[12] Leo Breiman y Adele Cutler. *RandomForest*. URL: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. (accessed: 03.07.2020).

[13] Volkan Ş Ediger y Sertaç Akar. «ARIMA forecasting of primary energy demand by fuel in Turkey». En: *Energy Policy* 35.3 (2007), págs. 1701-1708. ISSN: 03014215. DOI: 10.1016/j.enpol.2006.05.009.

[14] R. E. Abdel-Aal y A. Z. Al-Garni. «Forecasting monthly electric energy consumption in eastern Saudi Arabia using univariate time-series analysis». En: *Energy* 22.11 (1997), págs. 1059-1069. ISSN: 03605442. DOI: 10.1016/S0360-5442(97)00032-7.

[15] M Y Cho, J C Hwang y C S Chen. «Customer short term load forecasting by using ARIMA transfer function model». En: *Proceedings 1995 International Conference on Energy Management and Power Delivery EMPD '95*. Vol. 1. Nov. de 1995, 317-322 vol.1. DOI: 10.1109/EMPD.1995.500746.

[16] Che Chiang Hsu y Chia Yon Chen. «Regional load forecasting in Taiwan - Applications of artificial neural networks». En: *Energy Conversion and Management* 44.12 (2003), págs. 1941-1949. ISSN: 01968904. DOI: 10.1016/S0196-8904(02)00225-X.

[17] Zaid Mohamed y Pat Bodger. «Forecasting electricity consumption in New Zealand using economic and demographic variables». En: *Energy* 30.10 (2005), págs. 1833-1843. ISSN: 03605442. DOI: 10.1016/j.energy.2004.08.012.

[18] Wei Chiang Hong. «Electric load forecasting by support vector model». En: *Applied Mathematical Modelling* 33.5 (2009), págs. 2444-2454. ISSN: 0307904X. DOI: 10.1016/j.apm.2008.07.010. URL: http://dx.doi.org/10.1016/j.apm.2008.07.010.

[19] L. Ekonomou. «Greek long-term energy consumption prediction using artificial neural networks». En: *Energy* 35.2 (2010), págs. 512-517. ISSN: 03605442. DOI: 10.1016/j.energy.2009.10.018. URL: http://dx.doi.org/10.1016/j.energy.2009.10.018.

[20] Wan He. «Load Forecasting via Deep Neural Networks». En: *Procedia Computer Science* 122 (2017), págs. 308-314. ISSN: 18770509. DOI: 10.1016/j.procs.2017.11.374. URL: https://doi.org/10.1016/j.procs.2017.11.374.

[21] J. F. Kreider y col. «Building Energy Use Prediction and System Identification Using Recurrent Neural Networks». En: *Journal of Solar Energy Engineering* 117.3 (ago. de 1995), págs. 161-166. ISSN: 0199-6231. DOI: 10.1115/1.2847757. URL: https://asmedigitalcollection.asme.org/solarenergyengineering/article/117/3/161/440937/Building-Energy-Use-Prediction-and-System.

[22] OTexts. *6.1 Time series components*. URL: https://otexts.com/fpp2/components.html. (accessed: 03.07.2020).

[23] Janine Vierheller. «Exploratory data analysis». En: *Communications in Computer and Information Science* 500 (2014), págs. 110-126. ISSN: 18650929. DOI: 10.1007/978-3-662-45006-2_9.

[24] Manish Gupta y col. «Gupta - Outlier Detection for Temporal Data - 2014». En: *IEEE Transaction on Knowledge and Data Engineering* 26.9 (2014), págs. 2250-2267.

[25] David J. Hill y Barbara S. Minsker. «Anomaly detection in streaming environmental sensor data: A data-driven modeling approach». En: *Environmental Modelling and Software* 25.9 (2010), págs. 1014-1022. ISSN: 13648152. DOI: 10.1016/j.envsoft.2009.08.010. URL: http://dx.doi.org/10.1016/j.envsoft.2009.08.010.

[26] Junshui Ma y Simon Perkins. «Online novelty detection on temporal sequences». En: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*. New York, New York, USA: ACM Press, 2003, pág. 613. ISBN: 1581137370. DOI: 10.1145/956750.956828. URL: http://portal.acm.org/citation.cfm?doid=956750.956828.

[27] R. S. Tsay. «Outliers in multivariate time series». En: *Biometrika* 87.4 (dic. de 2000), págs. 789-804. ISSN: 0006-3444. DOI: 10.1093/biomet/87.4.789. URL: https://academic.oup.com/biomet/article-lookup/doi/10.1093/biomet/87.4.789.

[28] Dae-Ki Kang, Doug Fuller y Vasant Honavar. «Learning Classifiers for Misuse Detection Using a Bag of System Calls Representation». En: 2005, págs. 511-516. DOI: 10.1007/11427995_51. URL: http://link.springer.com/10.1007/11427995%7B%5C_%7D51.

[29] Giovanni Silvestri y col. «The traditional way of». En: January (1994). DOI: 10.1109/ICNN.1994.374815.

[30] Andrew W. Williams, Soila M. Pertet y Priya Narasimhan. «Tiresias: Black-box failure prediction in distributed systems». En: *Proceedings - 21st International Parallel and Distributed Processing Symposium, IPDPS 2007; Abstracts and CD-ROM* (2007). DOI: 10.1109/IPDPS.2007.370345.

[31] Chung Chen y Lon-Mu Liu. «Joint Estimation of Model Parameters and Outlier Effects in Time Series». En: *Journal of the American Statistical Association* 88.421 (1993), pág. 284. ISSN: 01621459. DOI: 10.2307/2290724.

[32] Steffen Moritz y col. «Comparison of different Methods for Univariate Time Series Imputation in R». En: (2015). arXiv: 1510.03924. URL: http://arxiv.org/abs/1510.03924.

[33] J. Durbin y S. Koopman. *Time Series Analysis by State Space Methods.* Oxford University Press, 2012. ISBN: 9780199641178.

[34] Rob J. Hyndman y col. *forecast: Forecasting functions for time series and linear models.* English. Abr. de 2018.

[35] Matthias Templ, Alexander Kowarik y Peter Filzmoser. «Iterative stepwise regression imputation using standard and robust methods». En: *Computational Statistics and Data Analysis* 55.10 (2011), págs. 2793-2806. ISSN: 01679473. DOI: 10.1016/j.csda.2011.04.012. URL: http://dx.doi.org/10.1016/j.csda.2011.04.012.

[36] S. Van Buuren. *Flexible Imputation of Missing Data. Second Edition.* Chapman y Hall/CRC, 2018. ISBN: 9781138588318.

[37] W.L. Junger y A. Ponce de Leon. «Imputation of missing data in time series for air pollutants». En: *Atmospheric Environment* 102 (feb. de 2015), págs. 96-104. ISSN: 13522310. DOI: 10.1016/j.atmosenv.2014.11.049. URL: https://linkinghub.elsevier.com/retrieve/pii/S1352231014009145.

[38] J Honaker, G King y M Blackwell. «Amelia II: A program for missing data, R package version 1.5., 2012». En: *Journal of Statistical Software* 45.7 (2011), págs. 1-3.

[39] Kent State University. *SPSS Tutorials: Pearson Correlation.* URL: https://libguides.library.kent.edu/SPSS/PearsonCorr. (accessed: 03.07.2020).

[40] «Spearman Rank Correlation Coefficient». En: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, págs. 502-505. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_379. URL: https://doi.org/10.1007/978-0-387-32833-1_379.

[41] «Kendall Rank Correlation Coefficient». En: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, págs. 278-281. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_211. URL: https://doi.org/10.1007/978-0-387-32833-1_211.

[42] D. N. Reshef y col. «Detecting Novel Associations in Large Data Sets». En: *Science* 334.6062 (dic. de 2011), págs. 1518-1524. ISSN: 0036-8075. DOI: 10.1126/science.1205438. URL: https://www.sciencemag.org/lookup/doi/10.1126/science.1205438.

[43] Fernando Berzal. *Redes Neuronales y Deep Learning*. Amazon, 2018. ISBN: 9781731265388.

[44] Simon O. Haykin. *Neural Networks and Learning Machines, 3rd Edition*. Pearson, 2009. ISBN: 9780131471399.

[45] François Chollet. *Deep Learning with Python*. Manning Shelter Island, 2009. ISBN: 9781617294433.

[46] Abigail See. *Natural Language Processing with Deep Learning. Lecture 7: Vanishing Gradients and Fancy RNNs*. URL: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. (accessed: 03.07.2020).

[47] Sepp Hochreiter y Jürgen Schmidhuber. «Long Short-Term Memory». En: *Neural Computation* 9.8 (nov. de 1997), págs. 1735-1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735.

[48] G. E. Dahl y col. «Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition». En: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1 (ene. de 2012), págs. 30-42. ISSN: 1558-7916. DOI: 10.1109/TASL.2011.2134090. URL: http://ieeexplore.ieee.org/document/5740583/.

[49] Geoffrey Hinton y col. «Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups». En: *IEEE Signal Processing Magazine* 29.6 (nov. de 2012), págs. 82-97. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2205597. URL: http://ieeexplore.ieee.org/document/6296526/.

[50] Dan Ciregan, Ueli Meier y Jurgen Schmidhuber. «Multi-column deep neural networks for image classification». En: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2012), págs. 3642-3649. ISSN: 10636919. DOI: 10.1109/CVPR.2012.6248110. arXiv: 1202.2745.

[51] Manohar Battula. *Time series forecasting with deep stacked unidirectional and bidirectional LSTMs. Towards Data Science.* URL: https://towardsdatascience.com/time-series-forecasting-with-deep-stacked-unidirectional-and-bidirectional-lstms-de7c099bd918. (accessed: 03.07.2020).

[52] Roan Gylberth. *Momentum Method and Nesterov Accelerated Gradient - Konvergen - Medium.* URL: https://medium.com/konvergen/momentum-method-and-nesterov-accelerated-gradient-487ba776c987. (accessed: 03.07.2020).

[53] Kevin Swersky Geoffrey Hinton Nitish Srivastava. *Neural Networks for Machine Learning. Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude.* URL: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. (accessed: 03.07.2020).

[54] Diederik P. Kingma y Jimmy Lei Ba. «Adam: A method for stochastic optimization». En: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (2015), págs. 1-15. arXiv: 1412.6980.

[55] P.J. Werbos. *Backpropagation Through Time: What It Does and How to Do It.* 1990. URL: http://ieeexplore.ieee.org/document/58337/?reload=true.

[56] Mario Dagrada. *ML time series forecasting the right way.End-to-end guide to predicting the future with machine learning. Towards Data Science.* URL: https://towardsdatascience.com/ml-time-series-forecasting-the-right-way-cbf3678845ff. (accessed: 03.07.2020).

[57] Thomas G. Dietterich. «Machine learning for sequential data: A review». En: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 2396 (2002), págs. 15-30. ISSN: 16113349. DOI: 10.1007/3-540-70659-3_2.

[58] Gianluca Bontempi. *Machine Learning Strategies for TimeSeries Prediction. Machine Learning Summer School (Hammamet, 2013). Machine Learning Group, Computer Science Department.* URL: http://di.ulb.ac.be/map/gbonte/ftp/time_ser.pdf. (accessed: 03.07.2020).

[59] Canonical. *Sistema operativo Ubutnu.* URL: https://ubuntu.com/. (accessed: 03.07.2020).

[60] Guido Van Rossum y Fred L. Drake. *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[61] Jet Brains. *IDE PyCharm*. URL: https://www.jetbrains.com/es-es/pycharm/. (accessed: 03.07.2020).

[62] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2014. URL: http://www.R-project.org/.

[63] Martın Abadi y col. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[64] Yangqing Jia y col. «Caffe: Convolutional Architecture for Fast Feature Embedding». En: *arXiv preprint arXiv:1408.5093* (2014).

[65] Adam Paszke y col. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». En: *Advances in Neural Information Processing Systems 32*. Ed. por H. Wallach y col. Curran Associates, Inc., 2019, págs. 8024-8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[66] Theano Development Team. «Theano: A Python framework for fast computation of mathematical expressions». En: *arXiv e-prints* abs/1605.02688 (mayo de 2016). URL: http://arxiv.org/abs/1605.02688.

[67] Frank Seide y Amit Agarwal. «CNTK». En: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, ago. de 2016, págs. 2135-2135. ISBN: 9781450342322. DOI: 10.1145/2939672.2945397. URL: https://dl.acm.org/doi/10.1145/2939672.2945397.

[68] Jeff Hale. *Deep Learning Framework Power Scores 2018. Towards DataScience*. URL: https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a. (accessed: 03.07.2020).

[69] NVIDIA. *CUDA parallel programing platform*. URL: https://developer.nvidia.com/cuda-zone. (accessed: 03.07.2020).

[70] Sharan Chetlur y col. «cuDNN: Efficient Primitives for Deep Learning». En: (oct. de 2014). arXiv: 1410.0759. URL: http://arxiv.org/abs/1410.0759.

[71] Francois Chollet y col. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[72] U.S. Department of Commerce. *National Oceanic and Atmospheric Administration*. URL: https://www.noaa.gov/. (accessed: 03.07.2020).

[73] Juan Gómez Romero. *PROFICIENT: Deep Learning for Energy-Efficient Building Control*. URL: https://jgromero.github.io/proficient/. (accessed: 03.07.2020).

[74] Rob J. Hyndman. *Interp: Linear interpolation for non-seasonal series.* URL: https://www.rdocumentation.org/packages/forecast/versions/8.12/topics/na.interp. (accessed: 03.07.2020).

[75] Abraham. Savitzky y M. J. E. Golay. «Smoothing and Differentiation of Data by Simplified Least Squares Procedures.» En: *Analytical Chemistry* 36.8 (jul. de 1964), págs. 1627-1639. ISSN: 0003-2700. DOI: 10.1021/ac60214a047. URL: https://pubs.acs.org/doi/abs/10.1021/ac60214a047.

[76] Isaac Godfried. *Attention for time series forecasting and classification.* URL: https://towardsdatascience.com/attention-for-time-series-classification-and-forecasting-261723e0006d. (accessed: 03.07.2020).

[77] RStudio Team. *RStudio: Integrated Development Environment for R.* RStudio, PBC. Boston, MA, 2020. URL: http://www.rstudio.com/.

[78] J. D. Hunter. «Matplotlib: A 2D graphics environment». En: *Computing in Science & Engineering* 9.3 (2007), págs. 90-95. DOI: 10.1109/MCSE.2007.55.

[79] The pandas development team. *pandas-dev/pandas: Pandas.* Ver. 0.24.2. Feb. de 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[80] Travis Oliphant. *NumPy: A guide to NumPy.* USA: Trelgol Publishing. [Online; accessed ¡today¿]. 2006–. URL: http://www.numpy.org/.

[81] Guido Van Rossum. *The Python Library Reference, release 3.8.2.* Python Software Foundation, 2020.

[82] Michael Waskom y col. *mwaskom/seaborn: v0.8.1 (September 2017).* Ver. v0.8.1. Sep. de 2017. DOI: 10.5281/zenodo.883859. URL: https://doi.org/10.5281/zenodo.883859.

[83] F. Pedregosa y col. «Scikit-learn: Machine Learning in Python». En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.

[84] Rakesh Vidya Chandra y Bala Subrahmanyam Varanasi. *Python requests essentials.* Packt Publishing Ltd, 2015.

[85] Hadley Wickham y col. «Welcome to the tidyverse». En: *Journal of Open Source Software* 4.43 (2019), pág. 1686. DOI: 10.21105/joss.01686.

[86] Achim Zeileis y Gabor Grothendieck. «zoo: S3 Infrastructure for Regular and Irregular Time Series». En: *Journal of Statistical Software* 14.6 (2005), págs. 1-27. DOI: 10.18637/jss.v014.i06.

[87] Rob Hyndman y col. *forecast: Forecasting functions for time series and linear models.* R package version 8.12. 2020. URL: http://pkg.robjhyndman.com/forecast.

[88] Alexander Kowarik y Matthias Templ. «Imputation with the R Package VIM». En: *Journal of Statistical Software* 74.7 (2016), págs. 1-16. DOI: 10.18637/jss.v074.i07.

Apéndice A

Visualización de los datos transformados

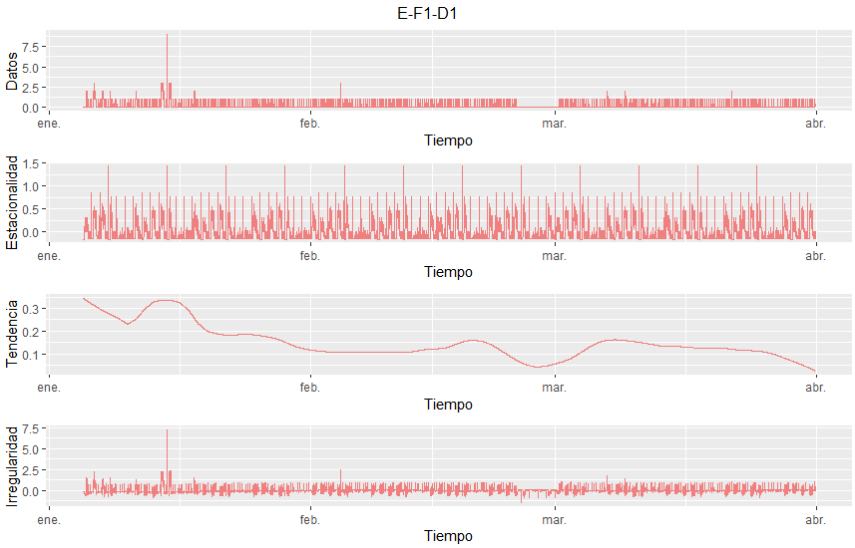
En este apartado se expone toda la información complementaria de la sección 4.3.2.

Para llevar a cabo la visualización y el análisis de la descomposición y autocorrelación de las distintas variables, las agruparemos de la misma manera que en la sección 4.2.3. Por tanto, vamos a empezar con las variables de consumo eléctrico.

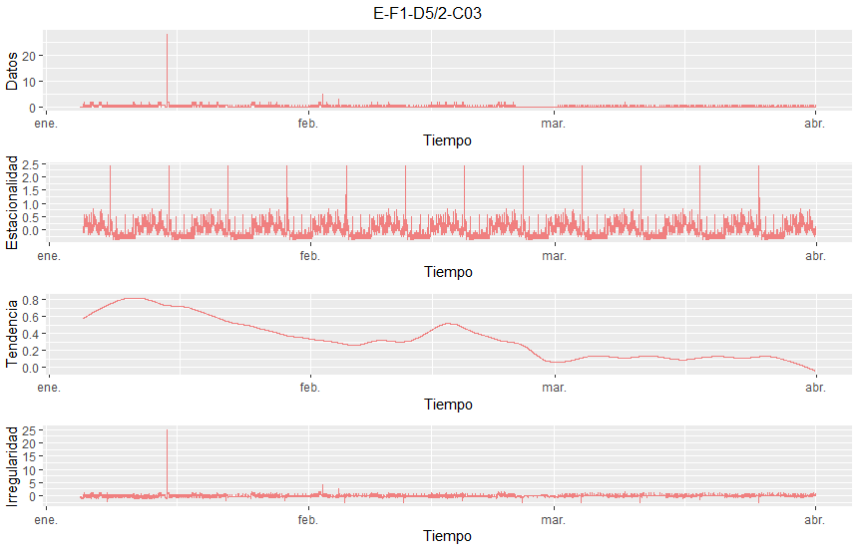
A.1. Variables de consumo eléctrico

En esta subsección vamos a visualizar y analizar la descomposición y autocorrelación de las variables de consumo eléctrico, para lo cual las hemos agrupado según la planta de la zona piloto del edificio ICPE a la que pertenecen.

En primer lugar vamos con las variables de consumo eléctrico de la primera planta.

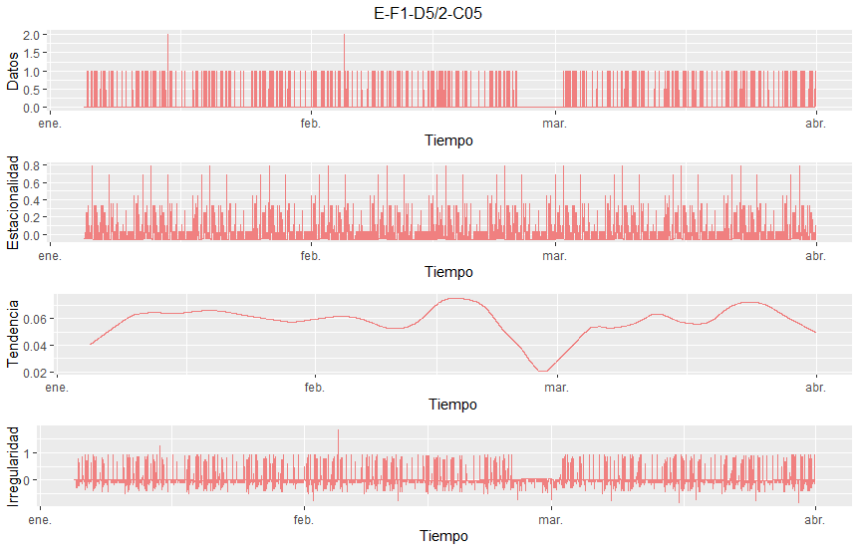


(a) E-F1-D1.

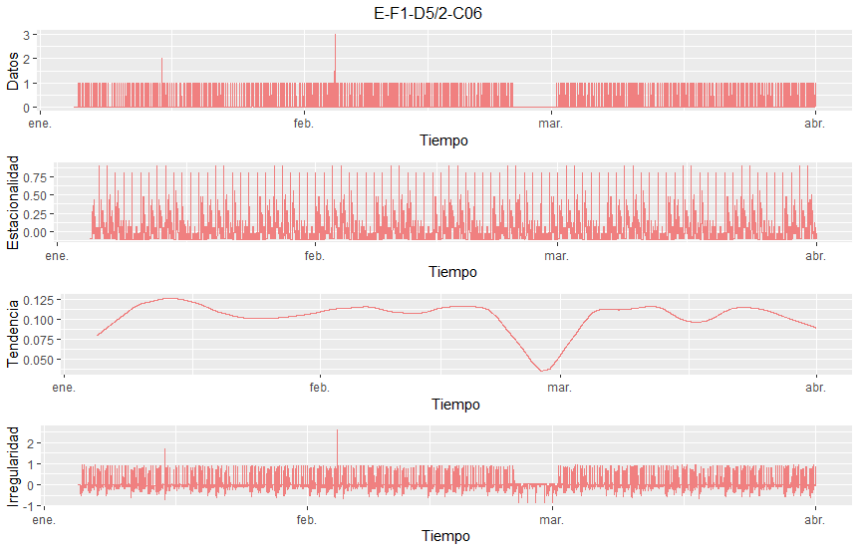


(b) E-F1-D5/2-C03.

Figura A.1: Gráficas de descomposición de las variables de consumo eléctrico de la planta 1 de la zona piloto del edificio ICPE.



(a) E-F1-D5/2-C05.



(b) E-F1-D5/2-C06.

Figura A.2: Gráficas de descomposición de las variables de consumo eléctrico de la planta 1 de la zona piloto del edificio ICPE.

En las figura A.1 y A.2 podemos ver las gráficas de descomposición de las variables de consumo eléctrico de la primera planta de la zona piloto. En cada subfigura podemos ver las cuatro gráficas de descomposición de una variable concreta, las cuales, si las observamos de arriba abajo nos muestran los valores de consumo real de la variable, su estacionalidad, su tendencia y sus irregularidades. En todas estas gráficas tenemos el tiempo en el eje x y los valores de la variable en el eje y.

