Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

Uso de Técnicas de Soft Computing en Problemas Energéticos en Edificios Inteligentes

Autor: Daniela Camila Durand Bartolo

Tutor: David Fernández Barrero

Cotutor: José Lisandro Aguilar Castro

2021

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado
Uso de Técnicas de Soft Computing en Problemas Energéticos en Edificios Inteligentes

Autor: Daniela Camila Durand Bartolo

Tutor: David Fernández Barrero

Cotutor: José Lisandro Aguilar Castro

TRIBUNAL:

Presidente: Julia María Clemente Párraga

Vocal 1º: María del Mar Lendínez Chica

Vocal 2º: David Fernández Barrero

FECHA: 5 de octubre de 2021

Agradecimientos

Este proyecto marca la culminación de una etapa muy importante en mi vida académica y profesional, y no habría sido posible sin el apoyo de las personas que, de una forma u otra, han estado presentes en todo este proceso.

En primer lugar, gracias a mi familia: mis padres Delia y Rafael, y mi hermana Melina, por haberme ayudado en todo lo que les ha sido posible, por haber confiado en mí en todo momento, y por haberme dado ánimos en momentos de flaqueza. Pero, sobre todo, agradezco el esfuerzo que han realizado mis padres para poder brindarme esta oportunidad.

Agradecimientos especiales a mis tutores del Trabajo de Fin de Grado, María Dolores Rodríguez Moreno y José Lisandro Aguilar Castro, por haberme guiado y ayudado en el desarrollo de este proyecto, así como a David Fernández Barrero, por haber mostrado su disposición a ayudar.

También agradezco al resto de personal de la universidad por haber contribuido a este periodo de formación y aprendizaje, así como a los compañeros con los que he compartido esta etapa, pues con ellos he ganado experiencias y recuerdos gratos.

Por último, sinceros agradecimientos a todas las demás personas que han contribuido de alguna forma a sacar este trabajo adelante, especialmente a aquellos que tienen el espíritu de cooperación y la dedicación para compartir en Internet sus conocimientos, sin los cuales elaborar este TFG habría sido una tarea mucho más ardua.

Resumen

Este Trabajo de Fin de Grado (TFG) explora el proceso de predicción del consumo de energía en edificios inteligentes en función del consumo de los dispositivos y aparatos. Para ello, se generan modelos de predicción empleando técnicas de Aprendizaje Automático (AA), teniendo en cuenta que se trabaja con series temporales, ya que el consumo en cierto momento puede estar relacionado con el consumo en momentos anteriores. El trabajo detalla los pasos del proceso de predicción, presentando en primer lugar los conceptos teóricos necesarios, para luego aplicarlos a un conjunto de datos reales.

Palabras clave: Predicción, Consumo de energía, Edificios Inteligentes, Aprendizaje Automático, Series Temporales.

Abstract

This Final Degree project (TFG) explores the process of prediction of energy consumption in Smart Buildings based on the devices' consumption. To do so, forecasting models are generated using Machine Learning (ML) techniques, taking into consideration that we are working with time series since the consumption at a given time could be related to the consumption at previous times. The project details each step of the prediction process, presenting first the necessary theoretical concepts, so then, they are applied to a real dataset.

Keywords: Forecasting, Energy Consumption, Smart Buildings, Machine Learning, Time Series.

Resumen extendido

A lo largo de este TFG se estudia el proceso de análisis de datos y generación de modelos de predicción del consumo de energía eléctrica en Edificios Inteligentes. Uno de los mayores desafíos actuales es el realizar un consumo eficiente de energía eléctrica debido a factores económicos, comerciales y medioambientales. Por ello, gracias a los datos recogidos en Edificios Inteligentes, se realiza un estudio sobre este consumo de energía en función del tiempo, para poder obtener un modelo capaz de estimar el consumo total conociendo el consumo de los dispositivos y aparatos de una estancia del Edificio Inteligente.

Para lograr esto, nos apoyamos en un campo del *Soft Computing* y de la Inteligencia Artificial: el Aprendizaje Automático (AA). Gracias al AA, se ha podido analizar una serie temporal con datos sobre el consumo energético y generar modelos de predicción. El proceso se ha realizado en una serie de fases definidas en la metodología CRISP-DM (Cross Industry Standard Process for Data Mining), dada su adecuación para proyectos de AA.

Para empezar a trabajar, es primordial entender el objeto de nuestro estudio y explorar investigaciones y técnicas aplicadas a este ámbito. De esta forma, los primeros capítulos de este TFG se centran en exponer las ideas que mueven este proyecto, así como los conceptos teóricos que se necesitan entender previamente para ser aplicados de forma práctica. Una vez hecho esto, se detalla el proceso experimental llevado a cabo.

En primer lugar, se realizó una búsqueda de conjuntos de datos que cumplan las condiciones para nuestro proyecto, es decir, que recoja datos del consumo energético total y de cada dispositivo. Teniendo una lista de candidatos, se valoró cuál era el conjunto más adecuado. El *dataset* seleccionado se trata de una serie temporal sobre el consumo energético del Ala B Este del edificio *Research Support Facility* del Laboratorio Nacional de Energía Renovable de Estados Unidos.

Ya teniendo el conjunto de datos, realizamos un análisis de sus variables para prepararlo, para obtener modelos de estimación precisos. Tras una primera limpieza de datos (columnas de ceros, valores NaN, etc.), pasamos a estudiar las relaciones de las variables del *dataset*. Por una parte, se analiza las relaciones entre variables, gracias a técnicas como las correlaciones de Pearson y Spearman, y modelos de Regresión Lineal Múltiple. A partir de los resultados obtenidos con ellas, se realiza una extracción de características con la técnica de Análisis de Componentes Principales (PCA). Por otra parte, se analiza la relación de cada variable consigo misma a través del tiempo, con técnicas como la autocorrelación (simple y parcial), y los modelos ARIMA, que nos ayudan a determinar la ventana temporal con la que generar los modelos de predicción.

Con el *dataset* resultante de la fase de preparación de los datos, pasamos a generar los modelos de estimación. En concreto, se generan modelos con redes neuronales *Long Short-Term Memory* (LSTM), debido a su arquitectura, que permite tratar series temporales, y a su memoria a "largo plazo". Esta fase genera una serie de modelos, entrenados con el 70% del total de datos, y probados con el 30% restante. Los modelos se van evaluando para ajustar sus parámetros e intentar generar, cada vez, un modelo más preciso que el anterior. Como métricas de evaluación se utiliza RMSE, MAPE y R^2. La generación de modelos de predicción se organiza en tres grupos, partiendo de un modelo inicial que relaciona sus parámetros con la ventana de tiempo obtenida en la fase anterior. Estos grupos de modelos se diferencian por las variables que se consideran como descriptoras en cada uno de ellos.

Posteriormente, se comparan los valores de las métricas de calidad obtenidos de los modelos de cada grupo, y se realiza una interpretación de los resultados, para proponer uno de ellos como el modelo más preciso. Este modelo resulta ser el que incluye como variable descriptora sólo al primer Componente Principal obtenido con PCA, confirmando la importancia de tratar las variables antes del modelado. Sin embargo, los valores de RMSE (0.07), MAPE (0.10) y R^2 (0.74) obtenidos con este modelo podrían ser mejorados, aunque esto no se realiza en este proyecto.

Por último, se concluye que el AA es una herramienta que permite realizar predicciones con series temporales de forma autónoma, aunque para ello hay que realizar una buena preparación de los datos, además de que demanda un consumo de recursos temporales y tecnológicos. También se destaca, entre otras cosas, que seguir una metodología ha sido beneficioso para desarrollo del proyecto. Por otro lado, se presentan posibles mejoras y ampliaciones, con las que se podría obtener modelos más precisos.