Dicho esto, observando las dos figuras vemos que:

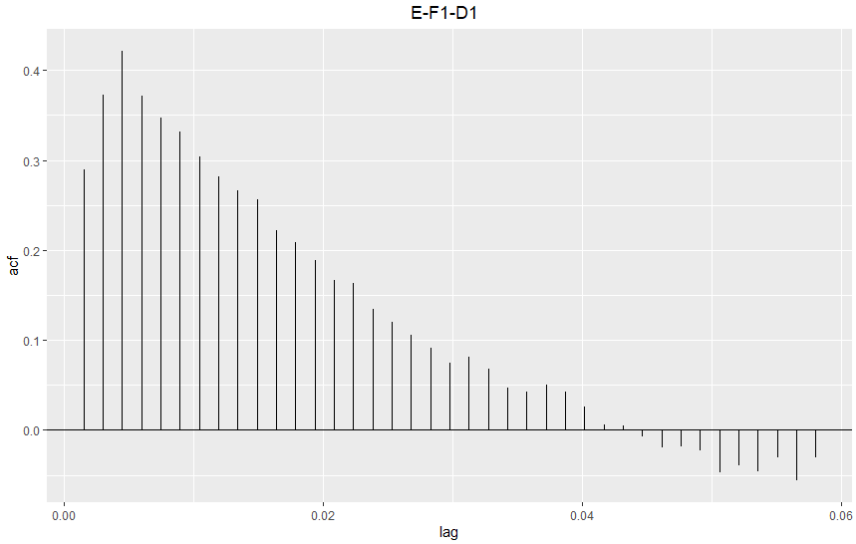
- En todas las variables hay un período de tiempo entre las semanas 8 y 10 durante el cual no hay consumo. Esto se debe a que dicho período se corresponde con los días 26, 27, 28 y 29 de febrero, que como ya hemos mencionado en apartados anteriores son días en los que no se tiene datos de ninguna variable de consumo.

- Si miramos las gráficas Datos y Estacionalidad de cada variable vemos que hay patrones de consumo que se repiten a lo largo del tiempo. Más en concreto, puede verse que estos patrones se repiten en el período de una semana, lo cual tiene sentido, pues en un mismo día (por ejemplo Jueves) de distintas semanas se deben registrar consumos similares en un mismo área. De hecho, se aprecia en la gráfica de Estacionalidad como el consumo es alto durante unos pocos días y después baja durante un par de días. Los días en los que el consumo es más alto son laborales mientras que aquellos en los que el consumo es más bajo se corresponderían con los fines de semana. Esta estacionalidad se ve más acentuada en la variable E-F1-D5/2-C03, lo que nos puede indicar que el área del edificio correspondiente a la misma tiene unos patrones de utilización y consumo más regulares que el resto.

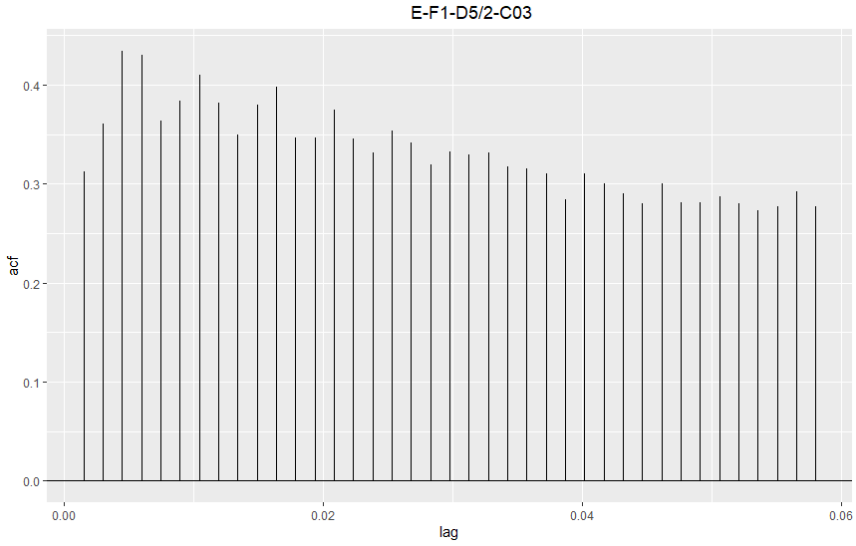
- Si miramos la gráfica de tendencia de cada variable, se observa que el consumo registrado por las variables E-F1-D1 y E-F1-D5/2-C03 tiende a disminuir (obviando los días 26, 27, 28 y 29). Mientras que el consumo registrado por las variables E-F1-D5/2-C05 y E-F1-D5/2-C06 tiende a mantenerse constante. Esto puede deberse a que las zonas de las áreas de las dos primeras variables se utilicen más en los meses de enero y febrero pero menos en marzo. Sin embargo, este no parece ser el caso ya que los patrones de consumo registrados en estas variables se mantienen también en marzo. Entonces lo que está ocurriendo es que parece que el consumo decae en estas dos variables pero en realidad lo hace de manera muy leve, pues en sus gráficas de Tendencia podemos ver que el consumo disminuye a lo largo de los meses desde 0.3 hasta 0.1.

- Si atendemos a la gráfica de Irregularidad de cada variable vemos que en general todas las variables tienen bastantes componentes irregulares o aleatorias que son incluso mayores que las componentes estacionales (mirar eje y de las distintas gráficas). Esto nos está mostrando la poca estacionalidad y regularidad en general de las observaciones de estas cuatro variables.

Ahora vamos a ver las gráficas de autocorrelación de estas cuatro variables.

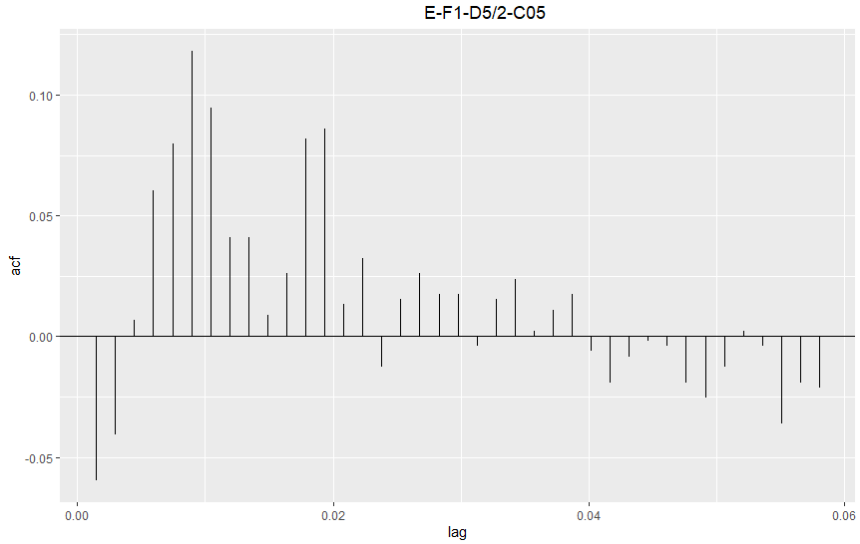


(a) E-F1-D1.

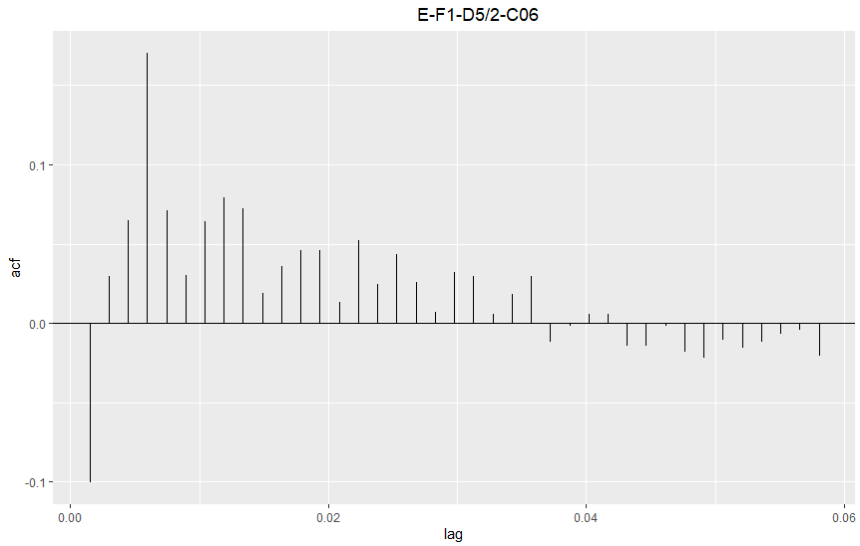


(b) E-F1-D5/2-C03.

Figura A.3: Gráficas de autocorrelación de las variables de consumo eléctrico de la planta 1 de la zona piloto del edificio ICPE.



(a) E-F1-D5/2-C05.



(b) E-F1-D5/2-C06.

Figura A.4: Gráficas de autocorrelación de las variables de consumo eléctrico de la planta 1 de la zona piloto del edificio ICPE.

En las figuras A.3 y A.4 vemos en cada subfigura la correlación (eje y) entre la serie temporal de una variable con distintas versiones atrasadas de sí misma un número concreto de lags o timesteps (eje x). Dicho esto, en estas dos figuras vemos lo siguiente.

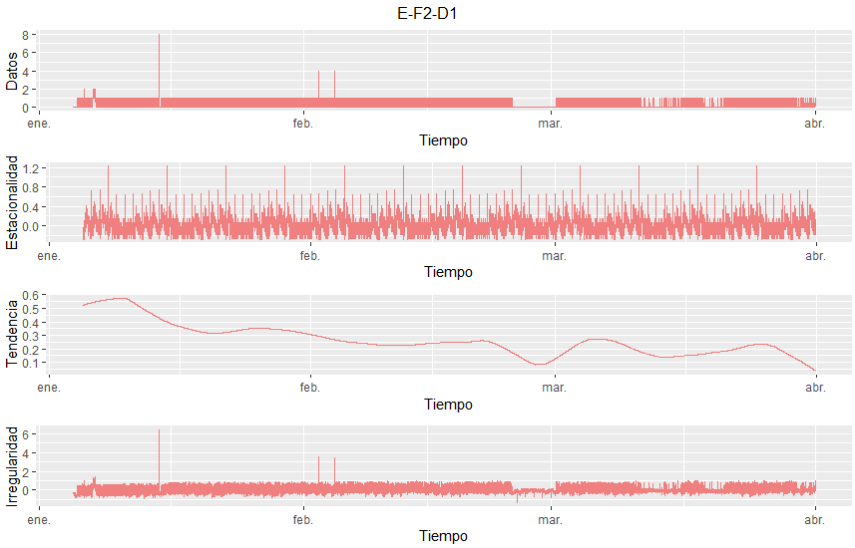
- Las variables E-F1-D1 y E-F1-D5/2-C03 (esta última sobre todo) presentan los niveles más elevados de autocorrelación, lo que significa que

sus valores futuros son los que más correlados están con sus valores pasados y por tanto, estas dos variables serían las más adecuadas de entre las cuatro para poder predecir sus valores futuros a partir de los pasados. Eso sí, los niveles de correlación de estas dos variables no son demasiado altos, pues en el lag en el que más correlación alcanzan con sigo mismas vemos un nivel de 0.4 de correlación.

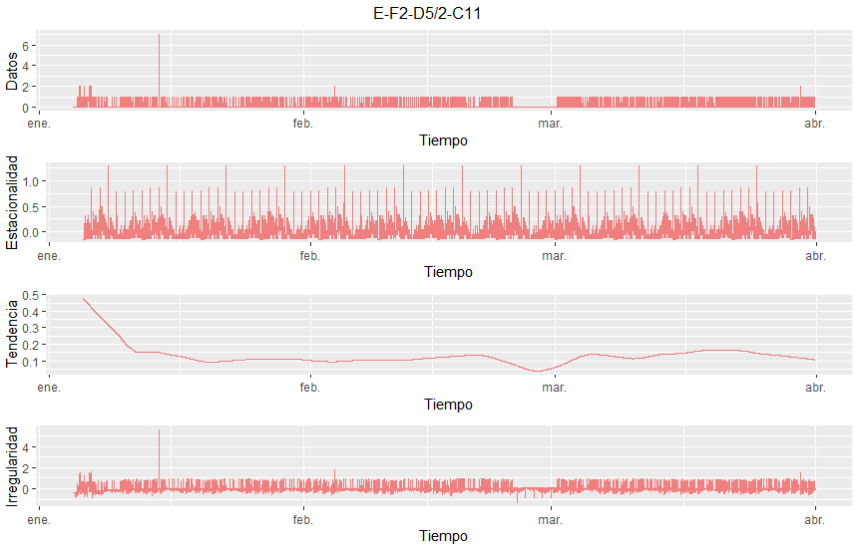
Este mayor nivel de autocorrelación de estas dos variables se debe a su mayor estacionalidad y regularidad, que hacen que a partir de valores pasados se puedan predecir valores futuros con más facilidad.

- Las variables E-F1-D5/2-C05 y E-F1-D5/2-C06 tienen unos niveles de correlación más bajos, lo cual se debe a su menor estacionalidad y mayor irregularidad.

A continuación vamos a ver las gráficas de descomposición de las variables de consumo eléctrico de la segunda planta.

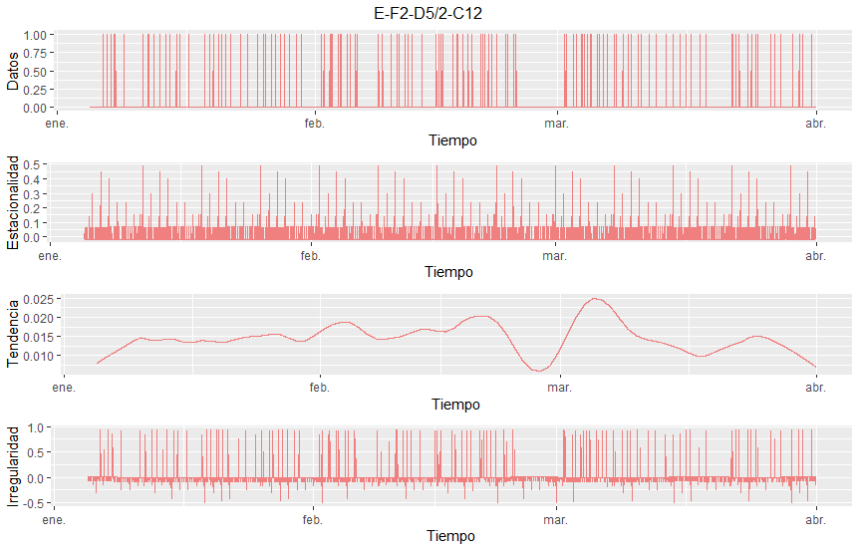


(a) E-F2-D1.

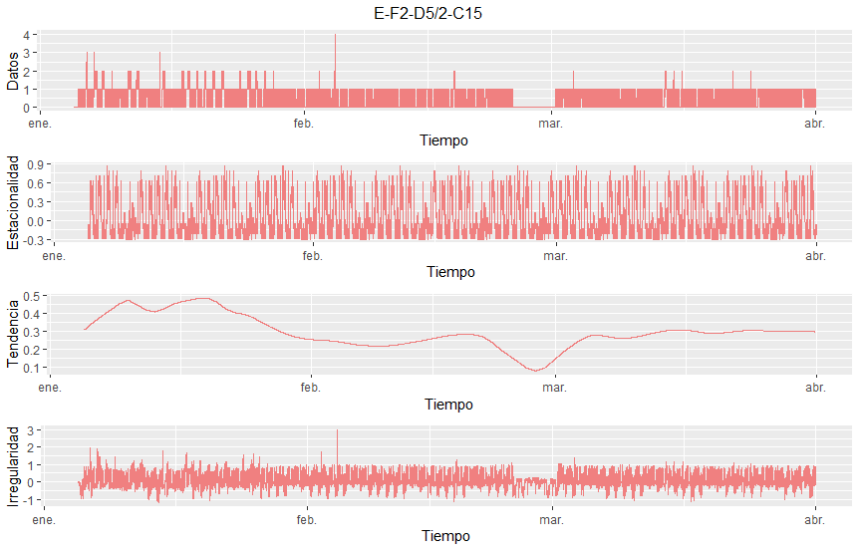


(b) E-F2-D5/2-C11.

Figura A.5: Gráficas de descomposición de las variables de consumo eléctrico de la planta 2 de la zona piloto del edificio ICPE.



(a) E-F2-D5/2-C12.



(b) E-F2-D5/2-C15.

Figura A.6: Gráficas de descomposición de las variables de consumo eléctrico de la planta 2 de la zona piloto del edificio ICPE.

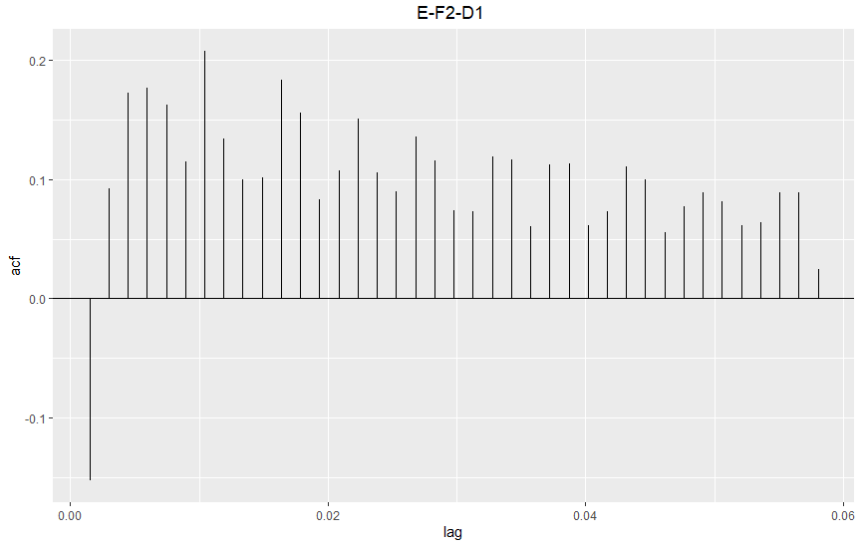
En las figuras A.5 y A.6 vemos en cada subfigura la descomposición de una variable concreta de consumo eléctrico de la planta 2. En estas subfiguras vemos lo siguiente.

- Todas las variables registran consumos eléctricos entre 0 y 1, salvo algunos «picos» puntuales, que pueden ser debidos a una mayor demanda de electricidad en la zona causada por una mayor presencia

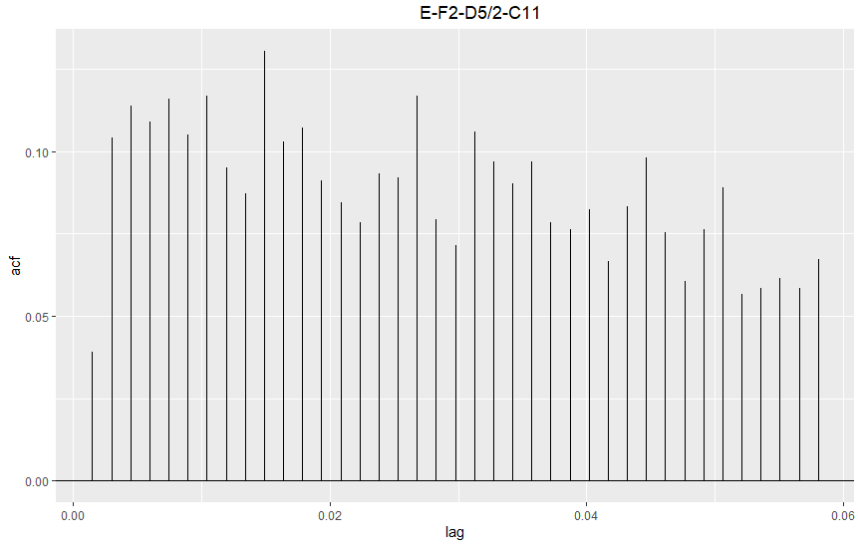
de trabajadores, mayor número de equipos electrónicos en uso en ese momento, fallos en el sensor, en el sistema eléctrico, etc.

- Al igual que sucedía con los sensores eléctricos de la primera planta, estos tienen un cierto comportamiento estacionario en el período de una semana. Aunque eso sí, dicha estacionalidad no está demasiado acentuada en ninguno de ellos, siendo los sensores E-F2-D1 y E-F2-D5/2-C15 los que más estacionalidad presentan.
- En cuanto a la tendencia, vemos que más o menos el consumo de las cuatro variables tiende a mantenerse en general, siendo la tendencia que más varía la de las variables E-F1-D1 y E-F2-D5/2-C11.
- Finalmente vemos que la componente irregular de cada variable es bastante elevada, lo cual encaja con la baja estacionalidad general.

Ahora veremos de forma gráfica la autocorrelación de cada una de las cuatro variables de consumo eléctrico de la segunda planta.

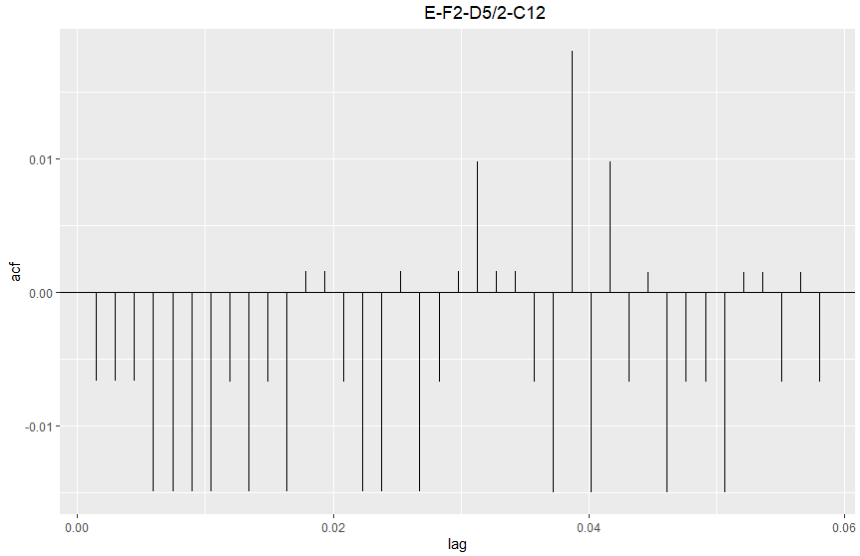


(a) E-F2-D1.

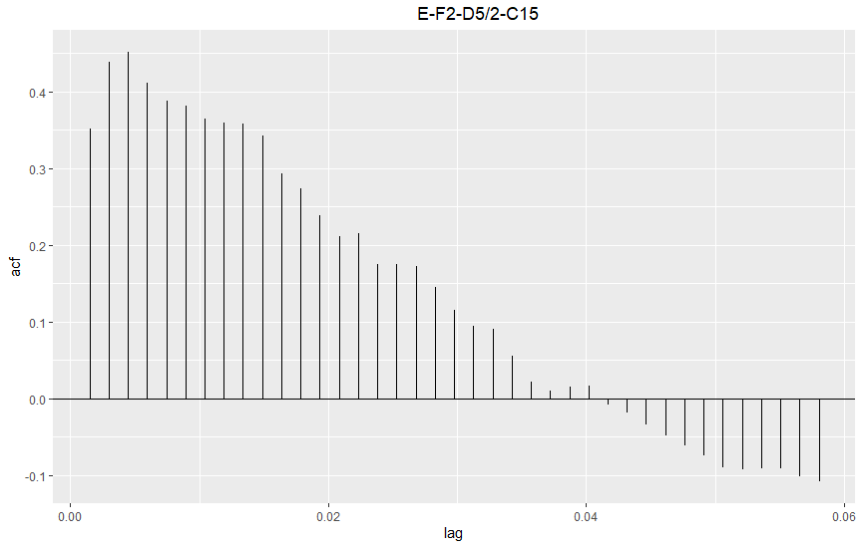


(b) E-F2-D5/2-C11.

Figura A.7: Gráficas de autocorrelación de las variables de consumo eléctrico de la planta 2 de la zona piloto del edificio ICPE.



(a) E-F2-D5/2-C12.



(b) E-F2-D5/2-C15.

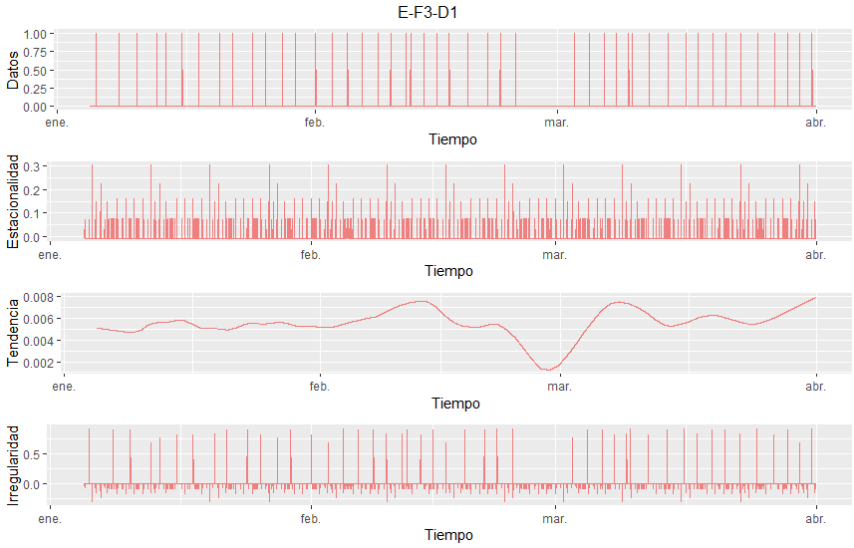
Figura A.8: Gráficas de autocorrelación de las variables de consumo eléctrico de la planta 2 de la zona piloto del edificio ICPE.