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

AA: Aprendizaje Automático.

ARIMA: Autoregressive Integrated Moving Average (Autoregresivo Integrado de Medias Móviles).

BPTT: Backpropagation Through Time (Propagación hacia atrás a través del tiempo).

CRIPS-DM: Cross Industry Standard Process for Data Mining.

Covid-19: Coronavirus Disease.

HVAC: Heating, Ventilation and Air Conditioning (Calefacción, Ventilación y Aire Acondicioneado).

IoT: Internet of Things (Internet de las Cosas).

LSTM: Long Short-term Memory.

MAPE: Mean Absolute Percentage Error (Error Porcentual Absoluto Medio).

ML: Machine Learning.

MLR: Multiple Linear Regression (Regresión Lineal Múltiple).

PC: Principal Component (Componente Principal).

PCA: Principal Component Analysis (Análisis de Componentes Pricipales).

R2: R Squared (R Cuadrado).

RF: Regression Forest (Bosque de Regresión).

RMSE: Root Mean Square Error (Error Cuadrático Medio).

RNN: Recurrent Neural Network (Red Neuronal Recurrente).

SEIRD: Susceptible, Exposed, Infected, Recovered, y Dead (Susceptible, Expuesto, Infectado, Recuperado y Muerto).

SVR: Support Vector Machine (Regresión de Vectores de Soporte).

VRF: Variable Refrigerant Flow.

XGB: Extreme Gradient Boosting (Aumento d Gradiente Extremo).

Capítulo 1.
Introducción

En este capítulo se realiza un primer acercamiento al estudio propuesto para tener claro de qué trata el proyecto y cómo se va a desarrollar. Para ello, se presentan cuatro secciones. En la primera de ellas se detalla el contexto de aplicación del estudio y las razones por las que se lleva a cabo, introduciendo también el concepto de Aprendizaje Automático. En la segunda sección se explica qué se busca con este proyecto, presentando para ello un objetivo general y una serie de objetivos específicos. En la tercera sección se da a conocer la metodología a seguir para el desarrollo del proyecto, así como sus fases y tareas, relacionándolas con lo realizado en el trabajo. La cuarta y última fase, presenta la organización de la presente memoria, con un breve resumen de cada capítulo.

1.1 Contexto y motivación

¿Alguna vez nos hemos preguntado si seríamos capaces de vivir sin energía eléctrica? Ciertamente, sería bastante complicado dado que hemos llegado a un punto en el que la energía eléctrica es de vital importancia en la mayoría de las sociedades actuales.

Prácticamente, cualquier situación o actividad de la vida cotidiana en la que podamos pensar requiere de consumo de energía eléctrica. Si queremos comunicarnos, necesitamos un dispositivo para hacerlo, el cual necesita ser cargado. Si tenemos frío o calor, usaremos la calefacción o aire acondicionado. Si tenemos hambre, usaremos la cocina, el microondas, la tostadora, la cafetera, además de la nevera que está en constante funcionamiento. Si hacemos limpieza, usaremos la aspiradora, la lavadora o la secadora. Si queremos tener un momento de ocio o entretenimiento usaremos la consola, la TV, el ordenador, la tablet, o el móvil. Si tenemos un coche eléctrico, necesitaremos cargarlo. Incluso, si simplemente estamos en un lugar, encenderemos las luces.

Otro aspecto importante en el que la energía eléctrica ha cobrado protagonismo es el trabajo, sobre todo en aquellos en los que se utiliza la tecnología y aparatos o herramientas eléctricas. Además, a raíz de la pandemia, trabajos que se realizaban en oficinas han pasado a realizarse en los hogares, así como las clases. Estas

últimas, antes impartidas de forma presencial en escuelas y universidades, han pasado a darse de forma on-line. Todo esto ha traído consigo el hecho de que el costo de la energía que antes asumían las empresas y centros educativos, pasen a ser asumidos por trabajadores y alumnos.

Ante este uso masivo de la energía eléctrica, es normal que nos planteemos la posibilidad de realizar un consumo eficiente, tanto en hogares, como en centros de trabajo o estudio y lugares públicos. Si hablamos de un hogar, una de las razones podría ser el evitar que nos llegue una factura de la luz con precios excesivos, o tener un comportamiento responsable para con el medio ambiente; mientras que, si hablamos de empresas o edificios, podría ser, además de las razones anteriores, para dar mayor comodidad a sus usuarios.

Precisamente, en este marco de consumo de energía eléctrica eficiente encontramos a los llamados Edificios Inteligentes o Smart Buildings. Un Edificio Inteligente [1] es aquel que está diseñado de tal manera que conecta subsistemas de iluminación, climatización y seguridad, entre otros, con ayuda de la tecnología, con el objetivo de ofrecer a sus ocupantes la mayor comodidad y personalización posible, a la vez de reducir el gasto económico y la contaminación del medio ambiente.

Para lograr un consumo eléctrico óptimo por parte de los Edificios Inteligentes, es necesario realizar un estudio sobre cómo acontece dicho consumo. De esta forma, se podrían detectar y solucionar posibles problemas energéticos, que es justamente el ámbito de estudio de este proyecto.

Entre los posibles casos a analizar sobre el consumo de energía en Edificios Inteligentes podemos encontrar los siguientes: la predicción de la carga eléctrica; la clasificación de los consumidores en base a ciertos factores, como información sociodemográfica; la clusterización de perfiles de carga en función del comportamiento de consumo; la detección de anomalías en el consumo; y la estimación del consumo de energía en casas y apartamentos, entre otros.

El foco de este proyecto se centrará en la estimación del consumo de energía en estancias de Edificios Inteligentes en función del consumo de los aparatos y dispositivos que haya en ellas.

Para realizar esto usamos técnicas de *Soft Computing*, en concreto, de Aprendizaje Automático (AA) o Aprendizaje de Máquina, del inglés Machine Learning (ML). El AA [2], [3] es una rama de la Inteligencia Artificial (IA) que se está usando en gran variedad de campos actualmente. A pesar de haber ganado fuerza en las últimas décadas, el término fue acuñado en 1959 por Arthur Samuel, quien lo definió como “el campo de estudio que da a los computadores la capacidad de aprender sin ser explícitamente programados”. En otras palabras, el AA recibe un conjunto de datos de entrada y eventualmente sus correspondientes datos de salida. A partir de ellos, es capaz de construir modelos de conocimiento. Por ejemplo, con datos de entrada y salida es capaz de entrenar un modelo, de tal forma que, si introducimos un nuevo dato de entrada, nos devuelve de forma aproximada su correspondiente valor de salida. La forma de “aprender” del AA intenta imitar el aprendizaje humano, es decir, realiza un cierto número de repasos sobre los datos de entrada y salida ajustando sus parámetros en este proceso.

Por otro lado, en este trabajo el AA será aplicado sobre Series Temporales o Time Series [4], [5]. A lo largo del desarrollo de este proyecto se presentarán más detalles sobre las series temporales y su análisis, pero, grosso modo, esto significa que los datos están ordenados en el tiempo. Por lo tanto, la estimación depende de ese orden temporal, el cual no puede ser alterado. La razón de trabajar con series temporales radica en que el consumo de energía viene marcado por épocas del año, días de la semana, o incluso horas en un mismo día. Por ejemplo, en épocas de Navidad puede haber un mayor consumo por las luces navideñas, mientras que en meses de vacaciones el consumo puede ser menor si no estamos en casa; o los fines de semana podríamos tener un consumo menor que los días de semana en una oficina; o el consumo por la noche podía ser menor que el consumo durante el día.

1.2 Objetivos del proyecto

Después de haber comprendido la importancia de realizar una buena gestión del consumo energético, tanto para nuestro bien como para el del planeta, y sabiendo que nos podemos apoyar en el AA para conseguirlo, podemos decir que el objetivo general de este proyecto consiste en: **proponer un modelo de estimación del consumo de energía en Edificios Inteligentes.**

Para lograr esto, se plantea una serie de objetivos específicos, enfocados en las tareas que son necesarias para obtener un modelo eficiente. Estos objetivos son enumerados a continuación:

- Búsqueda y obtención de datos históricos que puedan ser utilizados para entrenar un modelo de predicción. En concreto, nos centraremos en conjuntos de datos que muestren el consumo de energía eléctrica total y el de los diferentes aparatos y dispositivos medidos en casas, apartamentos o zonas de un Edificio Inteligente. Para ello se utilizará fuentes de datos abiertas.

- Elección del conjunto de datos, o dataset, más adecuado para realizar el modelado, dependiendo de factores como la cantidad y los tipos de variables, el tiempo de muestreo y la cantidad de observaciones.

- Análisis de las variables del conjunto de datos elegido, determinando previamente cuál será la variable a predecir y cuáles serán las variables usadas para ello. Posteriormente, se aplicarán distintas técnicas estadísticas para entender la relación entre estas variables y la relación con ellas mismas, entre otras cosas. De esta forma, se dejará el conjunto de datos preparado para proceder a la generación de modelos.

- Elección del tipo de modelo de predicción a generar y de las técnicas de AA a utilizar para ello. En base a ello, se pasará a la elaboración de distintos modelos de predicción y al ajuste de sus parámetros, para obtener mejores resultados. Estos modelos serán entrenados con una porción del conjunto de datos elegidos, llamados datos de entrenamiento.

- Evaluación de los modelos de predicción generados, en función de las pruebas de predicción realizadas sobre la parte restante del conjunto de datos, llamados datos de prueba, y según las métricas de calidad más adecuadas. De esta forma, valoraremos la precisión de cada modelo, y seremos capaces de interpretar los resultados del estudio.

- Obtención de un modelo de predicción de entre los evaluados anteriormente. Para ello, se realizará una comparación entre los modelos generados y se elegirá uno, el cual debe ser el más preciso y eficiente.

Este proyecto nos permitirá introducirnos en el campo del análisis y la predicción de datos, explorando algunas de las técnicas de Aprendizaje Automático que se están utilizando actualmente para ello, así como algunas de las técnicas

estadísticas más importantes sobre las que se apoyan los estudios para resolver este tipo de problemas. De esta manera, se podrá comprender la razón por la que el AA ha ganado tanta relevancia en los últimos tiempos, y qué ventajas aporta frente a los métodos tradicionales.

Además, al centrarnos en la predicción del consumo de energía en Edificios Inteligentes, comprenderemos los aspectos más importantes para la aplicación del AA en este ámbito, trabajando en un problema del mundo real. De este modo, intentaremos contribuir a la gestión eficiente del consumo de energía eléctrica, factor determinante en estas épocas, dado que esto conlleva un ahorro de energía y, por tanto, una reducción de costes. A esto hay que sumar el hecho de que los sistemas de gestión de consumo energético son de gran ayuda para la preservación del medio ambiente y al mismo tiempo, permiten ofrecer mayor comodidad y personalización a los usuarios de estos Edificios Inteligentes.

1.3 Metodología

En cuanto a la metodología a seguir, se ha optado por CRISP-DM (*Cross Industry Standard Process for Data Mining*) [10]-[13], ya que es la más utilizada actualmente en proyectos de minería de datos y Aprendizaje Automático, por las ventajas que presenta para el desarrollo de este tipo de proyectos. La razón de que esta metodología sea tan utilizada en procesos de minería de datos es su estructura intuitiva y lógica, que refleja de forma clara el ciclo de vida de estos procesos. CRISP-DM consta de seis etapas, cada una con una serie de tareas, que corresponden a los pasos a seguir cuando se aplica AA.

La primera etapa se denomina "Business Understanding" o "Entendimiento del negocio". En ella se define las necesidades del negocio o, en otras palabras, el porqué del desarrollo del proyecto, planteando los objetivos que se quieren alcanzar. Aunque en este caso se aplique a proyectos de AA, esta fase en realidad es común a cualquier tipo de proyecto, ya que se trata del punto de partida, en el que se proponen y establecen ideas para empezar a desarrollarlo. Las tareas de esta fase incluyen la determinación de los objetivos del negocio, la evaluación de la situación, la determinación de los objetivos de la minería de datos, y la elaboración de un plan de proyecto. En el desarrollo de nuestro proyecto en concreto, estas tareas comprenden los pasos seguidos para elaborar el anteproyecto, y la introducción y estado del arte de esta memoria.

La segunda etapa corresponde a "Data Understanding" o "Entendimiento de los datos". Se trata de la fase en la que, una vez teniendo claro qué es lo que se va a desarrollar, nos disponemos a buscar, elegir, analizar y, como su nombre lo indica, entender los datos con los que se va a trabajar. Las tareas que se incluyen en esta etapa son la recopilación de los datos iniciales, la descripción de los datos, la exploración de los datos y la verificación de su calidad. De esta manera, nos hacemos una idea sobre la clase de datos que vamos a usar y qué podemos hacer con ellos. Esta parte de la metodología corresponde al proceso de elección del conjunto de datos que se utilizará para obtener el modelo de estimación.

La tercera etapa recibe el nombre de "Data Preparation" o "Preparación de los datos". Es aquí donde decidimos, finalmente, el o los conjuntos de datos que se utilizarán. Es necesario realizar un análisis sobre estos datos para que lleguen a la fase de generación del modelo con las condiciones necesarias para ser utilizados por las herramientas de modelado. Esto quiere decir que tendremos que examinar sus variables, valores y patrones para transformar nuestro conjunto de datos, ya sea quitando valores que no aportan información, obteniendo atributos derivados, o formateando y normalizando el dataset. Entre las tareas de esta etapa se encuentran la selección de datos, la limpieza de datos, la construcción de datos, la integración de datos y el formateo de datos. Esta es la parte más laboriosa y la que más tiempo requiere de todo el proceso, ya que es necesario preparar bien los datos si queremos obtener modelos eficientes. En los puntos 3.1 y 4.1 de esta memoria se ve reflejada, de forma teórica y práctica, respectivamente, esta etapa de la metodología.

La cuarta etapa se refiere al “Modeling” o “Modelado”. Se trata de la fase en la que se utilizan los datos preparados en el paso anterior para generar uno o varios modelos de estimación, y ajustar sus parámetros, en función de las técnicas de AA que se hayan seleccionado para ello. Además, es necesario dividir nuestro conjunto de datos en subconjuntos de entrenamiento y de prueba, de forma que con el primero entrenamos el modelo y con el segundo comprobamos la precisión de las predicciones. Las tareas que se incluyen en esta fase son la selección de técnicas de modelado, la generación de un diseño de pruebas, la construcción y ajuste de los modelos, y la valoración de los modelos. De forma similar a la fase anterior, esta también se refleja en los puntos 3.2 y 4.2 de esta memoria, de forma teórica y práctica, respectivamente.

La quinta etapa corresponde a “Evaluation” o “Evaluación”. En esta fase interpretamos los resultados obtenidos, comparando los modelos generados, y evaluando cuál se ajusta más a los objetivos iniciales del proyecto. También es importante hacer retrospección para encontrar posibles defectos y determinar si se realizarán más iteraciones o se pasará al despliegue. Las tareas correspondientes a esta fase son la evaluación de resultados, la aprobación de modelos, y la determinación de los siguientes pasos. Esta etapa se encuentra documentada en los puntos 3.2 y 4.3 de esta memoria, de forma teórica y práctica, respectivamente.