En las subfiguras de las figuras A.7 y A.8 se observa lo siguiente:

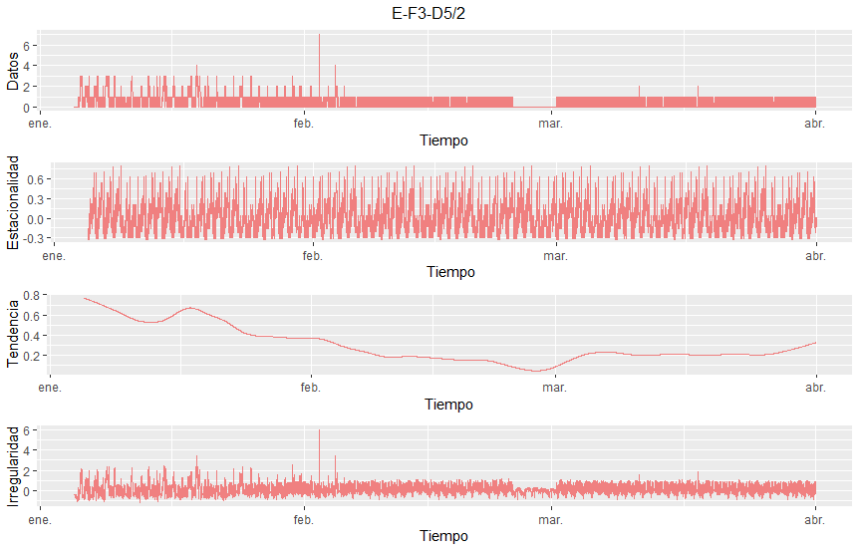
- Todas las variables tienen niveles de correlación bajos, siendo las que más las variables E-F2-D1 y E-F2-D5/2-C15, lo cual tiene sentido pues como dijimos anteriormente son las que más estacionalidad y menor irregularidad presentan.

- En general, los valores futuros de cada variable están poco correlados con los valores pasados, por lo que no son muy aptas para ser predichas.

Finalmente vamos con la descomposición y autocorrelación de las variables de consumo eléctrico de la tercera planta.



(a) E-F3-D1.



(b) E-F3-D5/2.

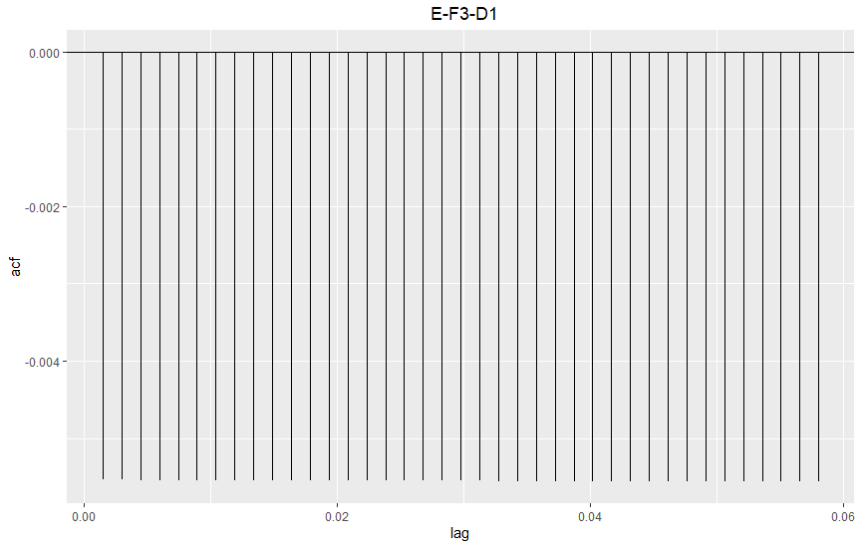
Figura A.9: Gráficas de descomposición de las variables de consumo eléctrico de la planta 3 de la zona piloto del edificio ICPE.

En las subfiguras a y b de la figura A.9 podemos ver la descomposición

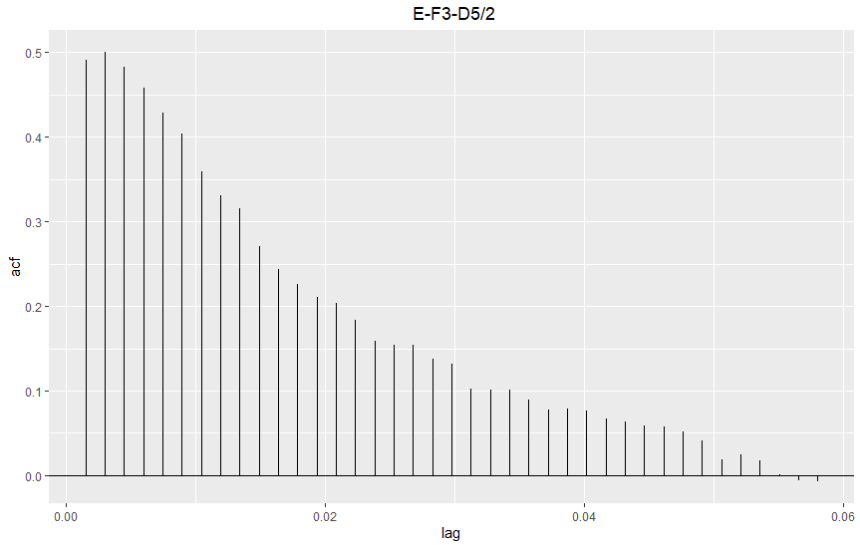
de las variables E-F3-D1 y E-F3-D5/2 respectivamente. En ellas vemos que:

- La variables E-F3-D5/2 registra un consumo mayor que E-F3-D1, lo que coincide con lo observado durante el análsisis exploratorio en la sección 4.2.3.1, es decir, que en el área D5/2 hay un mayor consumo eléctrico. Por su parte, la variable E-F3-D1 registra un consumo mucho menor y menos estacionario, dándonos una idea del irregular uso de la zona D1 en la tercera planta.
- La tendencia de la variable E-F3-D1 se mantiene más o menos con algunas variaciones bruscas, mientras que, la de la variable E-F3-D5/2 disminuye conforme pasan los meses.
- En ambas variables hay bastante irregularidad.

A continuación vamos con las gráficas de autocorrelación de E-F3-D1 y E-F3-D5/2.



(a) E-F3-D1.



(b) E-F3-D5/2.

Figura A.10: Gráficas de autocorrelación de las variables de consumo eléctrico de la planta 3 de la zona piloto del edificio ICPE.

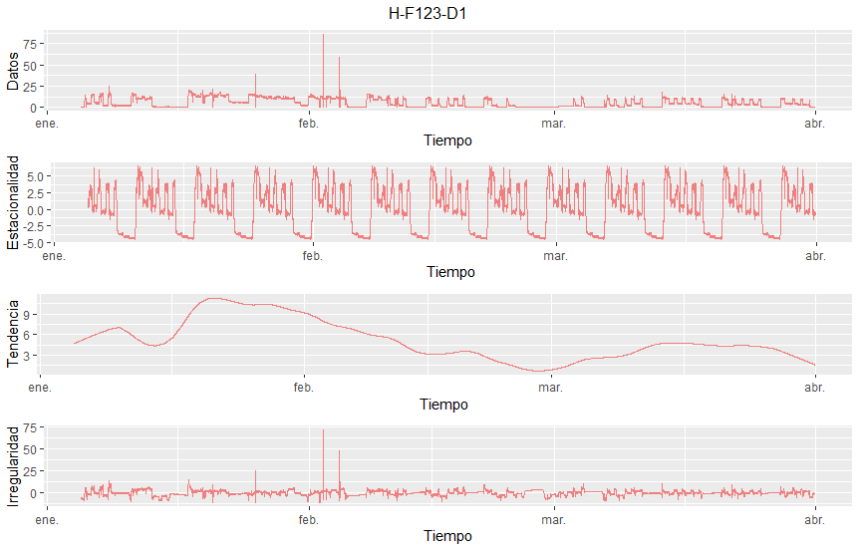
En las subfiguras a y b de la figura A.10 se observa que la variable E-F3-D1 tiene niveles de correlación muy bajos. Esto nos indica que el consumo en esta zona es tan irregular que sería difícil predecirlo. Por su parte, la variable E-F3-D5/2 llega a tener en algunos lags un nivel de correlación superior a

0.5.

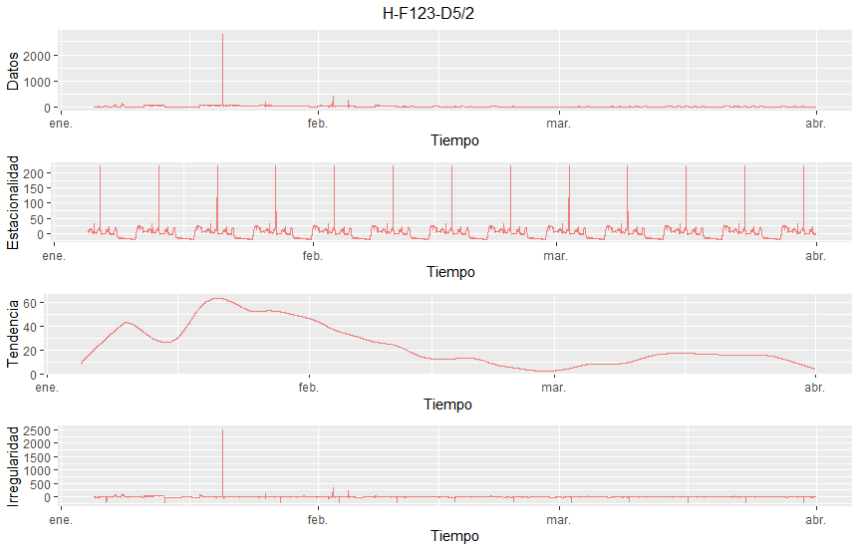
A.2. Variables de consumo de calefacción

A continuación vamos a visualizar y analizar la descomposición y autocorrelación de las variables de consumo de calefacción. Para ello, se han formado dos grupos, uno con las variables que miden el consumo de calefacción total de un área concreta en las plantas 1, 2 y 3 y otro con las variables de las plantas 1 y 2.

Así, empezaremos viendo la descomposición y autocorrelación de las variables de consumo total de calefacción en las plantas 1, 2 y 3.



(a) H-F123-D1.



(b) H-F123-D5/2.

Figura A.11: Gráficas de descomposición de las variables de consumo de calefacción de las plantas 1, 2 y 3 de la zona piloto del edificio ICPE.

En la figura A.11 vemos la descomposición de las variables H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b). En ellas se observa lo siguiente:

- En la gráfica de Datos de ambas subfiguras, se ve que ambas variables registran grandes consumos de calefacción, los cuales siguen patrones más o menos regulares salvo algún punto en el que se disparan. Así, vemos que la variable H-F123-D5/2 llega a registrar un consumo

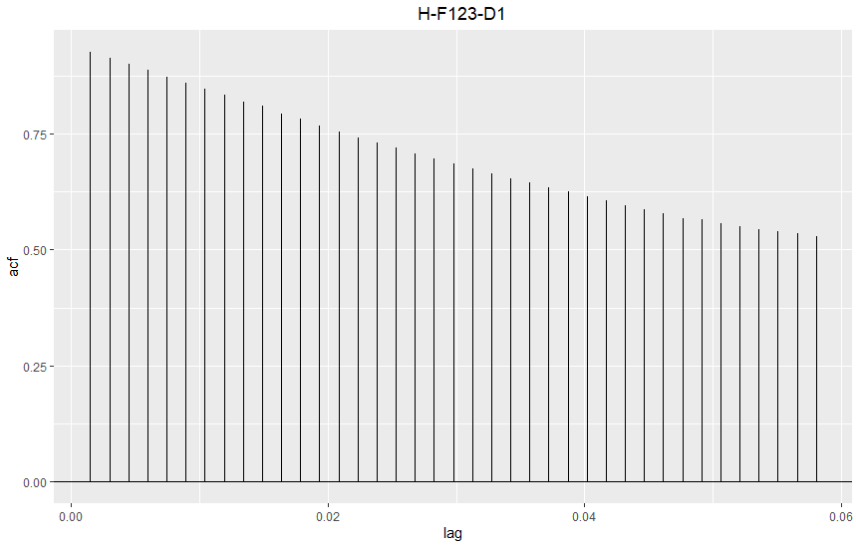
puntual enorme superior a 2000 KW (el consumo normal máximo es aproximadamente 70 KW). Estas elevaciones puntuales en el consumo se pueden deber a bajas temperaturas externas o a fallos en el propio sensor y pueden ser considerados outliers de acuerdo con la descripción que dimos de los mismos en el apartado 2.3.3.

- En la gráfica de Estacionalidad de ambas subfiguras se observa que ambas variables presentan una estacionalidad semanal bastante marcada. De esta forma, vemos que en cada patrón que se repite hay unos valores de consumo elevados que luego decaen durante un breve período de tiempo. Los valores elevados se corresponden con los días laborales (Lunes a Viernes) que es cuando hay personas trabajando y por tanto cuando más consumo hay. Mientras que los valores bajos de consumo tienen lugar durante los fines de semana, es decir, cuando no hay nadie trabajando en el edificio. Estos patrones son similares a los ya observados para la variable de ocupación, lo que deja entrever cierta relación entre esta última y las variables de consumo de calefacción de las plantas 1, 2 y 3.

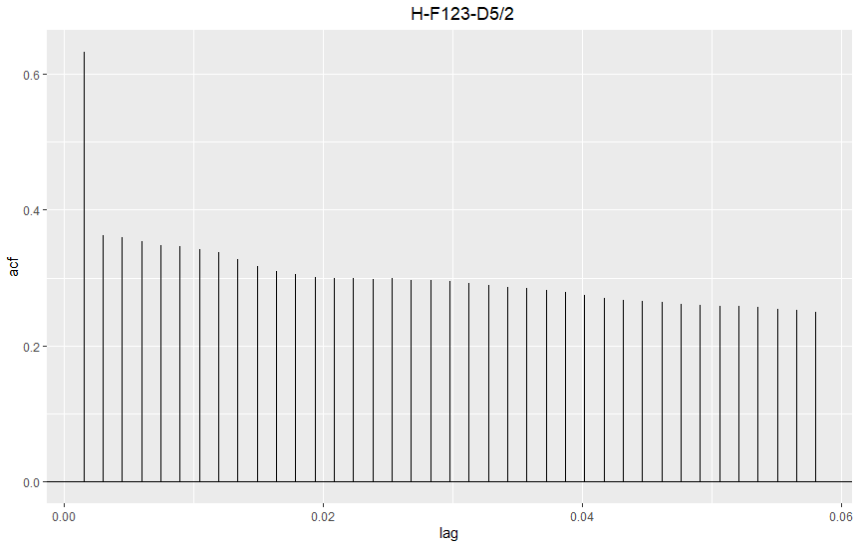
- En la gráfica Tendencia de ambas variables se ve que sus valores tienden a descender conforme avanzamos desde Enero hasta Marzo. Esto tiene sentido, pues conforme nos acercamos a Marzo nos nos encontramos más cerca de la primavera lo que significa un aumento en la temperatura exterior del edificio, dando lugar a un menor uso de la calefacción.

- En la gráfica de Irregularidad de ambas variables se ve que presentan pocos componentes irregulares, lo cual encaja con su marcada estacionalidad.

Dicho esto, vamos a ver las gráficas de autocorrelación de las variables H-F123-D1 y H-F123-D5/2.



(a) H-F123-D1.



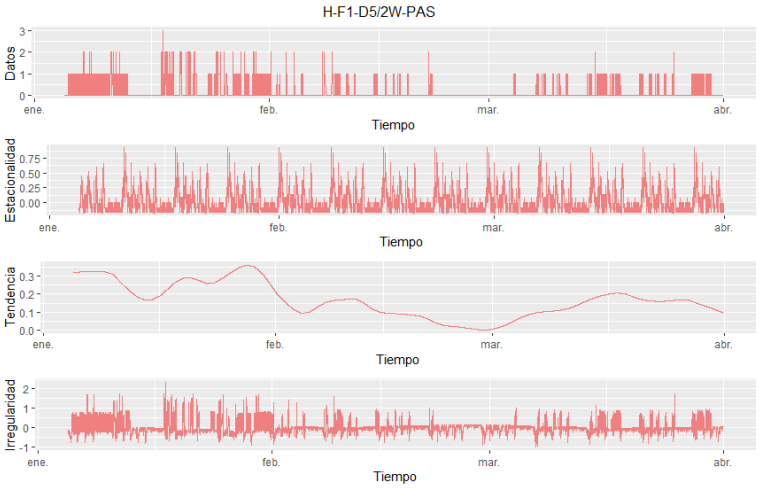
(b) H-F123-D5/2.

Figura A.12: Gráficas de autocorrelación de las variables de consumo de calefacción de las plantas 1, 2 y 3 de la zona piloto del edificio ICPE.

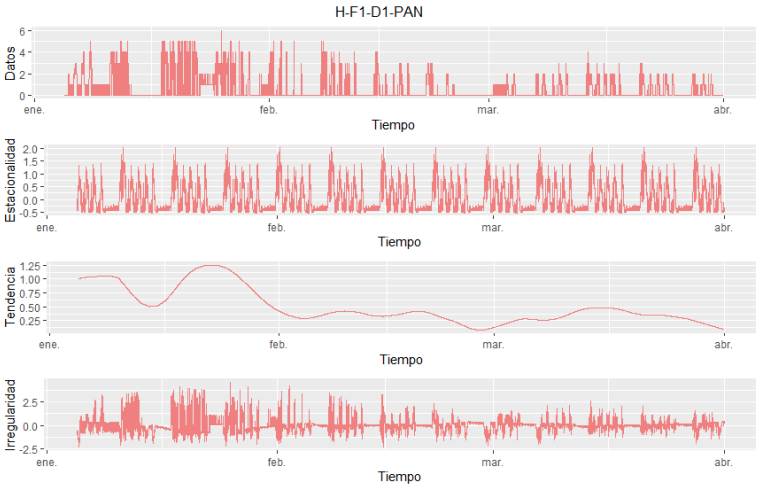
Si observamos las subfiguras a y b de la figura A.12 vemos que las variables H-F123-D1 y H-F123-D5/2 presentan unos niveles de correlación elevados. En especial la variables H-F123-D1, que en bastantes lags supera el 0.8 de correlación consigo misma. Esto nos indica que los valores de ambas

variables dependen o están relacionados con los valores pasados, lo cual hace que ambas sean susceptibles de ser predichas a partir de sus propios valores.

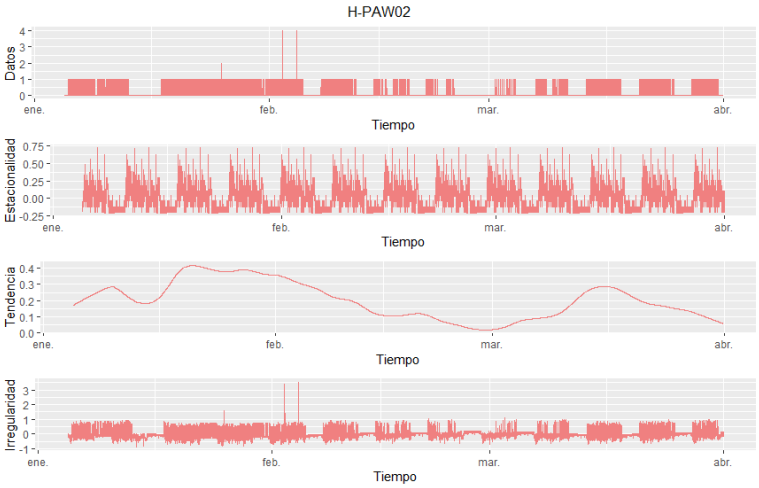
Vamos finalmente a ver las gráficas de descomposición y autocorrelación las variables de consumo de calefacción de las plantas 1 y 2.



(a) H-F1-D5/2W-PAS.

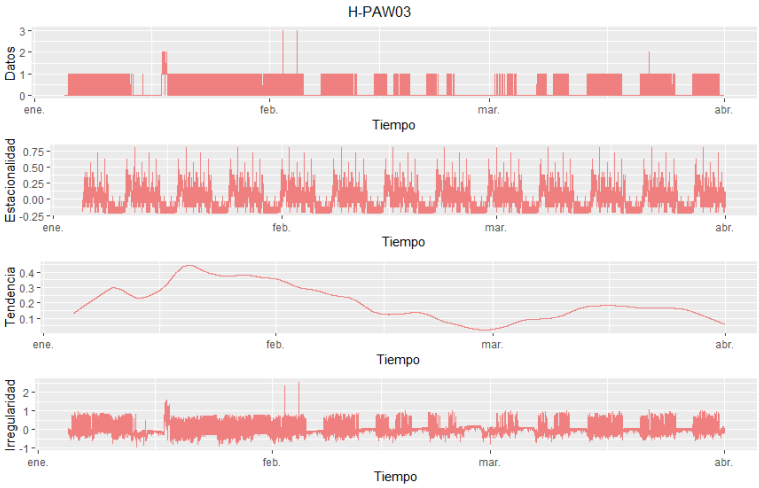


(b) H-F1-D1-PAN.

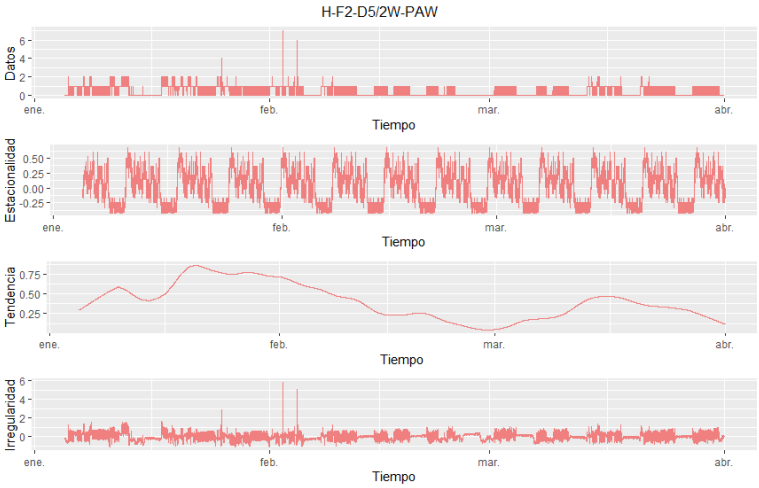


(c) H-PAW02.

Figura A.13: Gráficas de descomposición de las variables de consumo de calefacción de las plantas 1 y 2 de la zona piloto del edificio ICPE.



(a) H-PAW03.



(b) H-F2-D5/2W-PAW.

Figura A.14: Gráficas de descomposición de las variables de consumo de calefacción de las plantas 1 y 2 de la zona piloto del edificio ICPE.