Por último, la sexta etapa se denomina “Deployment” o “Despliegue”. En ella se presenta los resultados del estudio. En caso de tratarse de un proyecto que será puesto en producción, será necesario preparar el despliegue y futura monitorización de los datos, así como la documentación necesaria para presentar el proyecto a los clientes. Las tareas concernientes a esta fase son la elaboración de un plan de despliegue, de un plan de monitorización y mantenimiento, de un reporte final, y la revisión del proyecto. En nuestro caso concreto, esta fase corresponde a la preparación de esta memoria y la defensa del presente trabajo.

A pesar de que esta metodología presenta etapas muy diferenciadas en el proceso de desarrollo de un proyecto de AA, es bastante flexible, ya que estando en una etapa, es posible volver a etapas anteriores, según sea necesario. En la **Figura 1** se muestra cómo se ha planteado este proyecto de acuerdo a CRISP-DM.

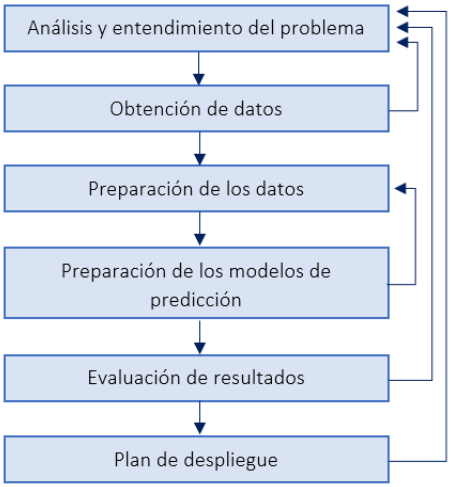

Figura 1. Proceso de desarrollo del proyecto con CRISP-DM

1.4 Estructura y contenidos

El contenido del presente documento se divide en cinco capítulos: introducción, estado del arte, base teórica, desarrollo práctico y, conclusiones y líneas futuras.

La **introducción** detalla el contexto en el que se ha desarrollado la idea de este proyecto, explorando el tema de los Edificios Inteligentes y la gestión eficiente del consumo energético en ellos. Además, se puntualizan los objetivos del proyecto, tanto el general como los específicos. Por otro lado, se explica la metodología elegida para el desarrollo del proyecto, en este caso CRISP-DM, describiendo cada una de sus fases. Por último, se incluye la presente sección, en la que se ofrece un resumen de cada capítulo.

La **base teórica**, por un lado, comprende una revisión de la literatura sobre el campo de estudio tratado. En concreto, se mencionan tres publicaciones, resumiendo el proceso seguido y las técnicas utilizadas en cada una. También se incluye un artículo sobre la generación de modelos de predicción con técnicas de Aprendizaje Automático en el contexto de la pandemia de Covid-19, ya que, aunque no está relacionado con nuestro campo de aplicación, realiza un estudio completo sobre la obtención de modelos con series temporales. A partir de ello, se presenta cuál será el enfoque seguido en este proyecto. Por otro lado, se explican los conceptos de las técnicas utilizadas en el desarrollo práctico. Cada concepto se encuentra en la sección que corresponde, dependiendo de si se ha utilizado para el entendimiento y preparación de datos, para el modelado, o para la evaluación de los modelos.

El **desarrollo práctico** incluye el proceso de obtención de los modelos de predicción. En esta sección se pone en práctica el entendimiento y preparación de datos, el modelado y la evaluación de los modelos, siguiendo de forma paralela el capítulo anterior, ya que se utilizan los conceptos explicados en él. Finalmente, se realiza una interpretación de los resultados y se presentan las conclusiones, para luego proponer el mejor modelo.

Por último**,** en **conclusiones y líneas futuras**, se realiza una revisión del proyecto, pasando por los puntos más relevantes, y puntualizando las ventajas e inconvenientes del Aprendizaje Automático. Por otro lado, se proponen posibles ampliaciones futuras, como el uso de otras técnicas, la comparación con otros tipos de modelo, la inclusión de variables climatológicas en el conjunto de datos, o su aplicación en entornos reales.

Capítulo 2.
Base teórica

En este capítulo se introducen los fundamentos teóricos necesarios para entender el proceso de desarrollo de este proyecto. Para ello, en primer lugar, se presenta el "estado del arte", es decir, una revisión de estudios recientes en el campo de la estimación del consumo energético en Edificios Inteligentes, así como un estudio en el que se aplican técnicas de AA a series de datos temporales. Posteriormente, se presentan tres secciones, correspondientes a las fases de entendimiento y preparación de los datos, modelado, y evaluación de modelos, respectivamente. En cada una de ellas, se explica de forma teórica las técnicas necesarias para el desarrollo práctico, para poder entender su uso y saber interpretar los resultados. Así, en la segunda sección se presenta la correlación de Pearson, los modelos de Regresión Lineal Múltiple, la correlación de Spearman, el Análisis de Componentes Principales, la autocorrelación simple y parcial, y los modelos ARIMA; la tercera sección presenta los modelos con redes neuronales *Long Short-Term Memory* (LSTM), haciendo un repaso previo sobre las Redes Neuronales Recurrentes; y la cuarta sección presenta las métricas de calidad con la se evalúan los modelos (RMSE, MAPE y R^2).

2.1 Estado del arte

La investigación en el campo del consumo de energía eléctrica se encuentra en pleno desarrollo debido a las ventajas que ofrece el hecho de tener un control sobre el consumo eléctrico. Hasta la fecha se han publicado múltiples estudios relacionados con la predicción del consumo de energía eléctrica. Cada estudio usa diferentes técnicas de Aprendizaje Automático para abordar el problema, intentando obtener un modelo robusto.

En concreto, para el desarrollo de este proyecto se ha prestado especial atención a tres publicaciones recientes [6]-[8]:

- La primera de ellas data del año 2017, y se titula *"Data partitioning and association mining for indentifying VRF energy consumption patterns under various part loads and refrigerant change directions"* [6].

  El estudio se centra en la evaluación de patrones de consumo de energía en sistemas de Refrigeración de Flujo Variable (VRF). Los sistemas VRF son un

tipo específico de sistemas de Calefacción y Aire Acondicionado (HVAC), y tienen la peculiaridad de que pueden trabajar bajo distintas condiciones de carga y refrigeración. Por esta razón, el artículo presenta una solución que mezcla dos técnicas para tratar esta situación. Por un lado, utiliza Agrupamiento de Datos (Data Clusterization) y, por otro, utiliza reglas de Asociación entre variables (Data Association Mining). De esta forma es capaz de obtener patrones y reglas de consumo energético de los sistemas VRF, con el objetivo de interpretarlos para poder mejorar la eficiencia energética y provocar un ahorro de energía. Además, el desarrollo de este estudio sigue los pasos básicos de un proyecto de AA, concretamente: preparación de los datos, Agrupamiento de Datos, Asociación entre variables, e interpretación de resultados.

- La segunda publicación también data del año 2017 y se titula *"Data driven modeling for energy consumption prediction in smart buildings"* [7].

  Este artículo aborda el problema de predicción del consumo de energía de un edificio a través de dos enfoques, para luego realizar una comparación entre ellos. Por un lado, se realiza un modelado con enfoque de caja negra, es decir, no se tiene en cuenta los aspectos físicos del problema. Sólo se considera el conjunto de datos recopilados en forma de serie temporal, que es utilizado para entrenar el modelo. Se generan varios modelos basados en técnicas estadísticas y de AA, como *Support Vector Regression* (SVR), *Regression Forest* (RF) y *Extreme Gradient Boosting* (XGB). Por otro lado, se realiza el modelado con un enfoque de caja gris, en el que sí se tiene en cuenta la información física del sistema energético del edificio, en concreto, datos físicos sobre la transferencia de calor. A partir de esto, se generan modelos en los que se incluyen ecuaciones físicas. Tras realizar la comparación, se observa que el primer enfoque, basado en AA muestra resultados más precisos que el enfoque que considera la física del problema. En concreto, los mejores resultados se obtienen con RF. Finalmente, los autores concluyen que los modelos basados en datos son más eficientes que los basados en modelos físicos.

- El tercer artículo data del año 2020, y se titula *"Multiple Electric Energy Consumption Forecasting Using a Cluster-Based Strategy for Transfer Learning in Smart Building"* [8].

  Esta publicación plantea un método para la estimación del consumo de energía de múltiples apartamentos en Edificios Inteligentes, ya que cada apartamento tiene un perfil de consumo distinto. Los enfoques tradicionales presentan modelos entrenados con series temporales de un determinado perfil, mientras que este artículo propone un enfoque con el que se puede realizar la estimación teniendo varios perfiles. Para ello, se divide el proceso en dos fases. En primer lugar, se realiza una clusterización de la carga energética diaria de los distintos perfiles utilizando K-means como algoritmo de agrupación de datos. De esta forma se obtienen los datos de entrenamiento, que son utilizados en la segunda fase para generar un modelo de pronóstico usando la técnica Long Short-Term Memory (LSTM). El modelo propuesto en este artículo presenta resultados favorables, además de suponer menor coste, tanto computacional como económico, frente a otros enfoques.

Por otro lado, para el desarrollo del proyecto se ha tenido como base el artículo *"Machine learning models for the prediction of the SEIRD variables for the COVID-19 pandemic based on a deep dependence analysis of variables"* [9]. Se trata de un estudio sobre la predicción de las variables SEIRD (*Susceptible*, *Exposed*, *Infected*, *Recovered*, y *Dead*), en función de una serie de variables sociodemográficas como la población, número de personas mayores de 65 años, índice de pobreza, tasa de morbilidad, edad media y densidad de población.

A pesar de que este artículo no está relacionado con la estimación del consumo de energía, refleja bastante bien el proceso a seguir para obtener modelos de predicción con series temporales. Además, en este estudio se explora la base teórica de las técnicas estadísticas y de AA necesarias para el proceso de análisis de datos. Algunas de estas técnicas son las correlaciones entre variables, autocorrelaciones, modelos de regresión y modelos ARIMA, en las que se profundizará más adelante.

Una vez revisados los estudios que se están llevando a cabo en el campo de la estimación del consumo energético, se plantea cuál será el enfoque que se seguirá en este proyecto: un modelado en el que se tendrá en cuenta únicamente la serie temporal sobre el consumo de energía en un espacio determinado, de forma similar al enfoque de caja negra propuesto en el segundo artículo [7].

Para ello se realizará, en primer lugar, un análisis de los datos utilizando algunas de las técnicas estadísticas expuestas en el artículo sobre la predicción de las variables SEIRD [9], como la correlación de Pearson, modelos de regresión lineal, la correlación de Spearman, las autocorrelaciones y modelos ARIMA. Una vez hecho eso, se pasará a la generación de modelos LSTM, que serán evaluados según las métricas de calidad RMSE, MAPE y R^2.

2.2 Entendimiento y preparación de datos

La obtención de un modelo preciso puede resultar una tarea complicada si no se entienden y preparan previamente los datos. Por esta razón, es necesario explorar nuestros datos y prepararlos de tal forma que los atributos y valores que incluyamos no afecten negativamente a los resultados del modelado. Para ello, existe una serie de técnicas que nos permiten analizar las relaciones entre nuestras variables, además de con ellas mismas, así como derivar nuevos atributos a partir de los originales, si fuera necesario. A esta área se le conoce en el mundo de AA como "Ingeniería de Características" (Feature Engineering en inglés). A continuación, se detallan las técnicas que se han utilizado en este proyecto para examinar estas dependencias.

2.2.1 Correlación de Pearson

El coeficiente de correlación de Pearson [9], [14]-[17], también llamado coeficiente de correlación del producto-momento de Pearson, indica la relación lineal que existe entre dos variables continuas. Dos variables están correlacionadas linealmente si el movimiento de una corresponde de forma proporcional al movimiento de la otra, ya sea en la misma dirección o en la contraria. En otras palabras, si una crece o decrece, la otra también lo hará de forma proporcional.

El coeficiente de correlación de Pearson se denota con una *r* y se calcula de la siguiente manera:

$$r = \frac{s_{xy}}{s_x s_y}$$

Donde:

s_{xy} es la covarianza entre las variables x e y,

s_x y s_y son las desviaciones estándar de las variables x e y, respectivamente.

El valor del coeficiente de correlación de Pearson se encuentra en el rango [-1, 1] y, dependiendo del sentido y de qué tan fuerte sea la relación lineal entre las variables, tenemos los siguientes casos:

- ***r* > 0**: Significa que existe una correlación lineal positiva, es decir, la variable x e y están directamente relacionadas. Si una crece, la otra también crece de forma proporcional; si una decrece, la otra también decrece de forma proporcional. Cuanto más cerca a 1 se encuentre el coeficiente, mayor será la correlación, y el cambio de una con respecto a la otra será más constante. Cuando *r* es 1, se habla de una correlación positiva perfecta.
- ***r* < 0**: Significa que existe una correlación lineal negativa, es decir, las variables x e y están inversamente relacionadas. Si una crece, la otra decrece de forma proporcional, y viceversa. Cuanto más se acerca se encuentre el valor a -1, mayor será la correlación y, por ende, los cambios más constantes. Cuando *r* es -1, se habla de una correlación negativa perfecta.
- ***r* = 0**: Significa que no existe una relación lineal entre las dos variables, es decir, los puntos están distribuidos de tal forma que no es posible

determinar el sentido de la correlación entre x e y. Cuanto más cercano se encuentra *r* a 0, menor correlación lineal entre variables. Si se da este caso, aunque no exista una relación lineal entre las variables, es posible que existan relaciones no lineales.

Teniendo un conjunto de observaciones de dos variables, la correlación de Pearson busca trazar la línea que mejor se ajuste a los puntos correspondientes a estas observaciones. El coeficiente indica el grado de proximidad de estas observaciones a la recta de mejor ajuste, es decir, con qué exactitud describe esta recta el conjunto general de puntos.

En la **Figura 2**, la **Figura 3** y la **Figura 4** [14] se muestra gráficamente lo explicado.