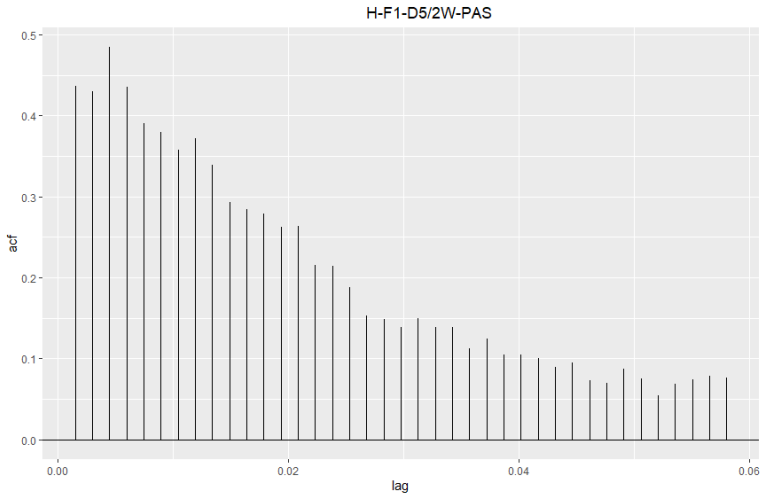
En las figuras A.13 y A.14 vemos las gráficas de descomposición de las variables H-F1-D5/2W-PAS, H-F1-D1-PAN, H-PAW02, H-PAW03 y H-F2-D5/2-PAW, en las cuales vemos los siguiente:

- Si observamos la gráfica de Datos de las cinco variables vemos que en todas ellas el consumo es mayor y más irregular en los meses de Enero y Febrero, y menor y más regular en Marzo. Esto puede deberse a que las temperaturas son muy bajas e inestables en los meses de Enero y Febrero, dando lugar a un mayor uso de la calefacción y a aumentos y disminuciones bruscas en su consumo y a que los equipos de calefacción funcionan en su mejor banda de rendimiento cuando no se les exige

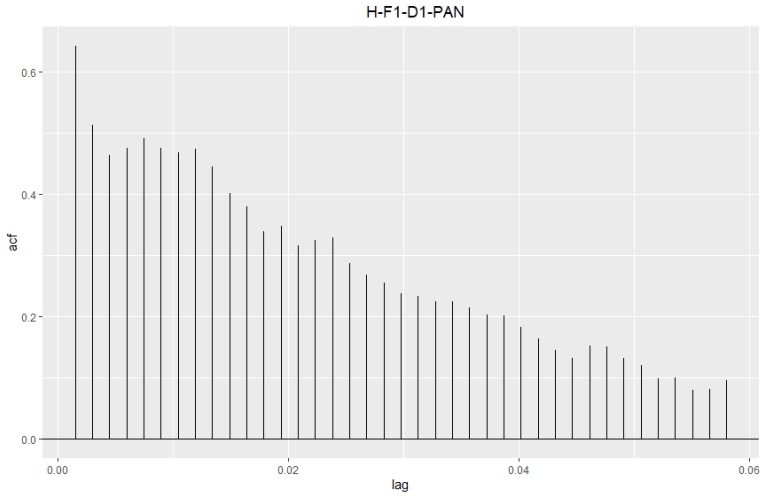
tanto y por tanto son más estables en el consumo.

- Si observamos la gráfica de Estacionalidad de cada variable vemos que todas presentan cierto comportamiento estacional semanal, siendo la variable H-F2-D5/2W-PAW la que más estacionalidad refleja.
- Mirando las gráficas de tendencia vemos que el consumo registrado por cada variable va disminuyendo, lo que confirma que en el mes de Marzo se consume menos calefacción que en los meses de Enero y Febrero.
- Finalmente vemos que las cinco variables presentan componentes irregulares, siendo la que menos la variables H-F2-D5/2W-PAW, lo que encaja con su mayor comportamiento estacional.

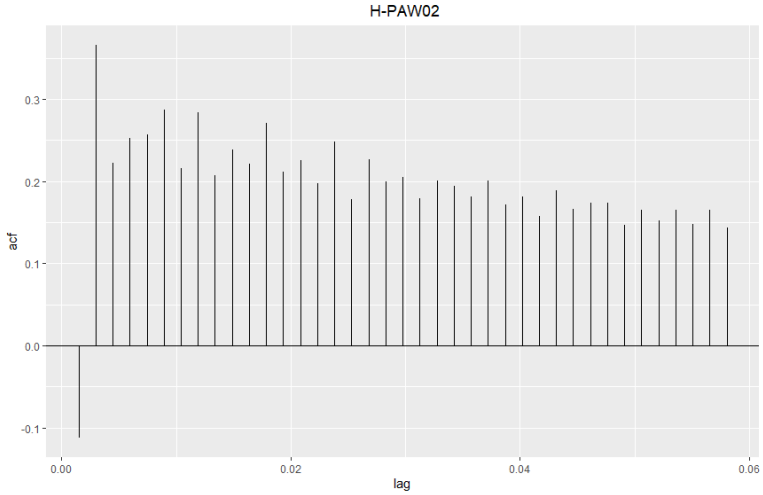
Ahora vamos con las gráficas de autocorrelación de estas cinco variables.



(a) H-F1-D5/2W-PAS.

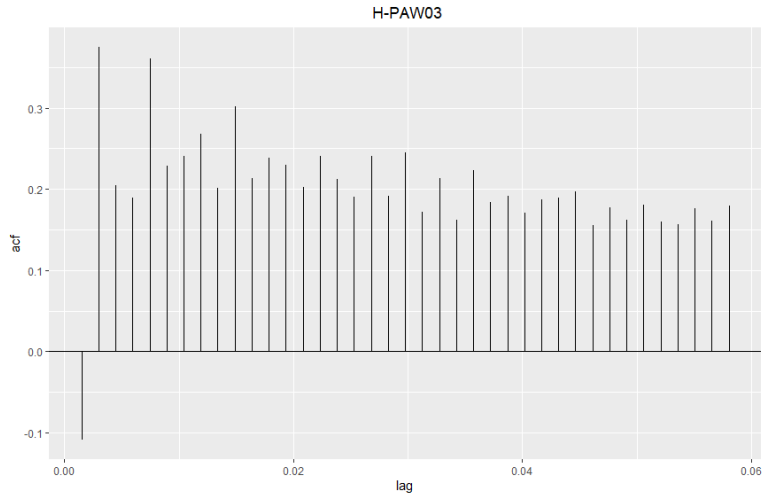


(b) H-F1-D1-PAN.

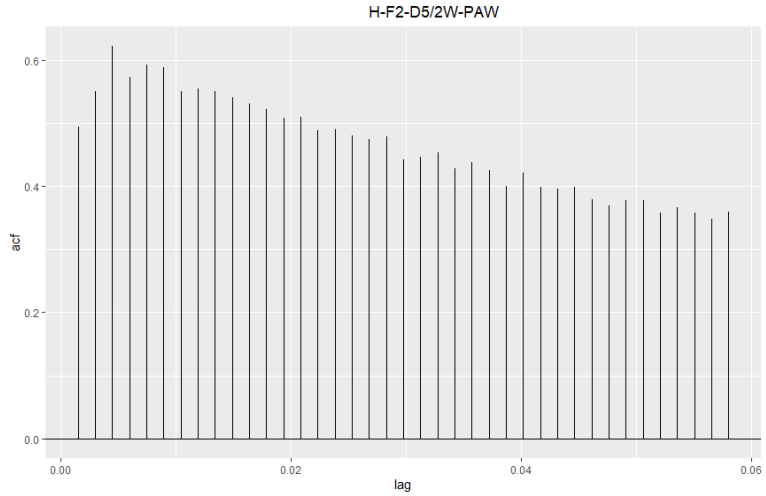


(c) H-PAW02.

Figura A.15: Gráficas de autocorrelación de las variables de consumo de calefacción de las plantas 1 y 2 de la zona piloto del edificio ICPE.



(a) H-PAW03.



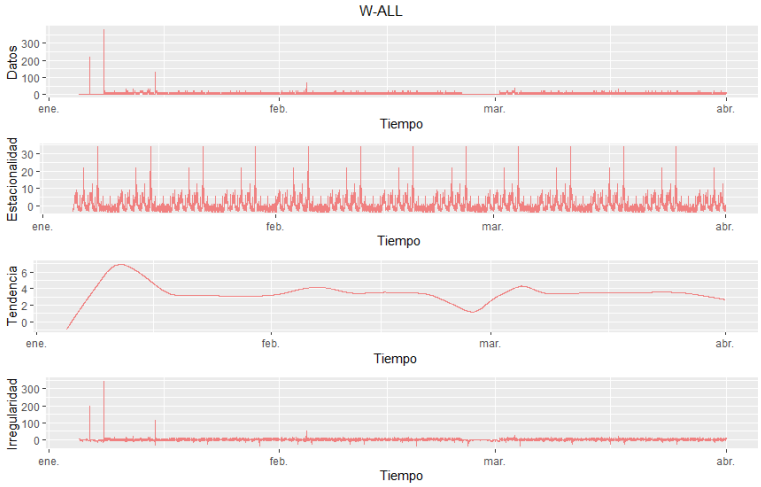
(b) H-F2-D5/2W-PAW.

Figura A.16: Gráficas de autocorrelación de las variables de consumo de calefacción de las plantas 1 y 2 de la zona piloto del edificio ICPE.

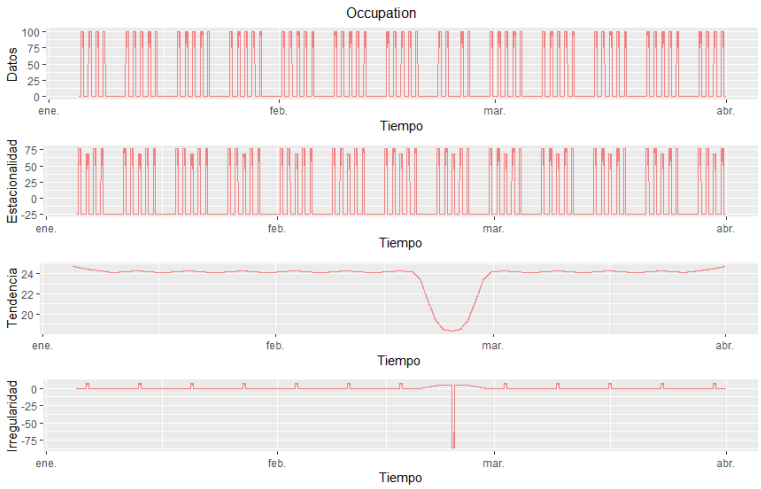
En las figuras A.15 y A.16 vemos las gráficas de autocorrelación de las variables H-F1-D5/2W-PAS, H-F1-D1-PAN, H-PAW02, H-PAW03 y H-F2-D5/2-PAW, en las cuales vemos que todas presentan niveles elevados de autocorrelación, sobre todo las variables H-F1-D1-PAN y H-F2-D5/W-PAW. En especial esta última, lo que concuerda con su mayor estacionalidad y regularidad.

A.3. Variables de consumo de agua, ocupación y temperatura exterior

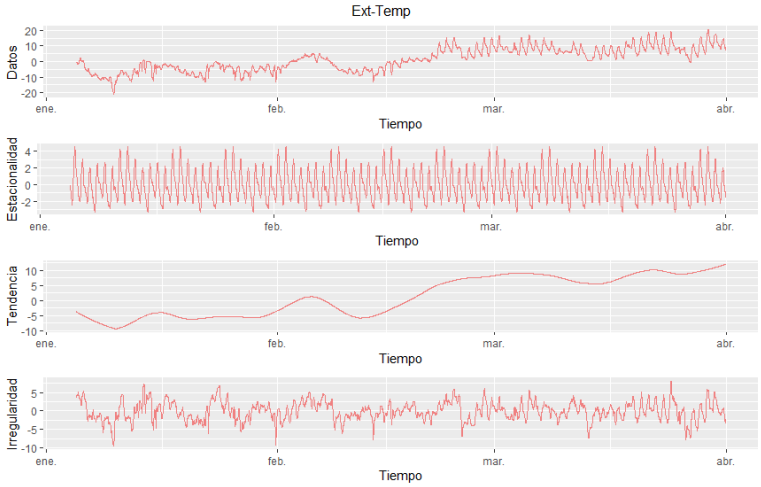
Finalmente vamos a ver y analizar las gráficas de descomposición y autocorrelación de las variables de consumo de agua, ocupación y temperatura externa.



(a) W-ALL.



(b) Occupation.



(c) Ext-Temp.

Figura A.17: Gráficas de descomposición de las variables de consumo de agua, ocupación y temperatura externa.

En la figura A.17 podemos ver las gráficas de descomposición de las variables W-ALL (subfigura a), Occupation (subfigura b) y Ext-Temp (subfigura c), en las cuales se observa los siguiente:

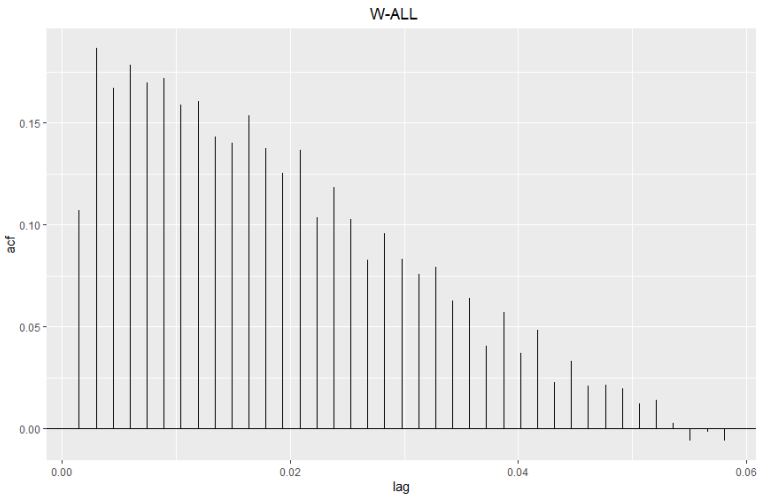
- Si atendemos a la gráfica de Datos de cada variable, vemos que la variable de consumo de agua registra valores estables de consumo a lo largo de los meses d enero, febrero y marzo salvo algunos consumos bruscos puntuales en enero y febrero. Por otra parte, la variable de ocupación, como ya se comentó en la sección 4.2.3.3 registra la misma ocupación siempre en los mismos días de la semana, salvo el día 24 de febrero el cual fue festivo en Bucarest y en el cual puede verse que los valores de ocupación son del 0 %. Por último, la variable de temperatura exterior vemos como registra valores cada vez mayores conforme nos acercamos a marzo debido a la llegada de la primavera.

- Mirando la gráfica de Estacionalidad de cada variable puede verse que la variable de consumo de agua presenta una componente estacional en el período de una semana, aunque no está tan definida como en el caso de las variables de ocupación y temperatura externa. En especial vemos como la gráfica de Estacionalidad de la variable de ocupación es prácticamente idéntica a su gráfica de Datos, por lo que es una variable puramente estacional.

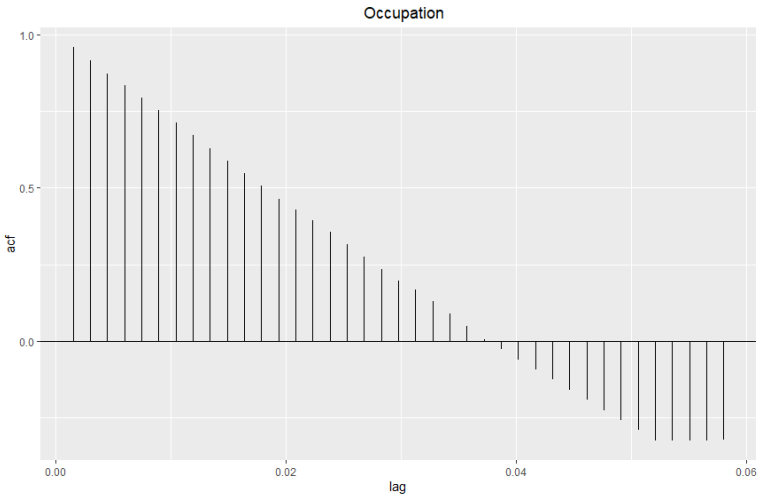
- Si observamos las gráficas de tendencia, vemos que la media de los valores de consumo de agua y ocupación se mantienen. En el caso de esta última, parece que descienden de forma brusca, pero en realidad solo desciende de 24 a 18 si miramos bien el eje y. Este descenso es provocado porque la ocupación es del 0 % el día 24 de febrero, que como hemos dicho es festivo. Por su parte, vemos que la media de la variable de temperatura exterior tiende a aumentar, lo que coincide con lo observado en su gráfica de Datos.

- En las gráficas de Irregularidad se observa que la variable de consumo de agua presenta varias irregularidades sobre todo en los meses de enero y febrero. Mientras, la variable de ocupación apenas presenta irregularidades salvo un valor muy negativo debido otra vez al día 24 de febrero. Por último, la variable de temperatura exterior presenta, al igual que la variable de consumo de agua bastantes irregularidades.

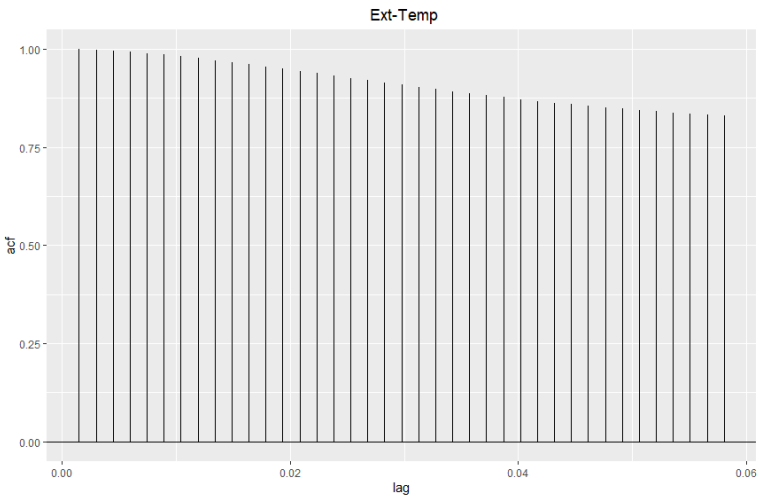
Ahora vamos con las gráficas de autocorrelación de estas tres variables.



(a) W-ALL.



(b) Occupation.



(c) Ext-Temp.

Figura A.18: Gráficas de sutocorrelación de las variables de consumo de agua, ocupación y temperatura externa.

En la figura A.18 podemos ver las gráficas de autocorrelación de las variables W-ALL (subfigura a), Occupation (subfigura b) y Ext-Temp (subfigura c), en las cuales vemos como la variable de consumo de agua presenta niveles de correlación bajos en todos los lags, lo que concuerda con su menor estacionalidad y mayor irregularidad. Mientras, las variables de ocupación y temperatura exterior muestran unos valores de autocorrelación muy elevados.

Apéndice B

Modelos de predicción

En este apartado se expone toda la información complementaria del capítulo 5

B.1. Experimentación

En este apartado se explican con detalle los experimentos llevados a cabo para encontrar modelos de calidad de XGBoost, MLP, RNN, CNN y Seq2Seq en los problemas de enero, febrero y marzo. Para ello, se ha hecho por cada uno de estos problemas lo siguiente:

1. Por cada tipo de modelo (XGBoost, MLP, RNN, CNN y Seq2Seq) se han entrenado varios modelos con distintos parámetros.
2. Se ha evaluado con la función de error NMAE el rendimiento de cada modelo sobre el conjunto de muestras de validación.
3. Se comparan los distintos modelos de cada tipo en función de su valor de NMAE para así obtener el mejor.

Como la inicialización de los pesos de los modelos de redes neuronales se realiza de forma aleatoria se ha ejecutado varias veces el entrenamiento de cada modelo específico quedándonos con su NMAE sobre el conjunto de validación promedio.

Cabe destacar por un lado que estos pasos se han llevado a cabo tratando de predecir únicamente H-F123-D1, pues es muy parecida a H-F123-D5/2 (como vimos en el apartado 4.3.5) y tienen casi (a diferencia de una) las mismas variables predictoras, por lo que los mejores modelos para predecir H-F123-D1 es muy probable que también sean los mejores para predecir H-F123-D5/2 y de esta forma nos ahorramos la mitad de los experimentos. Por

otro lado, todos los modelos de RNA (MLP, RNN, CNN y seq2seq) se han entrenado con los mismos parámetros de entrenamiento, los cuales han sido estimados con una pequeña fase experimental previa y son los siguientes:

- Entrenamiento por mini lotes de tamaño 72, es decir, 72 muestras o series temporales, muestreándolas de forma aleatoria en cada época.
- Entrenamiento durante 50 épocas, evaluando el rendimiento sobre el conjunto de validación del modelo al final de cada una de ellas.
- Nos quedamos con los pesos del modelo que menor error sobre el conjunto de validación haya tenido a lo largo de las 50 épocas (early stopping).
- Todas las neuronas de las capas ocultas totalmente conectadas y convoluciones emplean la función de activación relu, las neuronas de las capas LSTM emplean tanh (tangente hiperbólica) como función de activación y sigmoid como función de activación recurrente.

Todos estos experimentos son necesarios debido a que no es posible saber teóricamente cuáles son los parámetros óptimos de cada tipo de modelo para resolver cada problema, por lo que tenemos que aplicar un proceso de ensayo y error.

A continuación por cada tipo de modelo vamos a ver más específicamente con qué modelos hemos experimentado y cual ha sido el mejor que hemos logrado obtener para predecir H-F123-D1 en los problemas de enero, febrero y marzo.

B.1.1. Modelo XGBoost

En el caso de XGBoost se ha probado para cada problema el mismo conjunto de modelos, los cuales tienen distintos parámetros de número de árboles, profundidad máxima y tasa de aprendizaje. Más concretamente los distintos valores de estos parámetros con los que hemos probado se pueden ver en la siguiente tabla.

Número de árboles	{50, 100, 500}
Profundidad máxima	{51, 100, 500}
Tasa de aprendizaje	{0.05, 0.001, 0.1}

Tabla B.1: Parámetros posibles de los modelos de XGBoost.

En la tabla B.1 podemos ver los valores que hemos ido probando para cada parámetro a la hora de entrenar modelos de XGBoost. En este caso,

se han obtenido por cada problema tantos modelos como combinaciones posibles existen de dichos parámetros, es decir, 27. Así, el mejor modelo de XGBoost para cada problema lo podemos ver en la siguiente tabla.

Problema	*N. Árboles*	*Prof. Máxima*	*Tasa aprendizaje*
Enero	50	51	0.05
Febrero	100	100	0.001
Marzo	50	51	0.05

Tabla B.2: Mejores modelos de XGBoost para enero, febrero y marzo.

En la tabla B.2 podemos ver en cada fila los parámetros del modelo de XGBoost que mejor rendimiento a demostrado sobre un mes concreto. Asi pues, vemos que los modelos de XGBoost que mejor rendimiento obtienen cuentan con un número de árboles y una profundidad no demasiado altas, pues los mejores modelos de enero y marzo utilizan los valores mínimos de árboles y profundiad posibles vistos en la tabla B.1, mientras que el mejor modelo de febrero utiliza valores intermedios para ambos parámetros. Esta es la razón por la que no probamos a experimentar con modelos con un número de árboles y una profundiad superiores a 500, pues el rendimiento en términos de NMAE sobre el conjunto de validación tendía a empeorar. Esto se debe principalmente a que conforme aumentan estos dos parámetros el modelo de XGBoost es más potente y aumenta el riesgo de sobreaprender el conjunto de entrenamiento, empeorando así la capacidad de generalización del modelo.

B.1.2. Modelo MLP

Para el modelo MLP se han probado las siguientes arquitecturas para predecir H-F123-D1 en los problemas de enero, febrero y marzo.

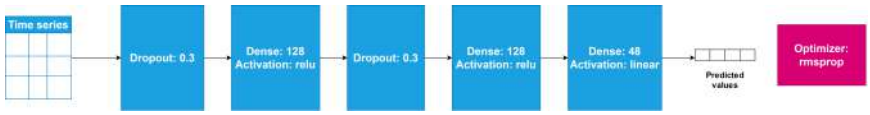
- Seis modelos MLP con una capa oculta totalmente conectada, cada uno con un número concreto de neuronas en dicha capa: 16, 32, 64, 128, 256, 512.
- Dieciocho modelos, resultantes de aplicar a la capa oculta de cada uno de los anteriores 6 modelos tres tasas distintas de dropout: 0.15, 0.3 y 0.5.
- Seis modelos de MLP con dos capas ocultas totalmente conectadas, cada uno con el mismo número de neuronas en ambas: 16, 32, 64, 128 y 512.
- Dieciocho modelos, resultantes de aplicar a ambas capas ocultas de cada uno de los anteriores 6 modelos tres tasas distintas de dropout:

0.15, 0.3 y 0.5. Siempre se le ha aplicado la misma tasa de dropout a ambas capas para evitar una explosión combinatoria en los parámetros y no tener que entrenar un número inviable de modelos.