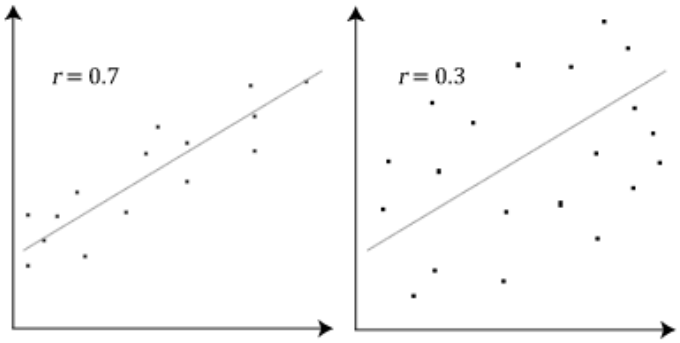

Figura 2. Correlación lineal positiva

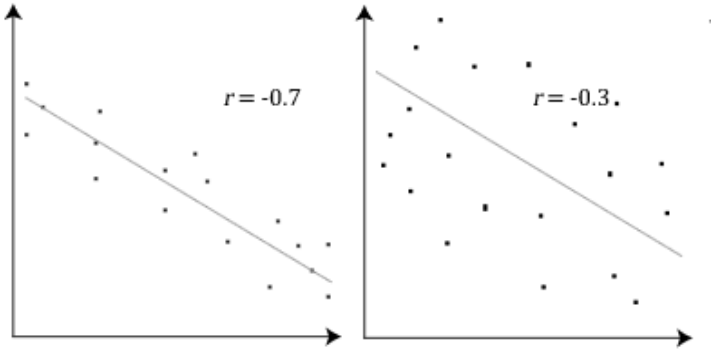

Figura 3. Correlación lineal negativa

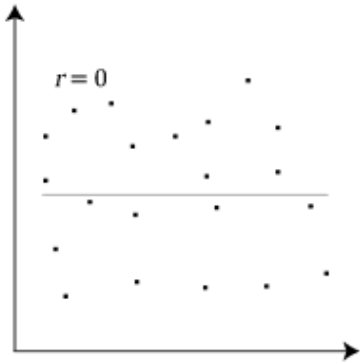

Figura 4. Correlación lineal nula

2.2.2 Regresión lineal múltiple

La Regresión Lineal Múltiple (MLR) [9], [18], [19] se trata de una técnica estadística con la que se puede generar un modelo de estimación a partir de la relación lineal que existe entre la variable a predecir (también llamada target), y las variables independientes (también llamadas descriptoras). Esta técnica es una extensión de la Regresión Lineal Simple [20], en la que se predice el valor de la variable dependiente a partir del valor de una única variable independiente.

Esta relación entre la variable target (*Y*) y las variables descriptoras viene dada por la siguiente expresión:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + e$$

Donde:
β_0 es la ordenada en el origen,
β_i con i desde 1 hasta n son los coeficientes de cada una de las n variables, descriptoras, es decir, el grado de influencia de cada variable descriptora sobre Y,
x_i es el valor de cada variable descriptora para esa observación,
e es el error del modelo.

Como esta técnica sirve para determinar el grado de influencia de una serie de variables sobre una en particular, también se puede usar para estudiar la multicolinealidad entre variables. Precisamente, esa será la aplicación que se le dará en este proyecto, ya que debemos saber qué variables tienen una dependencia lineal significativa entre ellas para poder tratar este problema y evitar que afecte a la generación del modelo.

Para ello, los modelos de MLR se analizarán utilizando dos métricas:

- **R^2 Ajustado (Adjusted R^2):** Se trata de un valor entre 0 y 1, que indica el grado de variación en la variable target que puede ser explicada por las variables descriptoras. Un valor de 0 significa no hay relación lineal entre las variables independientes y la dependiente, mientras que un valor cercano a 1 significa que existe una fuerte relación lineal.
- **Valor-p (p-value):** Se trata de un valor entre 0 y 1, que indica si la regresión es estadísticamente significativa, es decir, si existe correlación lineal entre las variables independientes y la dependiente. Para explicar esto debemos saber que se plantean dos hipótesis: la hipótesis a probar (hay correlación entre variables independientes y dependiente) y la hipótesis nula (no hay correlación entre variables independiente y dependiente). Para que la regresión sea estadísticamente significativa, el valor-p debe ser menor que 0,05. En ese caso, podemos negar la hipótesis nula con menos de un 5% de probabilidad de equivocarnos, esto es, que se puede aceptar la hipótesis de que sí hay correlación entre la variable target y las descriptoras.

2.2.3 Correlación de Spearman

El coeficiente de correlación de Spearman [9], [20]-[23], también llamado coeficiente de correlación de rangos de Spearman, indica la dirección y el grado de correlación entre dos variables. La diferencia con el coeficiente de correlación de Pearson radica en que mide la correlación entre variables clasificadas (rangos) y la relación monótona entre variables.

Una relación es monótona si, cuando una variable se mueve en una dirección, la otra se mueve siempre en la misma dirección o en la contraria, sin cambiar de dirección. Además, esta relación no es necesariamente lineal, como en el caso del coeficiente de Pearson. Es decir, si una variable crece, la otra puede crecer o decrecer a un ritmo no constante. La **Figura 5**, **Figura 6** y **Figura 7** muestran esto gráficamente.

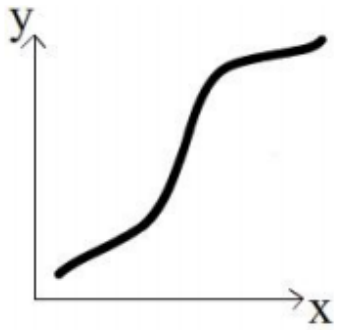

Figura 5. Relación monótona creciente

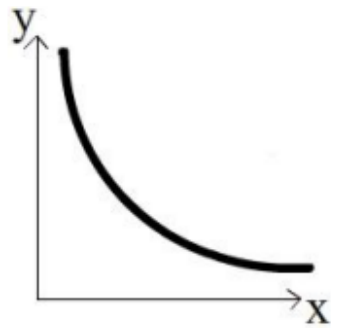

Figura 6. Relación monótona decreciente

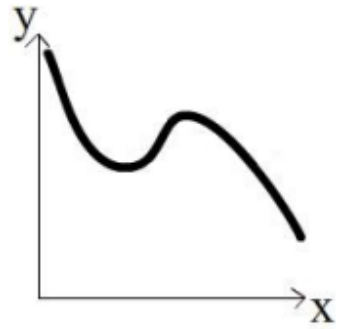

Figura 7. Relación no monótona

El coeficiente de correlación de Spearman se denota por r_s y se calcula mediante la siguiente expresión:

$$r_s = 1 - \frac{6 \sum_i d_i^2}{n(n^2 - 1)}$$

Donde:
d_i es la diferencia de rangos entre x e y en la observación i,
n es el número total de observaciones.

De forma similar al coeficiente de Pearson, el coeficiente de correlación de Spearman puede tomar los siguientes valores:

- $r_s > 0$: Significa que existe una correlación de rangos positiva o una correlación monótona creciente entre variables. Si una variable crece, la otra nunca decrecerá (y viceversa), pero no necesariamente será una relación lineal. Cuanto más cercano es el valor a 1, más fuerte es la relación monótona creciente entre variables.
- $r_s < 0$: Significa que existe una correlación de rangos negativa o una correlación monótona decreciente. Si una variable crece, la otra nunca crecerá (y viceversa), pero no necesariamente será una relación lineal. Cuanto más cercano es el valor a -1, más fuerte es la relación monótona decreciente entre variables.
- $r_s = 0$: Significa que no existe correlación de rangos o correlación monótona entre variables. Sin embargo, esto no quiere decir que no exista otro tipo de relación entre las variables.

El coeficiente de correlación de Spearman puede ayudarnos a determinar si existe una correlación entre variables que no haya sido detectada por el coeficiente de correlación de Pearson por no ser una relación lineal. Por ejemplo, si se da una relación curvilínea entre dos variables, es posible que tengamos un r_s igual o cercana a 1, pero un r menor, ya que no es posible ajustar una recta lo suficientemente precisa, como se muestra en la **Figura 8**.

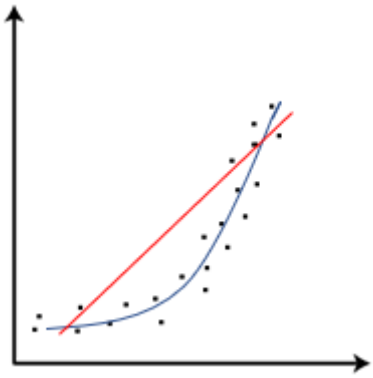

Figura 8. Correlación de Pearson frente a correlación de Spearman

2.2.4 Análisis de Componentes Principales (PCA)

En el proceso de generación de un modelo de predicción, tener una cantidad de variables elevada puede afectar negativamente, tanto a los resultados de las predicciones, como al rendimiento del modelo, ya que se consumiría más recursos. Ante este problema, se realiza una reducción de dimensionalidad, donde se pueden aplicar dos enfoques: la selección de características y la extracción de características.

La primera de ellas consiste en descartar las variables que consideremos menos relevantes y quedarnos con aquellas que creemos que más aportan al modelo. Esta técnica puede estar apoyada en las correlaciones explicadas en los puntos anteriores.

Por otro lado, el segundo enfoque, la extracción de características, consiste en generar un nuevo conjunto de variables a partir de las variables originales, como se muestra en la **Figura 9**. Para lograr esto, se utiliza la técnica de Análisis de Componentes Principales, del inglés Principal Component Analysis (PCA) [24]-[29].

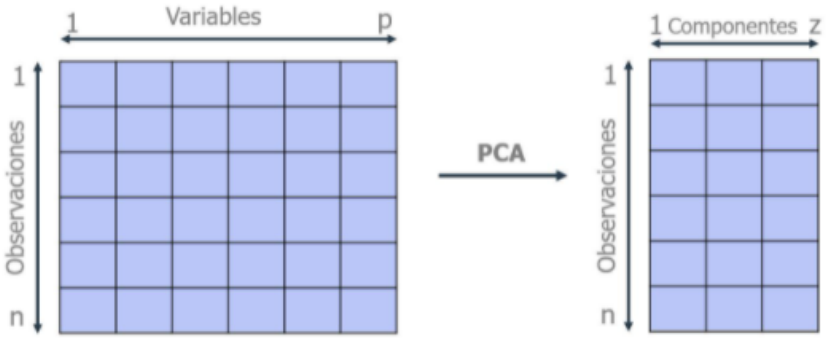

Figura 9. Extracción de características con PCA

PCA funciona de tal forma que genera un nuevo conjunto de variables, llamados Componentes Principales, que contienen información sobre cada una de las variables iniciales. Estos Componentes Principales en realidad son ejes perpendiculares entre sí que explican en mayor medida o menor medida la variabilidad del conjunto de datos inicial, como se ve en la **Figura 10**. Por esta razón, los Componentes Principales resultantes no están correlacionados entre ellos y están ordenados por importancia, es decir, el primer componente será el que mejor represente la información de las variables iniciales y el que mejor explique su variabilidad, el segundo componente será el que mejor lo haga después del primero, y así sucesivamente.

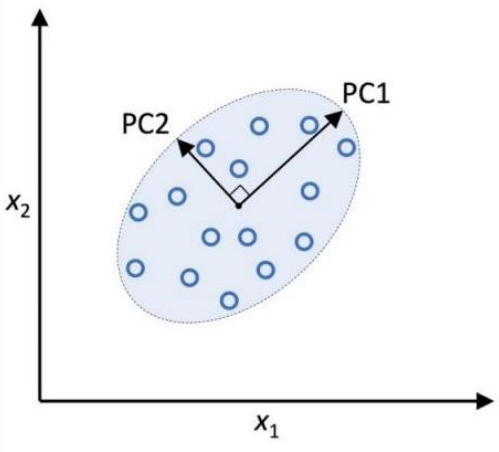

Figura 10. Componentes Principales

Los Componentes Principales son obtenidos gracias a un conjunto de autovectores y autovalores. Los autovectores representan las direcciones del componente, y los autovalores las importancias que se ha mencionado anteriormente. El proceso para obtener estos componentes es el siguiente: a partir de los datos estandarizados, se construye una matriz de covarianza, de la que se obtienen los autovectores y los autovalores. Los autovectores son ordenados de acuerdo a sus autovalores correspondientes, para luego seleccionar los *z* primeros, y construir una matriz de proyección con ellos. De esta forma, se obtiene el nuevo conjunto de datos, siendo los componentes principales los nuevos atributos.

Una vez aplicado PCA, podemos elegir con qué componentes quedarnos, ya que tenemos la certeza de que no perderemos información valiosa de ninguna variable inicial. Existen diferentes métodos para seleccionar los componentes con los que trabajar. Uno de ellos consiste en obtener la tasa de variabilidad explicada acumulada para cada componente y elegir aquellos con los que se alcance un umbral establecido. Otra forma interesante de decidir esto consiste en graficar un scree-plot de las variabilidades explicadas de cada componente y, partir de ella, descartar los componentes a partir del cual no se da una variabilidad explicada significativa (método del codo).

2.2.5 Autocorrelación

La autocorrelación [9], [30]-[33], también llamada correlación serial, es una medida estadística que sirve para determinar la correlación lineal que existe entre el valor de una observación y las observaciones en momentos previos de una variable de una serie temporal. En otras palabras, la autocorrelación indica qué tan relacionada linealmente y en qué sentido (negativa o positivamente), se encuentra una variable en un momento *t* con sus observaciones en momentos *t-i*, donde *i* es el retraso en el tiempo para esa variable de la serie temporal.

Existen dos tipos de autocorrelación:

- **Autocorrelación simple:** Indica la relación lineal entre el valor de una observación de una serie temporal y los valores de las observaciones en *i* retrasos, teniendo en cuenta las correlaciones con los retrasos intermedios.

Es decir, teniendo la variable *x*, las correlaciones entre x_t y x_{t-i} se ve afectada por las correlaciones entre x_t y x_{t-1} hasta x_{t-i-1}.

- **Autocorrelación parcial:** También indica la relación lineal entre observaciones de una variable de una serie temporal separadas por *i* intervalos de tiempo, pero en este caso no se considera las correlaciones con los intervalos intermedios. Es decir, la correlación entre x_i y x_{t-i} depende únicamente de x_t y x_{t-i}.

Las autocorrelaciones toman un valor en el rango de [-1, 1], donde el signo indica si la autocorrelación es directa (positivo) o inversa (negativo), y el número indica el grado de correlación (cuanto más cercana a 0, menor es la autocorrelación). Además, para saber si las autocorrelaciones son estadísticamente significativas, normalmente se utilizan funciones que crean gráficas con intervalo de confianza de 0,05. Las autocorrelaciones dentro de este intervalo no son significativas.

El cálculo de la autocorrelación es útil para determinar la ventana de tiempo con la que se va a realizar la predicción, es decir, el número de observaciones previas a tener en cuenta para predecir una determinada salida. En el siguiente punto se entrará en este aspecto con más detalle, ya que la autocorrelación es un aspecto fundamental para la generación de modelos ARIMA.

2.2.6 Modelo Autorregresivo-Integrado de Medias Móviles (ARIMA)

El modelo Autorregresivo-Integrado de Medidas Móviles, del inglés Autoregressive Integrated Moving Average (ARIMA) [9], [34]-[38] pretende realizar la predicción de los valores de una serie temporal en función de los valores en instantes anteriores, teniendo en cuenta que esto se realiza sobre una única variable.