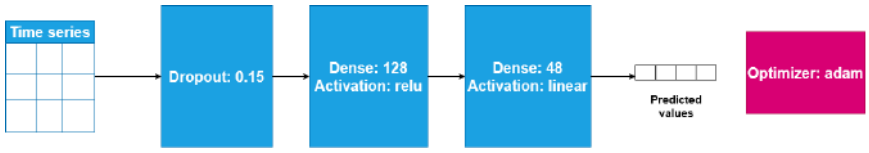
Además, cada uno de estos modelos ha sido entrenado tres veces, cada una con un algoritmo de optimización distinto: gradiente descendente estocástico, rmsprop y adam. Por tanto, si hacemos las cuentas, se han entrenado un total de 144 modelos por cada problema.

Vemos pues que estamos probando modelos con distinto número de capas, neuronas y tasas de dropout. Los valores con los que hemos probado han sido elegidos basándonos en una fase experimental previa, en la cual se ha observado que el rendimiento no mejora ni empleando modelos de más de dos capas ocultas, ni utilizando capas con un número de neuronas superior a 512 y ni aplicando una tasa de dropout a las capas superior a 0.5.

De esta forma, las mejores arquitecturas de MLP para cada problema las podemos ver en la siguiente figura.



(a) Enero.



(b) Febrero.

(c) Marzo.

Figura B.1: Mejores modelos de MLP.

En la figura B.1 podemos ver la mejor arquitectura obtenida para los problemas de enero (subfigura a), febrero (subfigura b) y marzo (subfigura c) en términos de NMAE sobre los valores de validación de H-F123-D1. Vemos que en cada arquitectura mostramos sus capas (rectángulos de color azul) de izquierda a derecha, cada cada una con sus distintos parámetros y junto a cada arquitectura se encuentra el algoritmo de optimización de la misma (rectángulo de color rojo). En este caso podemos observar lo siguiente.

- Tanto en enero como en marzo la mejor arquitectura es aquella formada por dos capas cada una con 128 neuronas. Mientras, que en febrero la mejor está formada por una sola capa de 128 neuronas. De esta forma, las capas que mejor resultados dan son aquellas con un número intermedio de neuronas, lo cual tiene sentido ya que los tres problemas no contienen muchos datos, por lo que no se necesita una gran capacidad de aprendizaje.

- Tanto en enero como en marzo ofrece mejores resultados el algoritmo de optimización Rmsprop, mientras que en febrero es Adam. Vemos por tanto que los algoritmos de optimización que mejores resultados ofrecen son aquellos que emplean una tasa de aprendizaje adaptativa, algo que ya anticipamos en el apartado 2.4.4.2.

B.1.3. Modelo RNN

Los experimentos llevados a cabo con modelos RNN para cada problema han consistido en coger la mejor arquitectura MLP para dicho problema y añadirle como primeras capas ocultas capas LSTM con distintos parámetros. Más específicamente se han probado los siguientes modelos:

- Cuatro modelos con una sola capa LSTM antes de la parte MLP o totalmente conectada de 16, 32, 64 y 128 neuronas.

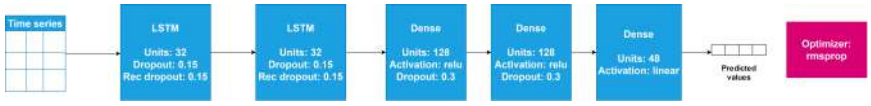
- Doce modelos con una sola capa LSTM antes de la parte MLP o totalmente conectada que resultan de aplicarle a los anteriores cuatro ratios de dropout y dropout recurrente de 0.15, 0.3 y 0.5.

- Cuatro modelos con dos capas LSTM antes de la parte MLP o totalmente conectada de 16, 32, 64 y 128. Empleamos siempre en ambas capas el mismo número de neuronas.

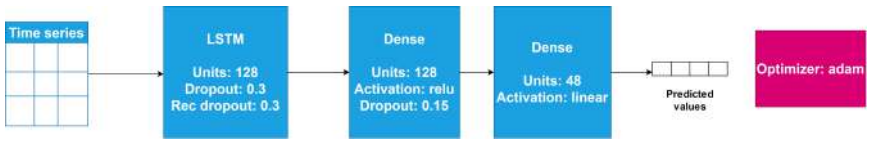
- Doce modelos con dos capas LSTM antes de la parte MLP o totalmente conectada que resultan de aplicarle a los anteriores cuatro ratios de dropout y dropout recurrente de 0.15, 0.3 y 0.5.

Vemos que estamos modificando, al igual que con los modelos MLP, el número de capas y neuronas y la tasa de dropout. En este caso no probamos a entrenar cada modelo con varios algoritmos de optimización distintos ya que utilizamos aquel que ofreció mejores resultados en cada problema con MLP. Por tanto, por cada problema se han entrenado 42 modelos distintos.

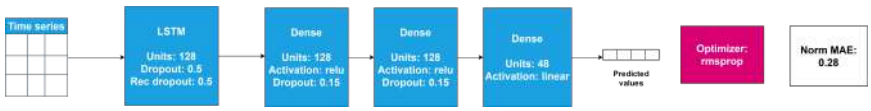
En la siguiente figura podemos ver los mejores modelos de RNN para cada problema resultantes de este proceso de experimentación.



(a) Enero.



(b) Febrero.



(c) Marzo.

Figura B.2: Mejores modelos de RNN.

En la figura B.2 podemos ver el mejor modelo de RNN para el problema de enero (subfigura a), febrero (subfigura b) y marzo (subfigura c) en términos de NMAE sobre los valores de validación de H-F123-D1. En ella podemos ver lo siguiente:

- Mientras que en los problemas de febrero y marzo el mejor modelo solo tiene una capa LSTM con 128 neuronas (units) y con dropout y dropout recurrente elevados, en enero el mejor modelo cuenta con dos capas LSTM pero con un número menor de neuronas (32) y unas tasas de dropout y dropout recurrente menores (0.15), que en potencia de calculo puede ser similar a una sola capa LSTM con 128 neuronas. Así, vemos que para obtener los mejores resultados no se necesita mucha capacidad de aprendizaje, lo cual puede ser debido a que cada problema no contiene demasiados datos.
- Al igual que ocurría con los mejores modelos de MLP, estos también tienen dropout en todas las capas, lo cual confirma la capacidad de generalización y menor tendencia al sobreajuste que proporciona esta técnica.

B.1.4. Modelo CNN

Los experimentos con modelos CNN han consistido en coger el mejor modelo de RNN de cada problema (figura B.2) y añadirle al principio capas convolucionales y de max pooling 1D. De esta manera se han entrenado los siguientes modelos por cada problema.

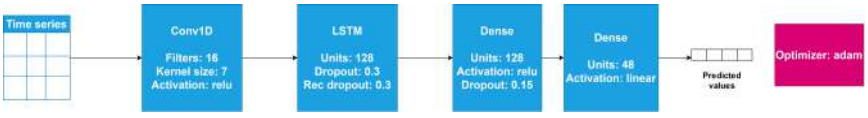
- Nueve modelos con una única capa convolucional 1D antes de la RNN, variando el número de filtros y su tamaño de la siguiente forma:

  - Capas con 16 filtros, con filtros de tamaño 3, 5 y 7.
  - Capas con 32 filtros, con filtros de tamaño 3, 5 y 7.
  - Capas con 64 filtros, con filtros de tamaño 3, 5 y 7.

- Nueve modelos con dos capas convolucionales 1D ambas con el mismo número y tamaño de filtros, variando estos de la misma forma que con los modelos de una capa. Todos estos modelos tendrán, además, una capa de max pooling 1D de tamaño 3 entre las dos capas para reducir la dimensión de las secuencias de salida de la primera capa.

En la siguiente figura podemos ver la arquitectura del mejor modelo de CNN para cada problema resultante de este proceso de experimentación.

(a) Enero.



(b) Febrero.

(c) Marzo.

Figura B.3: Mejores modelos de CNN.

En la figura B.3 podemos ver el mejor modelo de CNN para enero (subfigura a), febrero (subfigura b) y marzo (subfigura c) en términos de NMAE sobre los valores de validación de H-F123-D1. En ella observamos que el mejor modelo para el problema de enero tiene dos capas convolucionales 1D con 32 filtros de tamaño 5 cada una. Mientras que en febrero y marzo el mejor modelo tiene una sola capa convolucional 1D con 16 filtros de tamaño 5 y 7 respectivamente. Pasa entonces lo mismo que con los modelos de RNN, es decir, para el problema de enero se requiere un mayor número de capas.

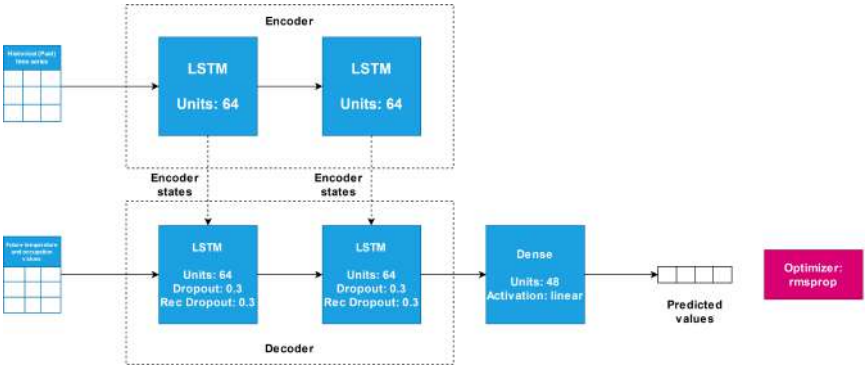
B.1.5. Modelo Seq2Seq

El modelo o arquitectura Seq2Seq es la última con la que hemos experimentado para tratar de predecir H-F123-D1 en enero, febrero y marzo. Como vimos en la sección 2.4.3.4, estas arquitecturas tienen la particularidad de que no solo reciben como entrada valores pasados, sino también valores futuros o contextuales de los n siguientes instantes de tiempo en los que vamos a predecir los valores de la variable objetivo. En nuestro caso, estos valores contextuales serán de temperatura (predicción meteorológica) externa del edificio ICPE y la ocupación (estimación), los cuales nos aportarán aún más información de cara a predecir los valores de la variable objetivo (H-F123-D1 en estos experimentos). Hemos probado para cada problema los siguientes modelos.

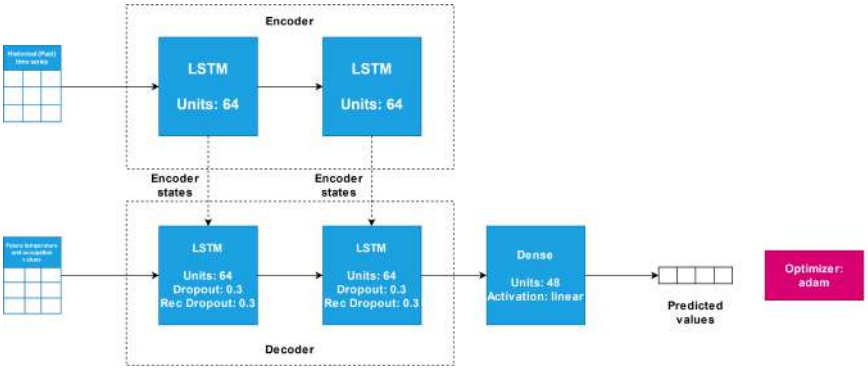
- Cuatro modelos con un codificador y decodificador de una sola capa LSTM con un número de neuronas variable entre los siguientes valores: 16, 32, 64 y 128. En todo momento la capa LSTM del codificador y la del decodificador tienen el mismo número de neuronas.
- Cuatro modelos con un codificador y decodificador de dos capas LSTM con un número de neuronas también variable entre 16, 32, 64 y 128. En todo momento las dos capas LSTM de codificador y del decodificador tienen el mismo número de neuronas.

En todas las capas LSTM del decodificador de todos los modelos se ha aplicado unas tasas de dropout y dropout recurrente del 0.3 para evitar el sobreajuste y mejorar la capacidad de generalización. Al codificador no se le aplica dropout debido a que es el encargado aprender cómo se codifica cada serie temporal de entrada en un vector de tamaño fijo, por lo que aplicar dropout en sus capas LSTM sería añadir ruido a dicho aprendizaje empeorando el rendimiento de los modelos.

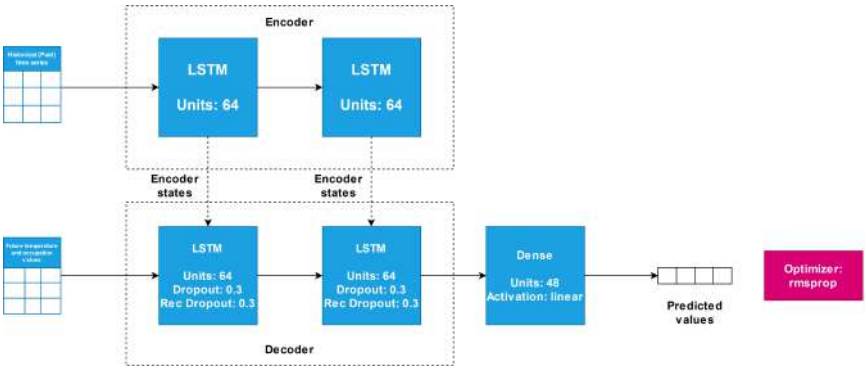
En la siguiente figura podemos ver las mejores arquitecturas Seq2Seq obtenidas para cada problema como consecuencia de este proceso de experimentación.



(a) Enero.



(b) Febrero.



(c) Marzo.

Figura B.4: Mejores modelos de Seq2Seq.

En la figura B.4 podemos ver la mejor arquitectura de Seq2Seq para el problema de enero (subfigura a), febrero (subfigura b) y marzo (subfigura c) en términos de NMAE a la hora de predecir los valores de validación de H-F123-D1. En ella vemos lo siguiente:

- La mejor arquitectura Seq2Seq para los tres problemas es la misma, es decir, una arquitectura formada por dos capas LSTM en el codificador (encoder) y en el decodificador (decoder), todas ellas de 64 neuronas.
- Aunque para todos los problemas la mejor arquitectura haya resultado ser la misma, los algoritmos de optimización no son los mismos, pues para los problemas de enero y marzo continuamos utilizando rmsprop y para febrero adam.

B.2. Rendimiento mejores modelos

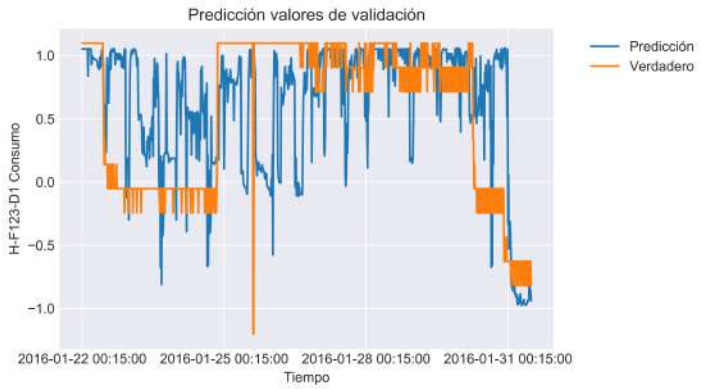
En este apartado se analiza de forma detallada el rendimiento de los mejores modelos de XGBoost, MLP, RNN, CNN y Seq2Seq a la hora de predecir H-F123-D1 y H-F123-D5/2 en los problemas de enero, febrero y marzo.

B.2.1. Problema de enero

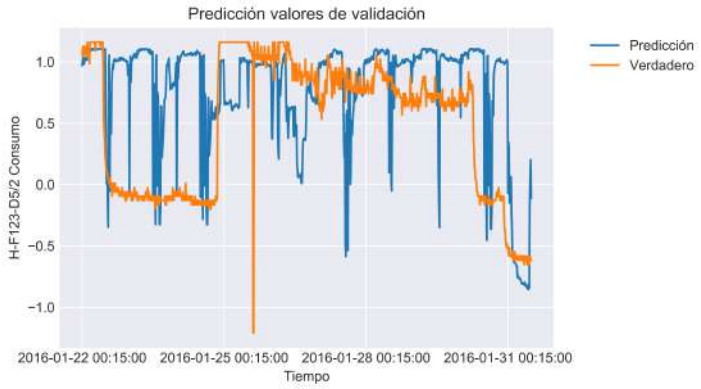
A continuación evaluaremos el rendimiento de los mejores modelos XGBoost, MLP, RNN, CNN y Seq2Seq para predecir en enero las variables H-F123-D1 y H-F123-D5/2.

B.2.1.1. XGBoost

El mejor modelo de XGBoost para el problema de Enero (apéndice B.1.1, tabla B.2 (fila 1)) obtiene un NMAE de 0.48 a la hora de predecir H-F123-D1 y de 0.56 para predecir H-F123-D5/2. Para hacernos una mejor idea de qué significan estos valores vamos a ver gráficamente la predicción que realiza este modelo de XGBoost de los valores de validación de H-F123-D1 y H-F123-D5/2.



(a) H-F123-D1



(b) H-F123-D5/2.

Figura B.5: Predicción de los valores de validación en enero mediante XGBoost.

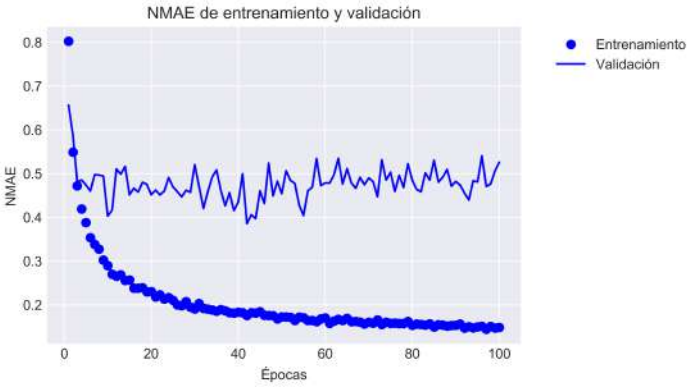
En la figura B.5 vemos dos gráficas con los valores de validación de H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) verdaderos (naranja) junto a los predichos (azul) (eje y) por el mejor modelo de XGBoost del problema de enero a lo largo del tiempo (eje x), pudiendo observar lo siguiente.

- Los valores de validación de H-F123-D1 y H-F123-D5/2 van desde el 2016-01-22 00:15:00 hasta el 2016-01-31 23:45:00, que son los instantes de tiempo reservados para validación en enero. Los valores verdaderos de validación de H-F123-D1 y H-F123-D5/2 son muy similares, con las mismas subidas y bajadas, con la diferencia de que los valores de H-F123-D5/2 parecen un poco más suaves.
- Comparando en ambas subfiguras los valores predichos con los verdaderos vemos que el modelo XGBoost no es capaz de predecir los patrones de consumo de H-F123-D1 y H-F123-D5/2, por lo que interpretamos que los valores de NMAE obtenidos por este modelo son

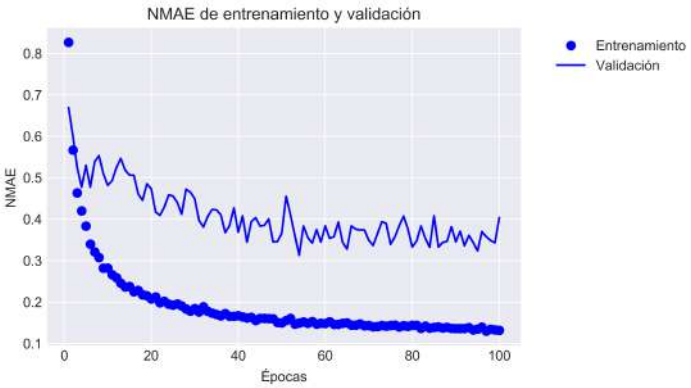
elevados. Esto se debe a que el modelo XGBoost está en realidad formado por 48 modelos cada uno entrenado de forma independiente para predecir el valor de la variable objetivo en un instante de tiempo futuro concreto a partir de una serie temporal de entrada con 384 observaciones, por lo que no es capaz de establecer relaciones de dependencia entre los valores predichos. Además, cada uno de los 48 árboles procesan la serie temporal de entrada de una sola vez, por lo que no tienen en cuenta las dependencias temporales entre los distintas observaciones de la misma.

B.2.1.2. MLP

El mejor modelo de MLP para el problema de enero (apéndice B.1.2, figura B.1 (a)) registra los siguientes valores de NMAE sobre el conjunto de entrenamiento y validación.



(a) H-F123-D1

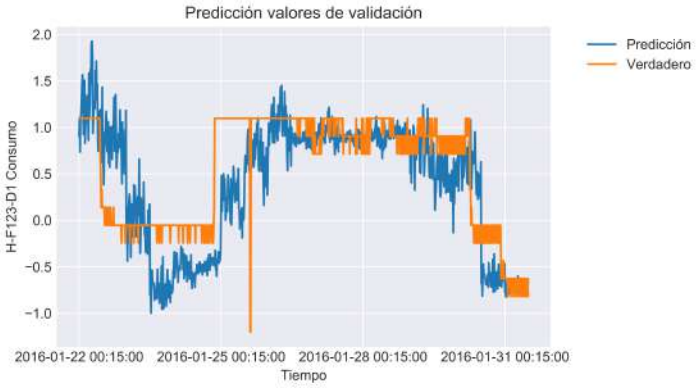


(b) H-F123-D5/2.

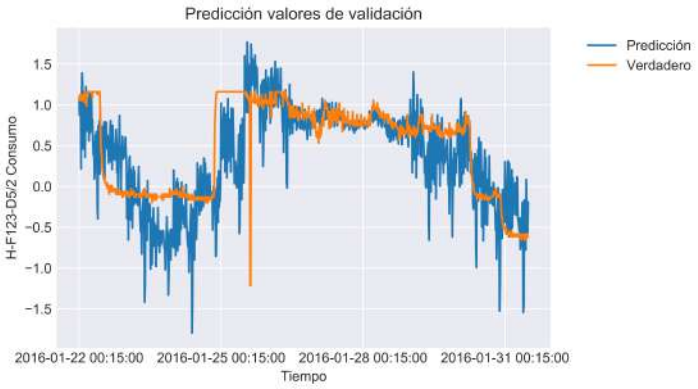
Figura B.6: Evolución del NMAE de entrenamiento y validación del mejor modelo MLP para el problema de enero

En la figura B.6 vemos dos gráficas con la evolución del NMAE de entrenamiento (linea con puntos) y validación (línea continua) a lo largo de 100 épocas (eje x) de entrenamiento para aprender a predecir las variables H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b). En ellas vemos que el NMAE sobre el conjunto de entrenamiento va descendiendo paulatinamente a lo largo de las épocas, lo cual tiene sentido, pues indica que el modelo va poco a poco aprendiendo a predecir las muestras de entrenamiento. En cambio, el NMAE sobre el conjunto de validación desciende en el caso de H-F123-D1 hasta llegar a 0.38 en la época 41 y a partir de ahí va ascendiendo ligeramente. Esto se debe a que el modelo MLP sobreaprende a partir de dicha época el conjunto de entrenamiento. Por el contrario, el NMAE sobre el conjunto de validación de H-F123-D5/2 desciende hasta llegar en la época 56 a 0.31, para a continuación seguir una tendencia más o menos constante el resto de épocas

Tomando pues los dos modelos de MLP con menor NMAE vamos a ver como predicen visualmente los valores de validación de H-F123-D1 y H-F123-D5/2.



(a) H-F123-D1



(b) H-F123-D5/2.

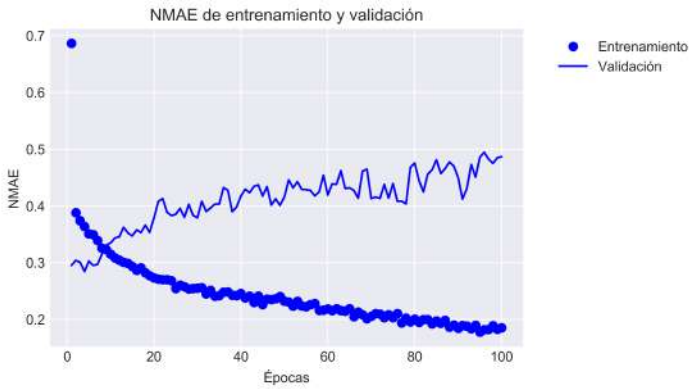
Figura B.7: Predicción de los valores de validación en enero mediante MLP.

En la figura B.7 vemos dos gráficas con los valores de validación verdaderos (naranja) junto a los predichos (azul) (eje y) de H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) a lo largo del tiempo (eje x), pudiendo observar lo siguiente.

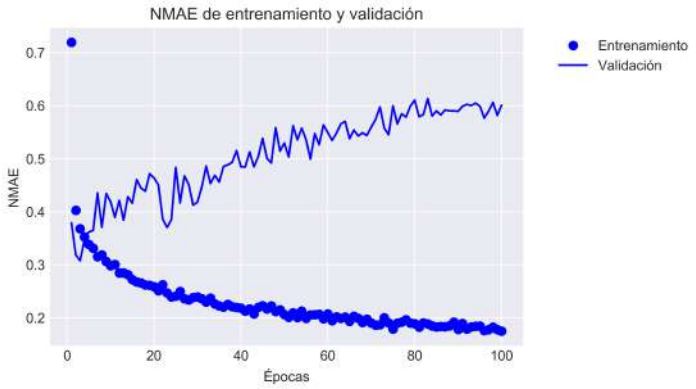
- El modelo MLP sí es capaz de predecir de forma aproximada los patrones (subidas y bajadas) de consumo de H-F123-D1 y H-F123-D5/2 a diferencia del modelo de XGBoost. Esto es así, porque el modelo MLP sí es capaz de tener en cuenta dependencias entre los datos de salida, pues sus capas están totalmente conectadas.
- Aunque es capaz de predecir las subidas y bajadas de consumo de forma general, este modelo tiene un MAE demasiado grande, el cual principalmente se debe a las bruscas oscilaciones locales de los valores que predice, cuando en realidad los valores verdaderos son más suaves.

B.2.1.3. RNN

Ahora vamos a ver el rendimiento del mejor modelo RNN (apéndice B.1.3, figura B.2 (a)) para el problema de enero a la hora de predecir H-F123-D1 y H-F123-D5/2.



(a) H-F123-D1



(b) H-F123-D5/2.

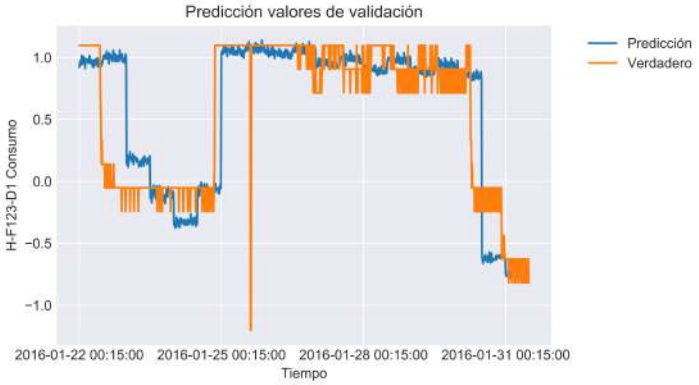
Figura B.8: Evolución del NMAE de entrenamiento y validación del mejor modelo de RNN para el problema de enero

En la figura B.8 se observa que durante el entrenamiento del mejor modelo RNN del problema da enero para aprender a predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) el NMAE de entrenamiento disminuye poco a poco a medida que transcurren las épocas de entrenamiento, mientras que el NMAE de validación disminuye durante las primeras épocas hasta alcanzar su valor mínimo para entonces empezar a subir drásticamente,lo cual es debido al sobreaprendizaje del conjunto de entrenamiento. Este sobreaprendizaje es más acentuado (a pesar de utilizar dropout) que con el modelo MLP por los siguientes motivos:

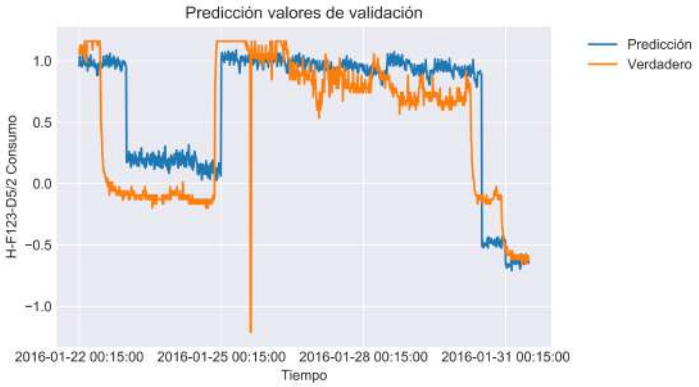
- Las redes neuronales recurrentes son muy eficaces a la hora de aprender patrones y dependencias temporales en los datos, lo que hace que aprendan muy rápido los patrones importantes del conjunto de entrenamiento para predecir el conjunto de validación.
- En un mes tenemos muy pocas muestras de entrenamiento y aún menos en enero, pues nos faltan los 5 primeros días.

- Los valores de entrenamiento difieren de los de validación, lo que hace que haya muy pocos patrones útiles por aprender en el conjunto de entrenamiento que nos ayuden a predecir el conjunto de validación.

Pero aún a pesar de este sobreaprendizaje, el modelo RNN consigue unos NMAE de validación sobre H-F123-D1 y H-F123-D5/2 menores que los conseguidos con XGBoost y MLP. Debido principalmente a que procesa cada serie temporal de entrada teniendo en cuenta el orden cronológico de sus instantes de tiempo y las dependencias temporales entre los mismos. Esto lo podemos ver mejor en la siguiente figura.



(a) H-F123-D1



(b) H-F123-D5/2.

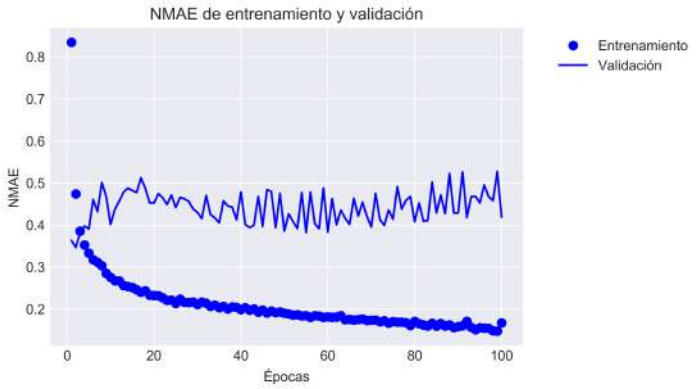
Figura B.9: Predicción de los valores de validación en enero mediante RNN.

Vemos en la figura B.9 como efectivamente el modelo RNN predice mucho mejor los patrones de consumo de H-F123-D1 (subfigura a) y de H-F123-D5/2 (subfigura b) y con unas oscilaciones locales más suaves. Aunque eso sí, vemos un desfase pequeño entre los valores predichos y los verdaderos, pareciendo predecir las bajadas y las subidas de consumo un poco más tarde. Esto se puede deber principalmente a la diferencia en las observaciones

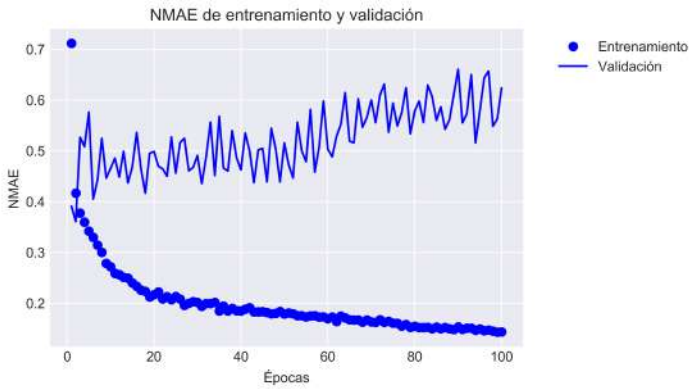
de entrenamiento y de validación, pues las subidas y bajadas de consumo de calefacción en los primeros 21 días de enero tienen lugar en momentos ligeramente distintos en comparación con las subidas y bajadas en los últimos 10 días.

B.2.1.4. CNN

Vamos ahora a ver qué ocurre al añadir al modelo RNN capas convolucionales, es decir, con el modelo CNN. En la siguiente figura podemos ver la evolución del NMAE sobre los conjuntos de entrenamiento y validación del modelo CNN.



(a) H-F123-D1



(b) H-F123-D5/2.

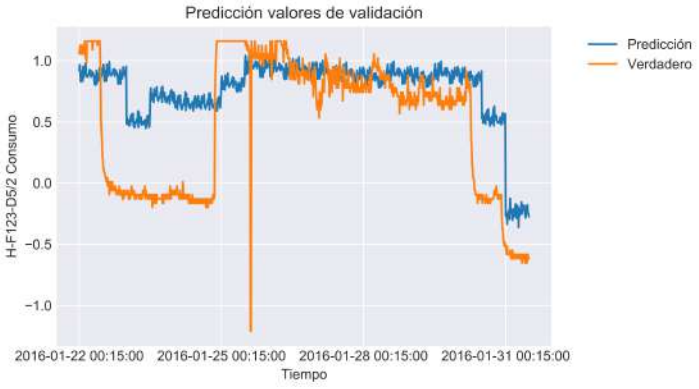
Figura B.10: Evolución del NMAE de entrenamiento y validación del mejor modelo de CNN para el problema de enero

En la figura B.10 vemos dos gráficas con la evolución del NMAE de entrenamiento (linea con puntos) y validación (línea continua) a lo largo de 100 épocas (eje x) para las variables H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b). Vemos en ambas subfiguras que el NMAE de entrenamiento

desciende lentamente a lo largo de las 50 épocas, mientras que el NMAE de validación oscila entre 0.4 y 0.5 sin llegar a sobreajustar pero sin converger en el caso de H-F123-D1 y desciende durante las primeras épocas para luego ascender en el caso de H-F123-D5/2 debido al sobreajuste. El mínimo NMAE que logra este modelo para predecir los valores de validación de H-F123-D1 y H-F123-D5/2 es 0.34 y 0.36 respectivamente que son mayores valores que los obtenidos con el modelo RNN, por lo que parece que las características temporales extraídas por las capas CNN no ayudan tanto a la predicción en este problema como las características que son capaces de extraer las capas LSTM.

Dicho esto, vamos a ver gráficamente los valores predichos de validación de H-F123-D1 y H-F123-D5/2 por el modelo CNN frente a los verdaderos.

(a) H-F123-D1



(b) H-F123-D5/2.

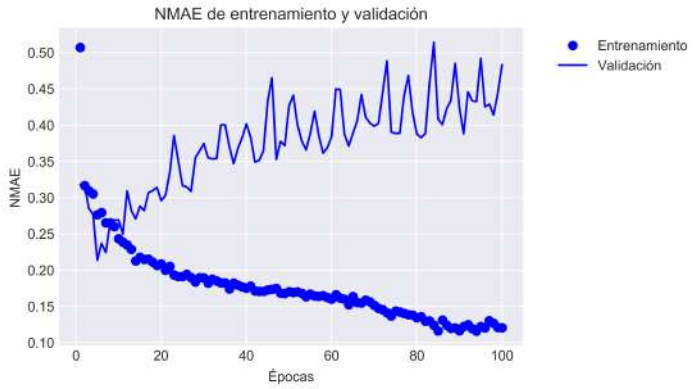
Figura B.11: Predicción de los valores de validación en enero mediante CNN.

Vemos en la figura B.11 como la predicción de los valores de validación de H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) del modelo CNN es peor que la del modelo RNN. En el caso de H-F123-D1 vemos que más o menos el modelo CNN si es capaz de predecir las bajadas y subidas de

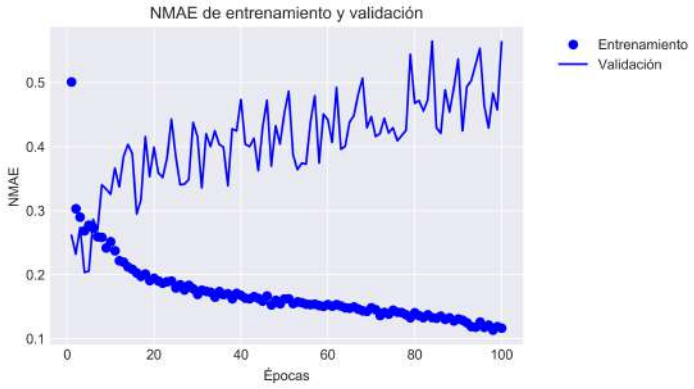
consumo de calefacción aunque vemos que a la hora de predecir el consumo de los días 21, 22, 23 y 24 de Enero comete bastante error. Mientras, en el caso de H-F123-D5/2 el modelo CNN también es capaz de predecir las subidas y bajadas en el consumo de calefacción pero con un mayor error. Así, podemos concluir que en el problema de Enero no ayuda la extracción automática de características temporales mediante capas convolucionales 1D a mejorar el rendimiento del modelo RNN.

B.2.1.5. Seq2Seq

Finalmente vamos a ver si para el problema de enero ayuda la utilización del modelo Seq2Seq (apéndice B.1.5, figura B.4 (a)). En la siguiente figura podemos ver como ha sido el entrenamiento del modelo Seq2seq para aprender a predecir H-F123-D1 y H-F123-D5/2.



(a) H-F123-D1



(b) H-F123-D5/2.

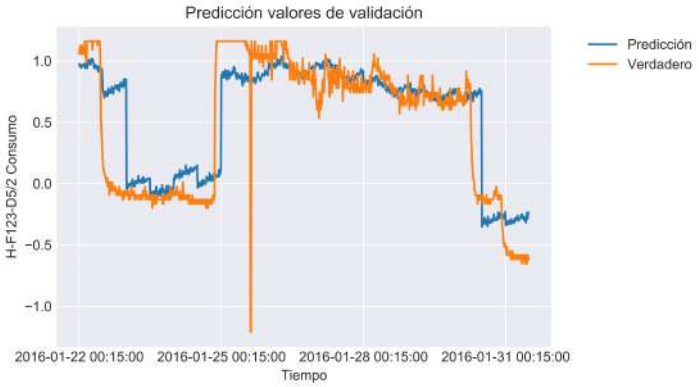
Figura B.12: Evolución del NMAE de entrenamiento y validación del mejor modelo de Seq2Seq para el problema de enero

En la figura B.12 vemos que durante el entrenamiento para aprender a

predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) ocurre algo muy similar con la evolución del NMAE de validación a lo que sucedía con el modelo RNN, es decir, el modelo Seq2Seq aprende muy rápido a predecir ambas variables, llegando a alcanzar el mínimo error en las primeras épocas y a partir de ahí sobreaprende el conjunto de entrenamiento y el NMAE de validación comienza a subir. Los motivos de esto son los mismos que los ya comentados para el modelo RNN. Eso sí, el modelo Seq2Seq alcanza un NMAE de validación de 0.21 para predecir H-F123-D1 y de 0.20 para predecir H-F123-D5/2, lo que mejora bastante el rendimiento del modelo RNN, por lo que al menos para el problema de enero los datos futuros de temperatura y ocupación ayudan a la predicción.

Vamos a ahora a ver visualmente como predice el modelo Seq2Seq los valores de validación de H-F123-D1 y H-F123-D5/2.

(a) H-F123-D1



(b) H-F123-D5/2.

Figura B.13: Predicción de los valores de validación en Enero mediante Seq2Seq.

En la figura B.13 podemos confirmar la mejora de rendimiento del modelo Seq2Seq con respecto al modelo RNN y los demás a la hora de predecir

H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b). Pues vemos que predice bastante bien las subidas y bajadas en el consumo de calefacción, aunque siguen habiendo algunas diferencias entre los valores predichos y verdaderos, sobre todo en los primeros y últimos días de validación. Esto se debe a lo ya comentado anteriormente con el modelo RNN, es decir, la diferencia entre los valores con los que entrenamos y con los que validamos.

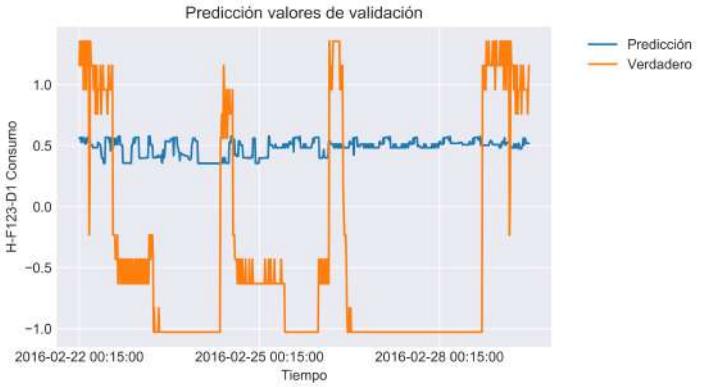
B.2.2. Problema de febrero

Tras evaluar el rendimiento de los mejores modelos de XGBoost, MLP, RNN, CNN y Seq2Seq del problema de enero, vamos a hacer lo propio sobre el problema de febrero.

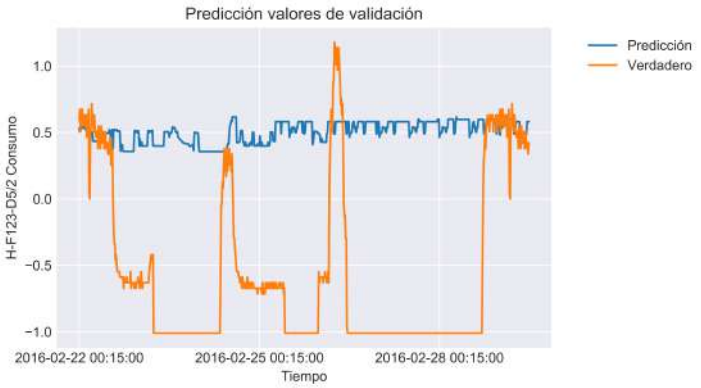
B.2.2.1. XGBoost

El mejor modelo de XGBoost (apéndice B.1.1, tabla B.2, (fila 2)) para el problema de Febrero obtiene un NMAE sobre el conjunto de validación a la hora de predecir H-F123-D1 y H-F123-D5/2 de 1.17 y de 1.10 respectivamente. Vemos por tanto, que es un NMAE bastante más elevado que el logrado por el modelo de XGboost en el problema de enero, lo que nos habla ya de la mayor dificultad de este problema.

Dicho esto, vamos a ver visualmente la predicción que realiza este modelo de los valores de validación de H-F123-D1 y H-F123-D5/2.



(a) H-F123-D1



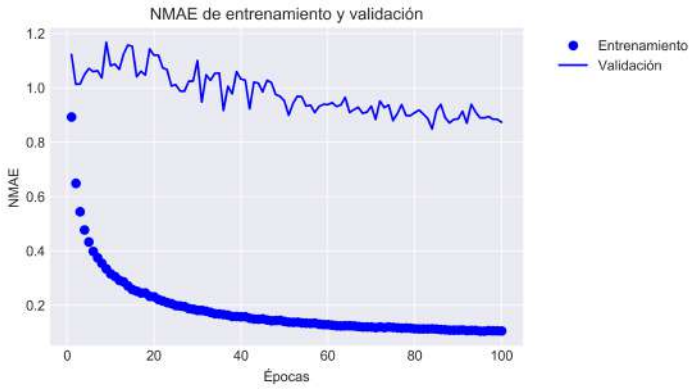
(b) H-F123-D5/2.

Figura B.14: Predicción de los valores de validación en febrero mediante XGBoost.

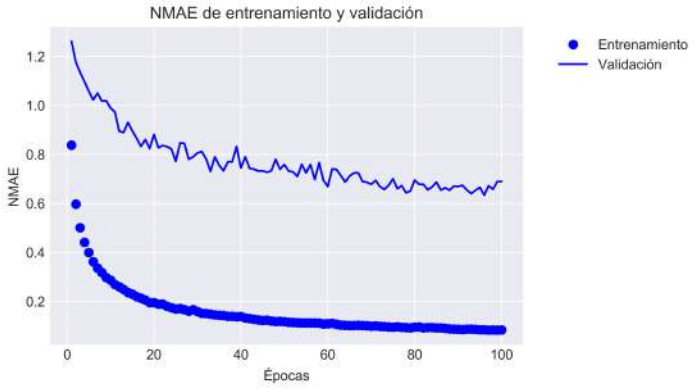
En la figura B.5 vemos dos gráficas con los valores de validación verdaderos (naranja) junto a los predichos (azul) por el mejor modelo de XGBoost del problema de Febrero (eje y) a lo largo del tiempo (eje x) para H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b). Estos valores de validación van desde el 2016-02-22 00:15:00 hasta el 2016-02-29 23:45:00 que son los instantes de tiempo de validación del problema del mes de febrero. En ambas gráficas vemos el motivo del elevado NMAE de validación registrado por el modelo XGBoost al predecir tanto H-F123-D1 y H-F123-D5/2, pues este modelo predice para ambas variables valores casi constantes siendo incapaz de predecir las subidas y bajadas de los valores verdaderos. Aquí se aprecia con más intensidad que en el problema de enero las carencias de este modelo XGBoost para la tarea de predicción de series temporales.

B.2.2.2. MLP

A continuación vamos a ver el rendimiento del mejor modelo de MLP (apéndice B.1.2, figura B.1 (b)).



(a) H-F123-D1



(b) H-F123-D5/2.

Figura B.15: Evolución del NMAE de entrenamiento y validación del mejor modelo de MLP para el problema de febrero

En la figura B.15 se observa que durante el entrenamiento del mejor modelo MLP del problema da febrero para aprender a predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) el NMAE de entrenamiento disminuye poco a poco a medida que transcurren las épocas de entrenamiento, mientras que el NMAE de validación disminuye poco a poco hasta la época 60 para después oscilar manteniendo una tendencia constante. Vemos que el NMAE de validación mínimo alcanzado tanto para H-F123-D1 y H-F123-D5/2 es de 0.82 y 0.61 que son menores que los conseguidos con XGBoost, pero siguen siendo demasiado elevados.

Vamos ahora a ver cómo predice visualmente los valores de validación el modelo MLP.

(a) H-F123-D1

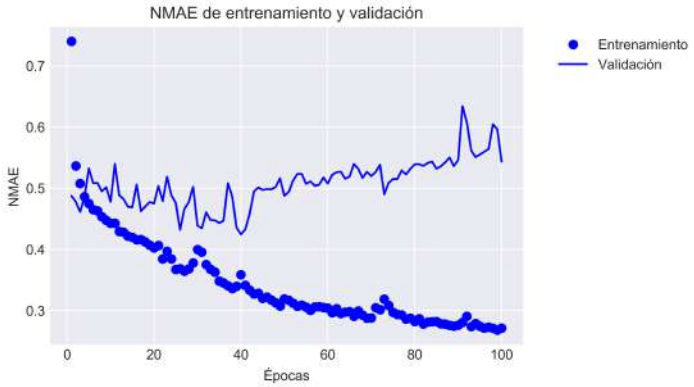
(b) H-F123-D5/2.

Figura B.16: Predicción de los valores de validación en febrero mediante MLP.

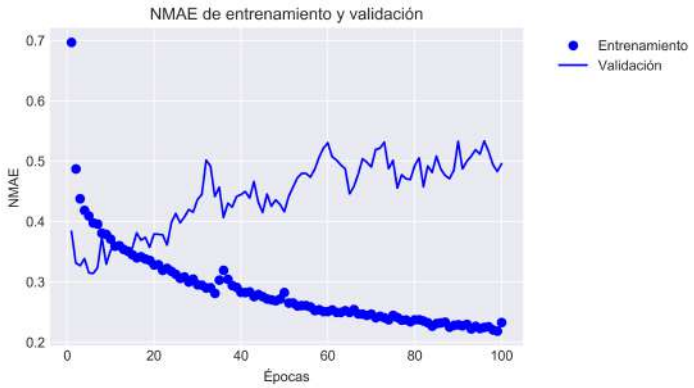
En la figura B.16 se observa que el modelo MLP, al igual que el modelo XGBoost, no es capaz de predecir los patrones de consumo de los valores de validación de H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b), pues los valores de validación predichos presentan grandes oscilaciones (subidas y bajadas) que no coinciden con las de los valores de validación reales. Esto muestra otra vez, la mayor dificultad de este problema.

B.2.2.3. RNN

Vamos ahora a comprobar el rendimiento del mejor modelo RNN (apéndice B.1.3, figura B.2 (b)) para el problema de febrero.



(a) H-F123-D1

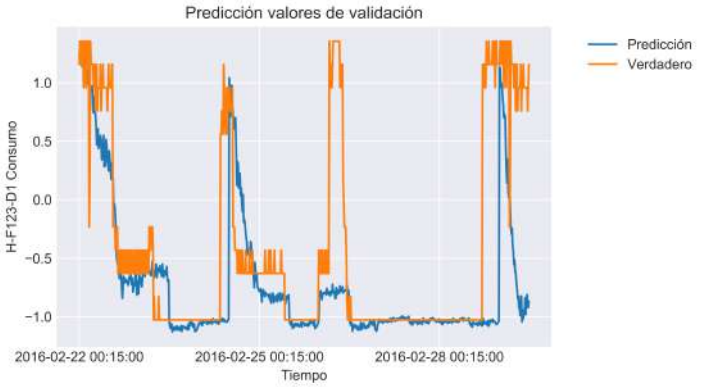


(b) H-F123-D5/2.

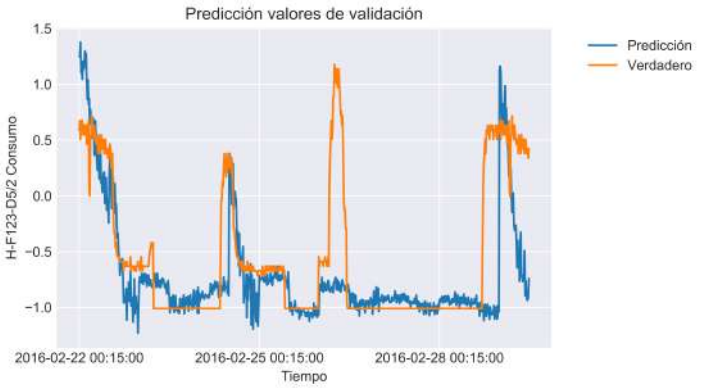
Figura B.17: Evolución del NMAE de entrenamiento y validación del mejor modelo RNN para el problema de febrero

En la figura B.17 se observa que durante el entrenamiento del mejor modelo RNN del problema de febrero para aprender a predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) el NMAE de entrenamiento disminuye poco a poco a medida que transcurren las épocas, mientras que el NMAE de validación disminuye solo en las primeras épocas para luego aumentar, por lo que el modelo RNN sobreajusta el conjunto de entrenamiento. Vemos como el mínimo valor de NMAE de validación que alcanza el modelo RNN es de 0.43 y de 0.31 para H-F123-D1 y H-F123-D5/2 respectivamente por lo que este modelo mejora considerablemente el rendimiento de los modelos de XGBoost y MLP, al igual sucedía con el problema de enero.

Vamos ahora a ver cómo predice visualmente los valores de validación el modelo RNN.



(a) H-F123-D1



(b) H-F123-D5/2.

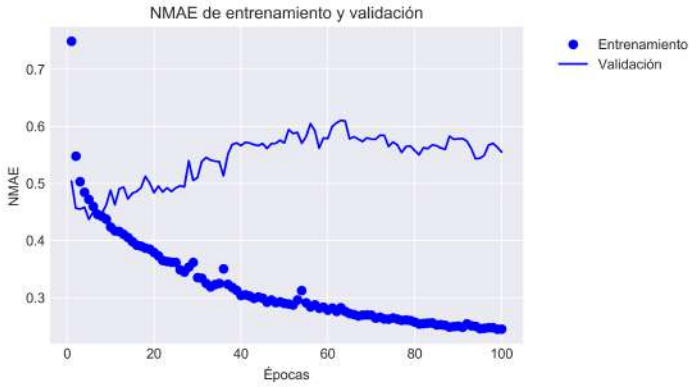
Figura B.18: Predicción de los valores de validación en febrero mediante RNN.

En la figura B.18 podemos ver como, efectivamente, el mejor modelo de RNN para el problema de febrero es bastante mejor que los modelos de XGBoost y MLP para predecir los valores de validación de H-F123-D1 (subfigura a) y H-F123-D52 (subfigura b), pues para ambas variables se observa que este modelo es capaz de predecir aproximadamente todas las subidas y bajadas de sus valores verdaderos. Sin embargo, vemos que en torno al día 26 de febrero hay en ambas variables una subida y bajada que el modelo no es capaz de predecir correctamente, la cual si recordamos es originada por nuestro método de imputación (sección 4.3.4) con el que imputamos todos los valores perdidos de los días 26, 27, 28 y 29 de febrero. De acuerdo con este método de imputación, las observaciones imputados del día 26 de Febrero son iguales a las del mismo día (Viernes) de la semana pasada, es decir, el día 19 de febrero, el cual es un día con cuyas observaciones entrenamos, por lo que se podría pensar que si entrenamos para predecir los valores de H-F123-D1 y H-F123-D5/2 de ese día, los cuales son idénticos a los del día 26, el modelo tendría que saber predecir los valores de ambas variables

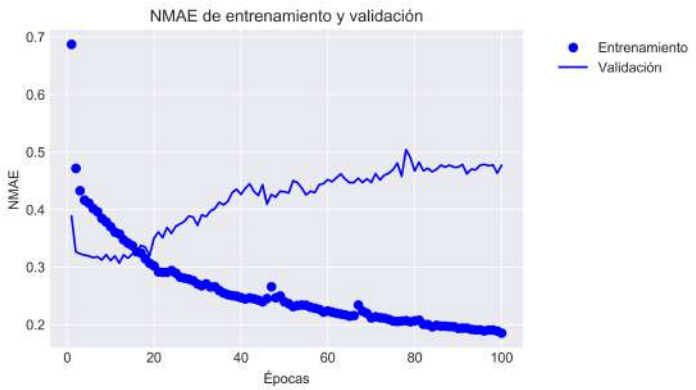
objetivo de este último a la perfección. Pero esto no es así, debido a que en el conjunto de entrenamiento del problema de Febrero las observaciones que preceden al día 19 y a partir de los cuales predecimos los valores de H-F123-D1 y H-f123-D5/2 en dicho día son distintas a las que preceden al día 26. Por lo que el modelo RNN no ha aprendido a predecir la subida y bajada del día 26 de febrero de los valores de H-F123-D1 y H-F123-D5/2 a partir de las observaciones anteriores a dicho día, ya que estos no se encuentran en el conjunto de entrenamiento. Esto vemos que contribuye bastante a aumentar el NMAE de validación para H-F123-D1 y H-F123-D5/2 y puede ser una de las causas de la mayor dificultad de este problema.

B.2.2.4. CNN

A continuación vamos con el mejor modelo CNN (apéndice B.1.4, figura B.3 (b)) para el problema de febrero.



(a) H-F123-D1



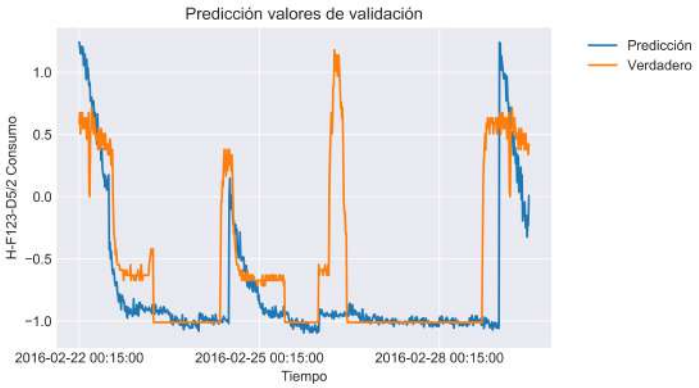
(b) H-F123-D5/2.

Figura B.19: Evolución del NMAE de entrenamiento y validación del mejor modelo de CNN para el problema de febrero

En la figura B.19 se observa que durante el entrenamiento del mejor modelo CNN del problema de febrero para aprender a predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) el NMAE de entrenamiento disminuye poco a poco a medida que transcurren las épocas, mientras que el NMAE de validación disminuye durante las primeras épocas hasta alcanzar su valor mínimo, para entonces empezar a subir lentamente. Esto se debe al sobreaprendizaje, el cual tiene lugar incluso aunque hayamos empleado la técnica de dropout tanto en las capas LSTM como en las capas totalmente conectadas del modelo CNN. Aun así, gracias a la utilización de dropout este sobreaprendizaje es leve. Vemos que el mínimo NMAE de validación alcanzado por el modelo CNN es de 0.43 y 0.30 a la hora de predecir H-F123-D1 y H-F123-D5/2 respectivamente. Por tanto, este modelo obtiene un rendimiento muy similar al modelo RNN.

Vamos a continuación a ver cómo el modelo CNN con estos valores mínimos de NMAE de validación predice visualmente los valores de validación de H-F123-D1 y H-F123-D5/2.

(a) H-F123-D1



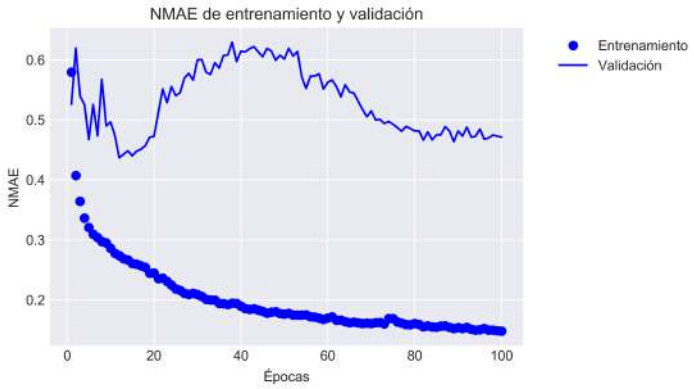
(b) H-F123-D5/2.

Figura B.20: Predicción de los valores de validación en febrero mediante CNN.

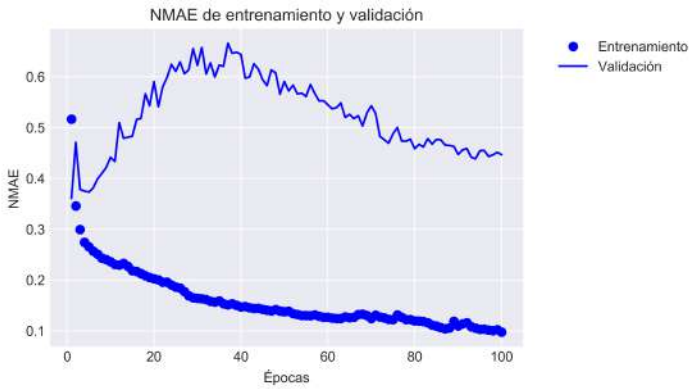
Contemplando la figura B.20 se confirma que el rendimiento del mejor modelo CNN para el problema de febrero a la hora de predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) es similar al proporcionado por el mejor modelo de RNN para febrero, pues el modelo CNN es capaz de predecir aproximadamente los patrones de consumo de ambas variables objetivo pero sigue sin ser capaz de predecir los valores del día 26, lo cual refuerza aun más la hipótesis acerca de nuestro método de imputación formulada en el apartado anterior.

B.2.2.5. Seq2Seq

Finalmente vamos a evaluar el rendimiento del mejor modelo Seq2Seq (apéndice B.1.5, figura B.4 (b)) para el problema de febrero.



(a) H-F123-D1



(b) H-F123-D5/2.

Figura B.21: Evolución del NMAE de entrenamiento y validación del mejor modelo de Seq2Seq para el problema de febrero

En la figura B.21 se observa que durante el entrenamiento del mejor modelo Seq2Seq del problema da febrero para aprender a predecir H-F123-D1

(subfigura a) y H-F123-D5/2 (subfigura b) el NMAE de entrenamiento disminuye poco a poco a medida que transcurren las épocas, mientras que el NMAE de validación disminuye durante las primeras épocas hasta alcanzar su valor mínimo para luego aumentar como consecuencia del sobreaprendizaje del conjunto de entrenamiento hasta la época 40 a partir de la cual empieza a disminuir de nuevo para finalmente mantenerse mas o menos constante durante las últimas 20 épocas. Este último descenso del NMAE de validación tras un ascenso es algo que no hemos visto con ningún otro modelo anterior y puede deberse a que el modelo en primer lugar sobreaprende los patrones de la series temporales pasadas para predecir los valores de H-F123-D1 y H-F123-D5/2 y a continuación empieza a aprender nuevos patrones presentes en las series temporales futuras de ocupación y temperatura, lo que le permite mejorar en la predicción de las muestras de validación. También cabe la posibilidad de que haya pasado esto mismo pero a la inversa.

Podemos ver, que el NME de validación mínimo alcanzado por el modelo Seq2Seq es de 0.43 y 0.36 para predecir H-F123-D1 y H-F123-D5/2 respectivamente, que son valores muy similares a los alcanzados por los modelos RNN y CNN. Dicho esto, vamos a continuación a ver cómo el modelo Seq2eq con estos valores minimos de NMAE predice visualmente los valores de validación de H-F123-D1 y H-F123-D5/2.

(a) H-F123-D1

(b) H-F123-D5/2.

Figura B.22: Predicción de los valores de validación en febrero mediante Seq2Seq.

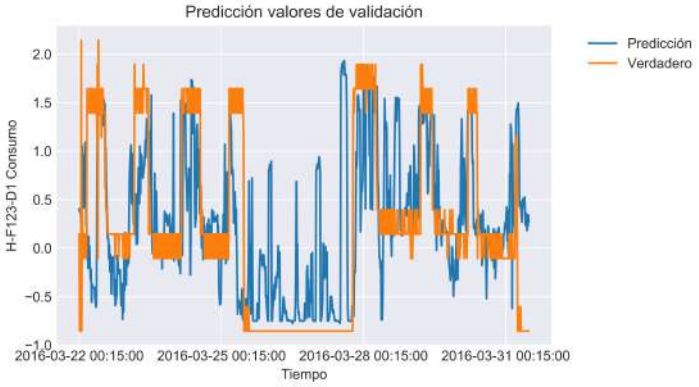
En la figura B.22 vemos que pese a tener un rendimiento en términos de NMAE de validación similar a los modelos de RNN y CNN, el mejor modelo Seq2Seq para el problema de febrero no predice demasiado bien los patrones o subidas y bajadas de consumo tanto para H-F123-D1 (subfigura a) como para H-F123-D5/2 (subfigura b). Pues vemos que predice algunas oscilaciones inexistentes y algunos valores bastante bajos o altos de lo normal. Por tanto, parece que la información contextual o futura sobre temperatura y ocupación no es de demasiada ayuda para el problema de febrero. Aunque eso sí, los valores imputados del día 26 si los predice mejor que los modelos de RNN y CNN.

B.2.3. Problema de marzo

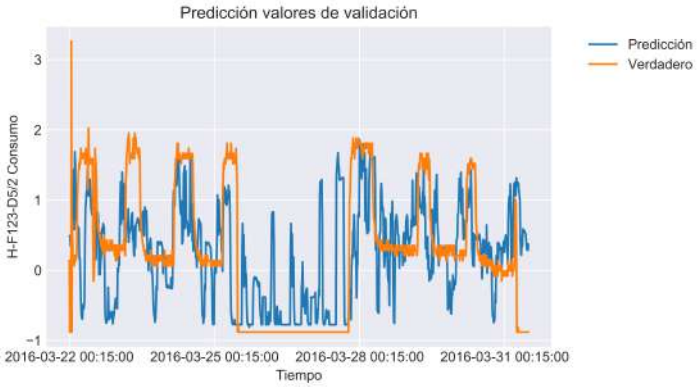
Tras evaluar el rendimiento de los mejores modelos de XGBoost, MLP, RNN, CNN y Seq2Seq en los problemas de enero y febrero vamos a hacer finalmente lo propio con el problema de marzo.

B.2.3.1. XGBoost

El mejor modelo de XGBoost (apéndice B.1.1, tabla B.2 (fila 3)) obtiene un NMAE de validación de 0.56 y 0.60 a la hora de predecir H-F123-D1 y H-F123-D5/2 respectivamente, los cuales parecen valores bastante elevados. Por tanto vamos a ver si de verdad son elevados o no para este problema visualizando para ambas variables objetivo sus valores de validación predichos junto a los verdaderos.



(a) H-F123-D1



(b) H-F123-D5/2.

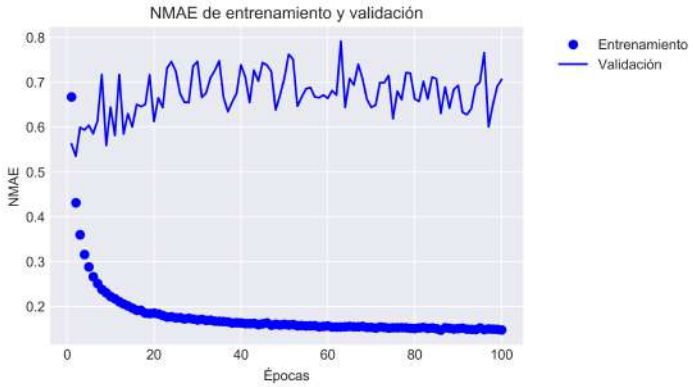
Figura B.23: Predicción de los valores de validación en marzo mediante XGBoost.

En la figura B.23 vemos dos gráficas con los valores de validación verdaderos (naranja) junto a los predichos por el mejor modelo de XGBoost del problema de Marzo (azul) a lo largo del tiempo (eje x) para H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b). Estos valores de validación van desde el instante de tiempo 2016-03-22 00:15:00 hasta el 2016-03-31 23:45:00 los cuales son los instantes de tiempo de validación del problema del mes de marzo. En esta figura podemos comprobar que los valores de NMAE ob-

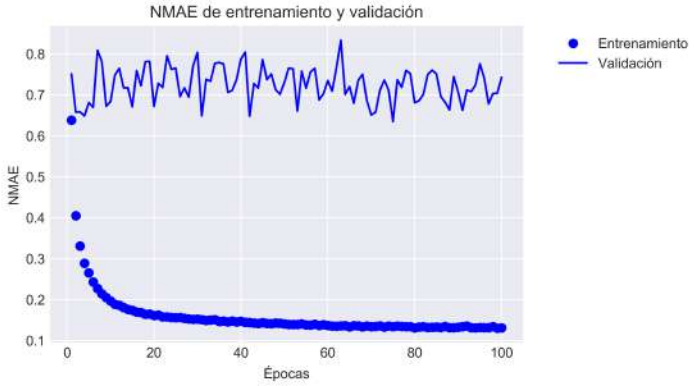
tenidos por el mejor modelo de XGBoost del problema de marzo a la hora de predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) son bastante elevados. Pues vemos que los valores predichos se parecen poco a los verdaderos, tal y como ocurre con los modelos de XGBoost en los problemas de enero y Febrero.

B.2.3.2. MLP

A continuación vamos a ver el rendimiento del mejor modelo MLP (apéndice B.1.2, figura B.1 (c)) para el problema de marzo.



(a) H-F123-D1



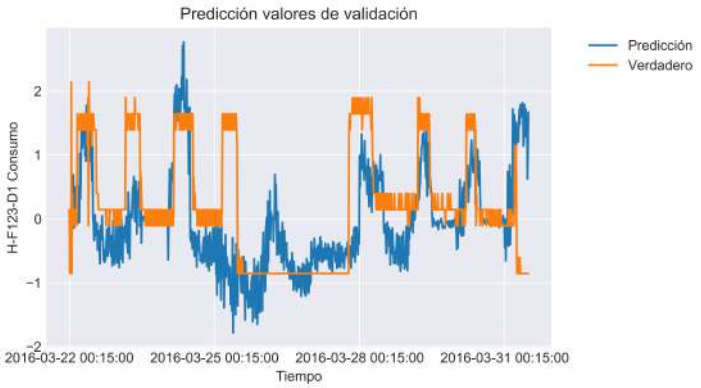
(b) H-F123-D5/2.

Figura B.24: Evolución del NMAE de entrenamiento y validación del mejor modelo de MLP para el problema de marzo

En la figura B.24 se observa que durante el entrenamiento del mejor modelo MLP del problema da marzo para aprender a predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) el NMAE de entrenamiento (línea) disminuye poco a poco a medida que transcurren las épocas de entrenamiento, mientras que el NMAE de validación (línea con puntos) oscila con-

tinuamente, sin llegar realmente a disminuir, entre 0.5 y 0.8 en el caso de H-F123-D1 y entre 0.6 y 0.8 en el caso de H-F123-D5/2. En estas oscilaciones el modelo MLP alcanza un NMAE de validación mínimo de 0.53 para H-F123-D1 y de 0.64 para H-F123-D5/2, que son bastante similares a los obtenidos con XGBoost.

Veamos ahora visualmente el rendimiento de los modelos de MLP que obtienen estos NMAE de validación.



(a) H-F123-D1

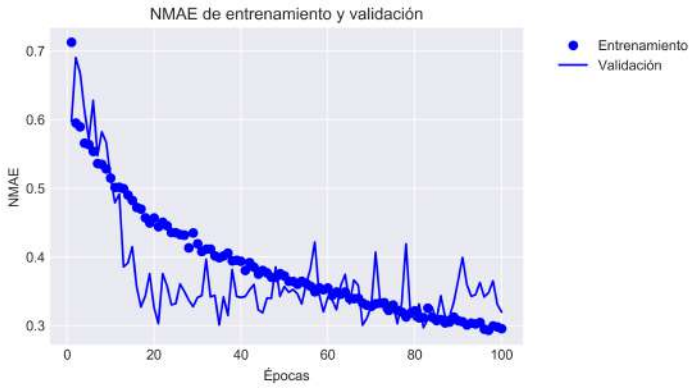
(b) H-F123-D5/2.

Figura B.25: Predicción de los valores de validación en marzo mediante MLP.

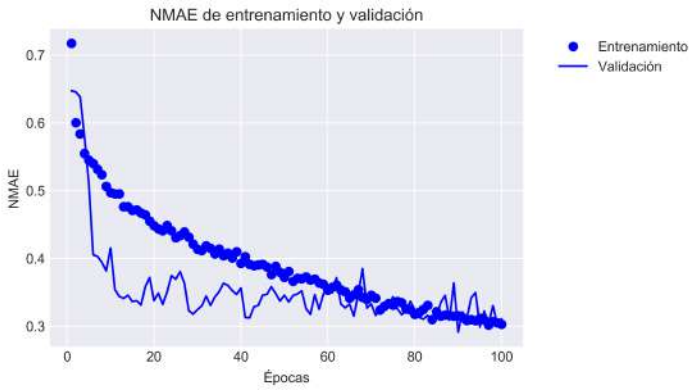
En la figura B.25 podemos observar que las predicciones de los valores de validación de H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) llevadas a cabo por el modelo MLP son mejor que las del modelo XGBost (figura B.23). Si recordamos, esto también sucedía en los problemas de enero y febrero y ya explicamos que se debe a la capacidad que tiene el modelo MLP para establecer relaciones y dependencias entre los valores de consumo predichos. Aunque a pesar de esto, las predicciones se alejan bastante de los valores predichos, sobre todo en los valores de los días 22 al 25 de marzo.

B.2.3.3. RNN

Vamos ahora a comprobar el rendimiento del mejor modelo RNN (apéndice B.1.3, figura B.2 (c)) para el problema de marzo.



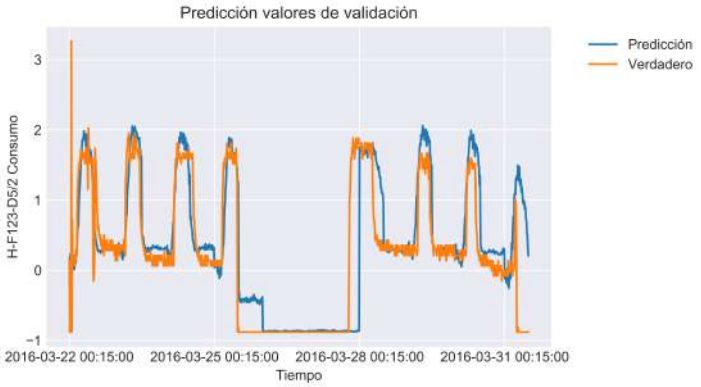
(a) H-F123-D1



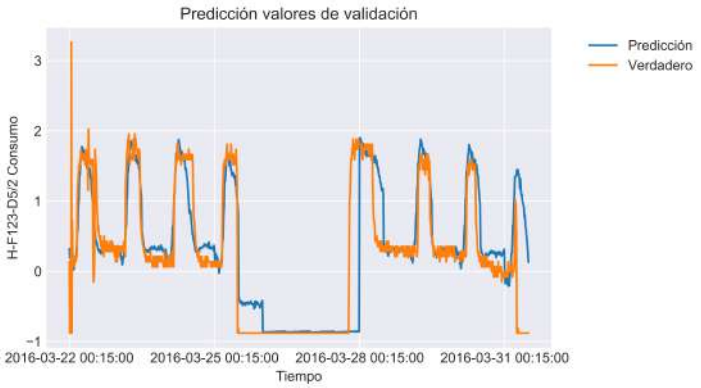
(b) H-F123-D5/2.

Figura B.26: Evolución del NMAE de entrenamiento y validación del mejor modelo de RNN para el problema de marzo

En la figura B.26 vemos durante el entrenamiento del mejor modelo de RNN para el problema de marzo para aprender a predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) que el NMAE de validación (línea con puntos) disminuye bastante durante las primeras 20 épocas para a continuación oscilar entorno a 0.30 y 0.40. Vemos que el NMAE de validación mínimo es de 0.29 tanto para H-F123-D1 como para H-F123-D5/2, por lo que el modelo RNN mejora en casi el doble el rendimiento de los modelos MLP y XGBoost, algo que ya pasaba también en los problemas de enero y febrero. Entonces, vamos a confirmar este rendimiento en términos de NMAE de validación de manera visual.



(a) H-F123-D1



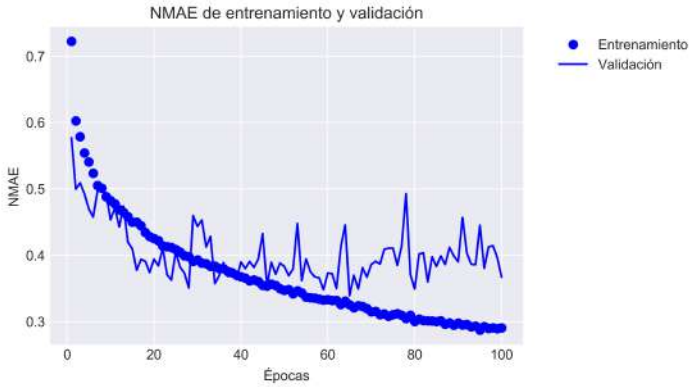
(b) H-F123-D5/2.

Figura B.27: Predicción de los valores de validación en marzo mediante RNN.

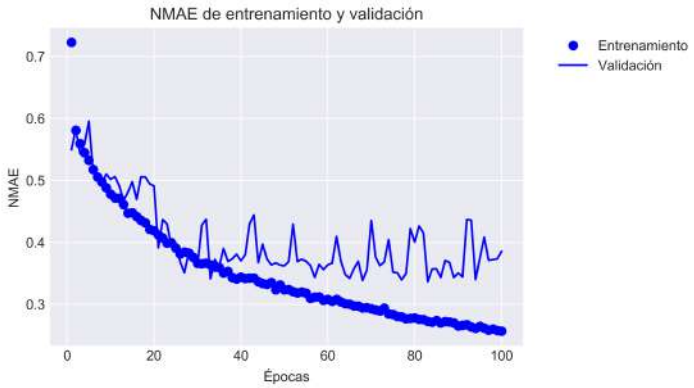
En la figura B.27 se puede ver que la predicción de los valores de validación de H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) es mucho más cercana a los valores verdaderos que en el caso de los modelos XGBoost y MLP. Por lo que confirmamos la mejora en el rendimiento observada anteriormente en términos de NMAE de validación, la cual sigue confirmando la capacidad que tienen los modelos RNN para extraer características temporales de las series temporales de entrada para facilitar el aprendizaje. Además de esto, podemos ver que el modelo RNN de este problema ha conseguido predicciones más parecidas a los valores verdaderos que en los problemas de enero y febrero, lo cual se debe principalmente a que en marzo los datos son más estacionales y por tanto fáciles de predecir, pues se repiten una y otra vez las mismas oscilaciones semanales en el consumo.

B.2.3.4. CNN

A continuación vamos a comprobar el rendimiento del mejor modelo de CNN (apéndice B.1.4, figura B.3 (c)) para el problema de marzo.



(a) H-F123-D1

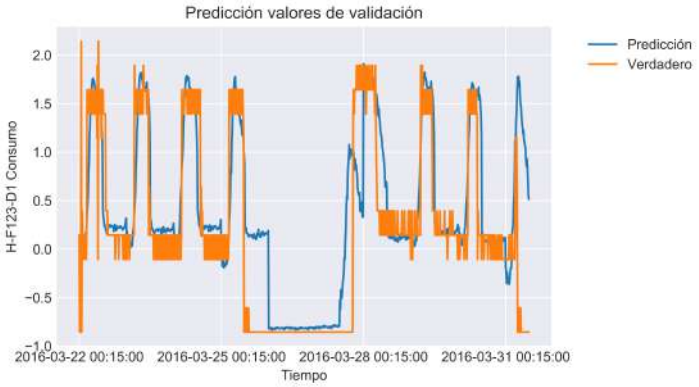


(b) H-F123-D5/2.

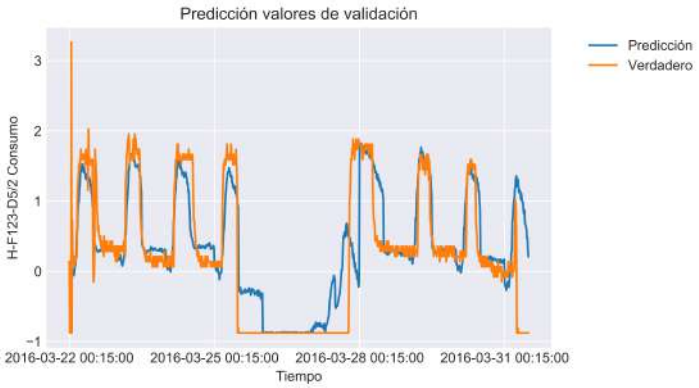
Figura B.28: Evolución del NMAE de entrenamiento y validación del mejor modelo de CNN para el problema de marzo

En la figura B.28 vemos que durante el entrenamiento del mejor modelo CNN del problema de marzo para aprender a predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b), tanto el NMAE de entrenamiento como de validación evolucionan de la misma forma en que lo hacían con el mejor modelo RNN de marzo (B.27). De hecho, podemos ver que el NMAE de validación mínimo alcanzado por el modelo CNN es de 0.34 y 0.33 a la hora predecir H-F123-D1 y H-F123-D5/2 respectivamente que son valores muy similares a los alcanzados por el modelo RNN aunque ligeramente superiores.

Dicho esto, veamos visualmente como predice este modelo CNN los valores de validación de ambas variables objetivo.



(a) H-F123-D1



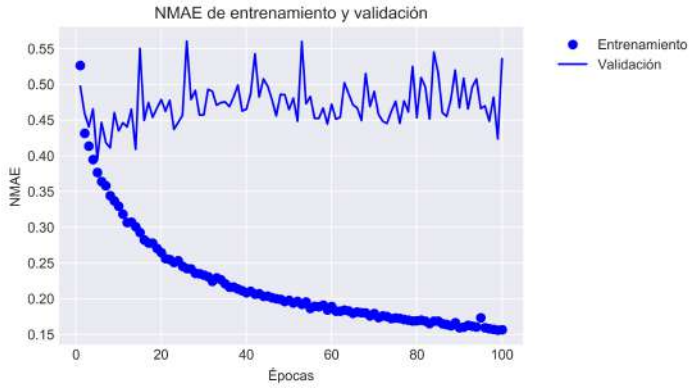
(b) H-F123-D5/2.

Figura B.29: Predicción de los valores de validación en marzo mediante CNN.

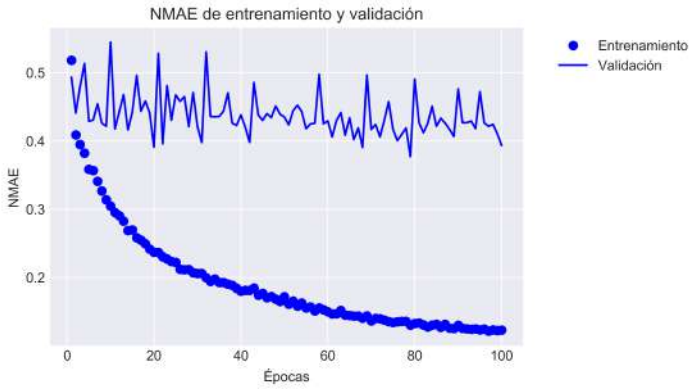
En la figura B.29 vemos que efectivamente el rendimiento del modelo CNN a la hora de predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) es similar al del modelo RNN (figura B.27. Sin embargo, el modelo CNN tiene un poco más de retardo en las predicciones y estima peor los valores de consumo nulos como los de los días 26, 27 y 28 de marzo.

B.2.3.5. Seq2Seq

Por último, veamos cómo rinde el mejor modelo Seq2Seq (apéndice B.1.5, figura B.4 (c)) para el problema de marzo.



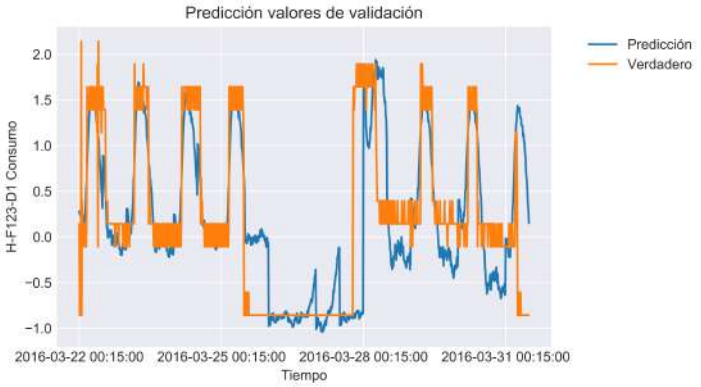
(a) H-F123-D1



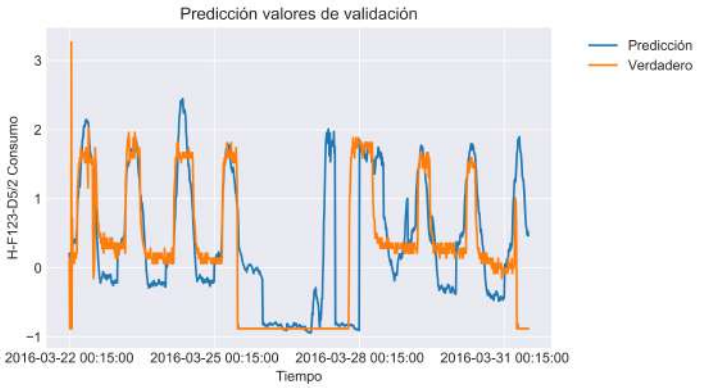
(b) H-F123-D5/2.

Figura B.30: Evolución del NMAE de entrenamiento y validación del mejor modelo de Seq2Seq para el problema de marzo

Vemos en la figura B.30 que durante el entrenamiento del mejor modelo Seq2Seq de marzo para aprender a predecir H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) el NMAE de validación (línea) disminuye un poco durante las primeras épocas para luego permanecer el resto de épocas oscilando entre 0.4 y 0.5 sin llegar del todo a sobreajustar. El sobreajuste puede estar evitado por la información contextual de temperatura y ocupación y por el dropout presente en las capas LSTM del decodificador del modelo Seq2Seq. En este caso vemos que el NMAE de validación mínimo alcanzado es de 0.39 y 0.37 a la hora de predecir H-F123-D1 y H-F123-D5/2. Por tanto, con el modelo Seq2Seq se obtiene un rendimiento peor que con los modelos RNN y CNN. Vamos a comprobarlo del todo de forma visual.



(a) H-F123-D1



(b) H-F123-D5/2.

Figura B.31: Predicción de los valores de validación en marzo mediante Seq2Seq.

En la figura B.31 se puede confirmar que el rendimiento del mejor modelo Seq2Seq del problema de marzo a la hora de predecir los valores de validación de H-F123-D1 (subfigura a) y H-F123-D5/2 (subfigura b) es peor que el mostrado por los modelos RNN (figura B.9) y CNN (figura B.13). Pues podemos ver en ambas subfiguras que los valores predichos presentan algunas oscilaciones que no están presentes en los valores verdaderos, sobre todo en los días 25, 26, 27 y 28 y son más bajos que los verdaderos en las horas laborales de todos los días en el caso de H-F123-D5/2 y de los días 28, 29, 30 y 31 en el caso de H-F123-D1. Esto último puede deberse a que en el conjunto de entrenamiento, el modelo ha aprendido que cuando los valores futuros de ocupación son bajos el consumo de calefacción también decae en una determinada cantidad y aplica esta regla al conjunto de validación, pero en el conjunto de validación cuando los valores de ocupación futuros decaen esta reducción en los valores del consumo de calefacción no es tan grande como en los valores de entrenamiento.

Apéndice C

Implementación

En este apartado se describe de forma general la implementación del proyecto, es decir, los scripts de código de python y R de los que se compone y las bibliotecas que se han empleado en ellos.

Este trabajo se ha desarrollado en un proyecto con el lenguaje de programación Python 3.6 [**60**] en el IDE PyCharm 2019.3.5 [**61**]. Además, también se ha empleado el lenguaje de programación R 3.6.1 [**62**] en el IDE RStudio 1.2.5033 [**77**]. Dicho esto, vamos a describir los distintos ficheros de los que se compone el proyecto y las bibliotecas que estos emplean.

C.1. Ficheros del proyecto

El proyecto está almacenado en el directorio energy-processing, dentro del cual se pueden encontrar otros dos directorios:

- src: Donde se localizan todos los scripts del proyecto.
- doc: Donde se almacenan algunos ficheros útiles para el proyecto, tal y como veremos a continuación.

Dentro de la carpeta src se encuentran los siguientes scriptsdel proyecto:

- data_extraction.py: Script que contiene la clase ICPEDAtaExtractor, la cual proporciona los atributos y métodos necesarios para conectarse al API que nos permite acceder a los datos de los sensores del edificio ICPE y extraerlos.
- data_analysis.py: Script que contiene un conjunto de funciones empleadas para llevar a cabo distintas tareas de análisis exploratorio y preprocesamiento.

- preprocessing.R: El único script escrito en R, el cual contiene el código necesario para llevar a cabo las tareas de preprocesamiento de visualización de los datos transformados, tratamiento de outliers y tratamiento de valores perdidos.

- XGBoost.py: Script que contiene la clase XGBoostTimeSeries, la cual proporciona una interfaz de alto nivel con los atributos y métodos necesarios para:

    1. Definir un problema de predicción multistep de serie temporal multivariable, estableciendo para ello el conjunto de datos que almacena la serie temporal, la variable objetivo que se desea predecir, el valor de los parámetros lookback y delay y las fechas de inicio y fin de entrenamiento y validación.
    2. Definir un modelo de XGBoost mediante el uso de una estrategia directa para afrontar el problema definido estableciendo su número de árboles, su profundidad máxima y la tasa de aprendizaje.
    3. Entrenar y validar el modelo de XGBoost definido. Generando para ello las muestras de entrenamiento y validación.
    4. Evaluar visualmente el modelo de XGBoost entrenado pintando en una gráfica los valores predichos de la variable objetivo a partir de las muestras de validación junto a los valores verdaderos.

    Para el desarrollo d este script nos hemos basado en **[56]**.

- NeuronalNetworks.py: Script que contiene tres clases empleadas para definir y experimentar con los modelos de redes neuronales empleados en este trabajo. Estas clases son las siguientes:

    - BasicTimeSeriesANN: Clase que proporciona una interfaz de alto nivel con los atributos y métodos necesarios para:
        1. Definir un problema de predicción multistep de serie temporal multivariable, estableciendo para ello el conjunto de datos que almacena la serie temporal, la variable objetivo que se desea predecir, el valor de los parámetros lookback y delay y las fechas de inicio y fin de entrenamiento y validación.
        2. Definir una arquitectura de RNA secuencial (MLP, RNN o CNN) con la que abordar el problema, estableciendo sus capas de entrada y de salida, las capas ocultas y sus hiperparámetros y su función de error y algoritmo de optimización.
        3. Entrenar y validar el modelo de red neuronal definido. Para ello la clase establece el tamaño de los lotes de entrenamiento

y validación, el número de épocas y si se muestran a la red las muestras de entrenamiento y validación en orden aleatorio o cronológico, tras lo cual genera con dichos parámetros los conjuntos de muestras de entrenamiento y validación con los cuales finalmente se entrena y valida respectivamente.

4. Evaluar visualmente el modelo de RNA entrenado pintando en una gráfica los valores predichos de la variable objetivo a partir de las muestras de validación junto a los valores verdaderos.

Para el desarrollo de esta clase nos hemos inspirado en [**45**].

- BasicTimeSeriesANNTunner: Clase que haciendo uso de un objeto de la clase BasicTimeSeriesANN permite probar diferentes arquitecturas de RNA secuenciales (MLP, RNN o CNN) para tratar de resolver un problema de predicción multistep de una serie temporal.
- SeqToSeqANN: Clase idéntica a la clase BasicTimeSeriesANN pero que únicamente permite definir modelos Seq2Seq. Para el desarrollo de esta clase nos hemos basado en [**51**].

- experiments.py: Script que contiene el código para ejecutar los experimentos dirigidos a la elección del mejor modelo de cada tipo (XGBoost, MLP, RNN, CNN y Seq2Seq) para cada problema (enero,febrero y marzo). En el caso de los experimentos con los modelos XGBoost y Seq2Seq, los hiperparámetros con los que hemos probado se encuentran en el código de este script pero los hiperparámetros con los que hemos probado para los modelos MLP, RNN y CNN se encuentran en los siguientes ficheros ubicados en el directorio experiments dentro del directorio doc:

  - modelos_mlp: Contiene todas las configuraciones de capas ocultas que hemos probado para los modelos MLP sobre los problemas de enero, febrero y marzo.
  - modelos_rnn_enero: Contiene todas las configuraciones de capas ocultas que hemos probado para los modelos RNN sobre el problema de enero.
  - modelos_rnn_febrero: Contiene todas las configuraciones de capas ocultas que hemos probado para los modelos RNN sobre el problema de febrero.
  - modelos_rnn_marzo: Contiene todas las configuraciones de capas ocultas que hemos probado para los modelos RNN sobre el problema de marzo.

- modelos_cnn_enero: Contiene todas las configuraciones de capas ocultas que hemos probado para los modelos CNN sobre el problema de enero.
- modelos_cnn_febrero: Contiene todas las configuraciones de capas ocultas que hemos probado para los modelos CNN sobre el problema de febrero.
- modelos_cnn_marzo: Contiene todas las configuraciones de capas ocultas que hemos probado para los modelos CNN sobre el problema de marzo.

Cada uno de estos ficheros almacena las configuraciones en una lista en formato json, donde cada elemento es una lista de diccionarios que representa una configuración de capas ocultas concreta (modelo de RNA), el cual en la posición i-ésima contiene un diccionario de pares clave-valor que especifica el tipo e hiperparámetros de la capa oculta i-iésima de dicha configuración. Cada uno de estos ficheros es leído y parseado por el modelo BasicTimeSeriesANNTunner el cual prueba todas y cada una de las configuraciones de capas ocultas del fichero sobre un problema concreto. Almacenamos estas configuraciones en disco para que el código de este script no sea demasiado largo y confuso.

- main.py: Script principal del proyecto que contiene el código desarrollado para llevar a cabo las siguientes tareas:

  1. Obtención de los valores de los sensores del edificio ICPE, haciendo uso de la clase ICPEDataExtractor definida en el script data_extraction.py.
  2. Análisis exploratorio y preprocesamiento de los valores de los sensores obtenidos, empleando las funciones del script data_analysis.py. Aunque las tareas de preprocesamiento de tratamiento de outliers y valores perdidos se lleva a cabo en R en el script preprocessing.R. Esto es debido a que el lenguaje R dispone de más herramientas para abordar estas dos tareas.
  3. Comparación del rendimiento de los mejores modelos de XGBoost, MLP, RNN, CNN y Seq2Seq para cada problema (enero, febrero y marzo) obtenidos en el script experiments.py. Para ello, hace uso de las clases XGBoostTimeSeries del script XGBoost.py y BasicTimeSeriesANN y Seq2SeqANN ambas del script NeuronalNetworks.py.

Todas las clases, funciones y métodos de estos scripts están debidamente comentados con el propósito de explicar su funcionamiento.

C.2. Dependencias del proyecto

Los scripts comentados en la anterior sección han sido desarrollados por nosotros haciendo uso de varias bibliotecas de python y R, las cuales se listan junto a su versión y propósito en las tablas C.1 y C.2.

Biblioteca	*Versión*	*Propósito*
CUDA [**69**]	8.0	Proporciona primitivas que permiten utilizar la targeta gráfica para computación paralela.
cuDNN [**70**]	6.0	Utilizar CUDA para ejecutar modelos de aprendizaje profundo de forma paralela.
Tensorflow [**63**]	1.4.3	Definir, entrenar y validar modelos de aprendizaje profundo empleando cuDNN y CUDA.
Keras [**71**]	2.1.5	Interfaz de alto nivel para Tensorflow.
Matplotlib [**78**]	3.0.0	Generar gráficas 2D.
Pandas [**79**]	0.24.2	Crear y gestionar conjuntos de datos.
Numpy [**80**]	1.18.2	Manejo de arrays de forma eficiente.
Pickle [**81**]	0.7.5	Almacenamiento de datos en disco en formato pkl.
XGBoost [**81**]	1-0.2	Creación de modelos de XGBoost.
Seaborn [**82**]	0.9.0	Generación de gráficas 2D.
Scikit-learn [**83**]	0.22	Utilidades para preprocesamiento y creación de modelos XGBoost de salida múltiple.
Requests [**84**]	2.22.0	Realizar peticiones al API para obtener los datos de los sensores del edificio ICPE.
Tensorboard [**63**]	2.1.1	Monitorización del entrenamiento de los modelos de aprendizaje profundo.

Tabla C.1: Bibliotecas empleadas en Python.

Biblioteca	*Versión*	*Propósito*
Tidyverse [**85**]	1.3.0	Utilidades para analizar y preprocesar conjuntos de datos y pintar gráficas 2D entre otras muchas.
TSA [**62**]	1.2	Utilidades para el análisis de series temporales.
Zoo [**86**]	1.8-7	Gestión de series temporales regulares e irregulares.
Forecast [**87**]	8.11	Imputación de series temporales.
Amelia [**38**]	1.7.6	Imputación de series temporales.
Mtsdi [**37**]	0.3.5	Imputación de series temporales.
VIM [**88**]	5.1.1	Imputación de series temporales.
Mice [**36**]	3.8.0	Imputación de series temporales.

Tabla C.2: Bibliotecas empleadas en R.

C.3. Reproducción del proyecto

Si se desea reproducir el proyecto llevado a cabo es necesario disponer de una tarjeta gráfica NVIDIA compatible con CUDA, tener instaladas en el sistema o en un entorno virtual las dependencias citadas en la tablas C.1 y C.2 con las versiones que aparecen en ellas, tener acceso al API que proporciona los datos de los sensores del edificio ICPE con un usuario y una contraseña y llevar a cabo los siguientes pasos:

1. Abrir en un proyecto de Pycharm los ficheros de python del directorio src.

2. Crear en PyCharm las variables de entorno USER y PASSWORD, las cuales almacenarán el usuario y la contraseña respectivamente que emplearemos para conectarnos con el API que nos proporcionará los valores de los sensores del edificio ICPE.

3. Ejecutar el código del script main.py del proyecto desde la línea 1 hasta la 386 para llevar a cabo las tareas de obtención de datos, análisis exploratorio y las primeras de preprocesamiento.

4. Abrir el script preprocessing.R con RStudio y ejecutarlo desde la línea 1 hasta la 518 para llevar a cabo la tarea de tratamiento de outliers y parte de la tarea de de tratamiento de valores perdidos.

5. Volver al script main.py en PyCharm y ejecutar el código desde la línea 389 hasta la 448 para llevar a cabo la imputación de valores perdidos con la técnica mice y mediante patrones similares.

6. Volver al script preprocessing.R en RStudio y ejecutarlo desde la línea 520 hasta el final, para terminar con el tratamiento de valores perdidos.

7. En el script main.py ejecutar el código desde la línea 450 hasta la 550 para llevar a cabo la tarea de selección de variables.

8. Ejecutar el script experiments.py para llevar a cabo la elección del mejor modelo de cada tipo para cada problema. Cada modelo entrenado, almacena unos logs en el directorio raíz del proyecto en la carpeta logs/{Nombre modelo}/{H-F123-D1}/{Mes}/{Nombre modelo}. Estos logs se pueden visualizar con la herramienta Tensorboard ejecutando en la terminal de PyCharm el comando $ python -m tensorboard.main –logdir={ruta al directorio con los logs de los modelos}.

9. Finalmente ejecutar el código de main.py desde la línea 552 hasta el final para entrenar los mejores modelos de cada tipo encontrados en el paso anterior para cada problema.