Un modelo ARIMA es una combinación de los modelos Autorregresivo (AR) y de Medias Móviles (MA). La letra I, de Integrado, significa que las observaciones son diferenciadas, ya que el modelo necesita trabajar con series estacionarias. Por esta razón, el modelo recibe tres parámetros, uno por cada parte de la que está compuesto:

- **p**: Indica el número de observaciones a tener en cuenta para la parte correspondiente al modelo Autorregresivo (AR). El modelo Autorregresivo [39] utiliza valores de las observaciones pasadas que se indiquen para ajustar una recta de regresión lineal con la cual predecir valores futuros. Este número de observaciones pasadas a tener en cuenta se decide gracias a la autocorrelación parcial, vista en el punto anterior. Sabiendo las autocorrelaciones parciales en cada intervalo de tiempo, se elige como valor p el instante hasta el cual la autocorrelación es estadísticamente significativa, es decir, se encuentra en el límite del intervalo de confianza del 5%.
- **d**: Indica el número de diferenciaciones necesarias para convertir la serie temporal en estacionaria [40]. Una serie temporal es estacionaria cuando no tiene tendencia ni estacionalidad. La tendencia es la inclinación del

conjunto de datos en una dirección, ya sea creciente o decreciente, como se observa en la **Figura 11**. La estacionalidad, por otra parte, indica que el conjunto de datos presenta ciclos, como se observa en la **Figura 12**. Para un modelo ARIMA es fundamental que la serie sea estacionaria, ya que la presencia de tendencia y estacionalidad influyen en la precisión del modelo. Existen pruebas para determinar si una serie es estacionaria o no, como el test Dickey-Fuller, donde un valor-p menor que 0,05 permite negar la hipótesis nula de que la serie es no-estacionaria y, por lo tanto, se puede asumir que es estacionaria.

- **q**: Indica el número de observaciones previas que se tendrán en cuenta para el modelo de Medias Móviles (MA). Un modelo de Medias Móviles [41] es similar a un modelo Autorregresivo, con la diferencia de que, en lugar de usar observaciones pasadas, utiliza los errores de predicciones pasadas. El valor q se elige a partir de la autocorrelación simple. Teniendo las autocorrelaciones simples de una serie temporal para cada intervalo de tiempo, se toma el instante hasta el cual la autocorrelación es estadísticamente significativa según el intervalo de confianza del 5%, tal y como se hacía con la autocorrelación parcial.

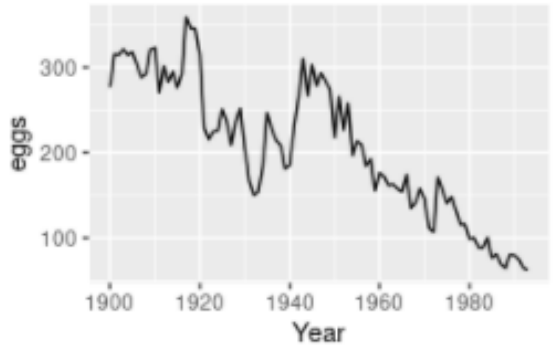

Figura 11. Tendencia

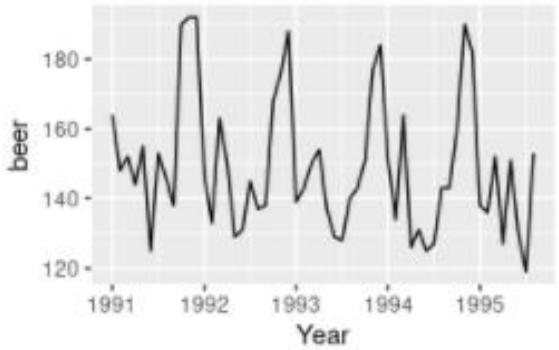

Figura 12. Estacionalidad

El modelo ARIMA se representa mediante la siguiente expresión, donde se puede distinguir cada parte del modelo:

$$Y_t = c + \phi_1 Y_{t-1} + \ldots + \phi_p Y_{t-p} + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Donde:

Y_t es el valor a predecir en el instante *t*,

c es la ordenada en el origen,
$\phi_1 Y_{t-1} + \ldots + \phi_p Y_{t-p}$ es la parte del modelo Autorregresivo,
$\theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$ es la parte del modelo de Medias Móviles,
ε_t es el error en el instante *t*.

En este proyecto, nos apoyaremos en los modelos ARIMA para determinar la relación temporal de cada variable, y de esta forma establecer una ventana de tiempo para la fase de modelado.

2.3 Modelado: *Long Short-Term Memory* (LSTM)

Los modelos *Long Short-Term Memory* (LSTM) son un tipo de modelos de predicción que utilizan, como su nombre indica, redes neuronales LSTM. Estas son, a su vez, un tipo de Redes Neuronales Recurrentes (RNN). Pero, ¿qué son las RNN?

Una Red Neuronal Recurrente, o Recurrent Neural Network (RNN) [42]-[45], es aquella que, a diferencia de las redes neuronales tradicionales, es capaz de procesar secuencias de datos, es decir, datos que necesitan ser interpretados en conjunto y en un orden específico para tener significado. Esto es posible gracias a que las RNN cuentan con una arquitectura que les permite, en cada instante de tiempo, recibir la entrada correspondiente a ese instante, además del valor de la activación en el instante anterior. Dicho de otra forma, para el instante de tiempo *t*, las entradas para la red neuronal serán el dato de entrada en *t* y la función de activación aplicada a la salida de *t-1*. Esto se ve representado de forma sencilla con una neurona en la **Figura 13**.

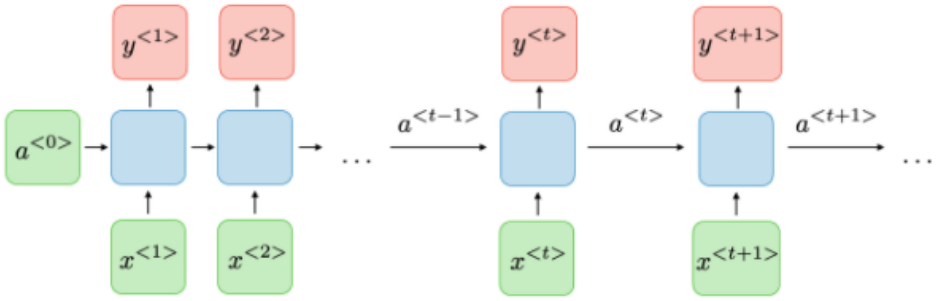

Figura 13. Arquitectura RNN con una neurona

La alimentación que tiene la red desde el instante de tiempo previo le permite tener cierta "memoria", ya que está recibiendo información de pasos anteriores. Esto ocurre gracias a la existencia de una celda de memoria, que es la que conserva el estado a lo largo del tiempo. Por otro lado, las RNN aplican *Backpropagation* a través del tiempo (BPTT), es decir, la propagación del error hacia atrás, permitiendo así actualizar los pesos para cada instante en el proceso de entrenamiento.

Otro aspecto a tener en cuenta cuando se trabaja con RNN, es que se pueden dar los problemas de "*Exploding Gradient*" y "*Vanishing Gradient*". El primero se da cuando el gradiente, es decir, la pendiente de la función de pérdida, es demasiado alta, lo que puede producir modelos inestables. Esto se puede solucionar reduciendo el número de neuronas de la capa oculta. El segundo problema se da cuando el gradiente se reduce demasiado, de forma que, si llega a cero, ya no se produce aprendizaje. Esta cuestión se soluciona con el uso de puertas, como en el caso de las redes LSTM.

Gracias a la arquitectura de las RNN, estas pueden tratar datos ordenados en el tiempo, es decir, series temporales. Esta ventaja ha causado que las RNN sean ampliamente utilizadas en *Deep Learning*, campo del AA. Algunas de las

aplicaciones más comunes de las RNN son el análisis de sentimientos, la generación de música, o las conversiones de voz a texto (y viceversa).

Sin embargo, la "memoria" de las RNN tradicionales es en realidad una memoria a corto plazo, ya que, al ir transmitiendo la información a través del tiempo, el efecto del estado de un instante será muy pequeño en un instante posterior muy alejado. Es decir, el estado en el instante *t-i* influirá muy poco en la salida del instante *t* si *i* es muy grande.

Ante este inconveniente, surgen las redes *Long Short-Term Memory* [45]-[48] (LSTM). Estas redes son capaces de mantener información tanto a corto, como a largo plazo, gracias la incorporación de una celda de estado *C*. Esta celda de estado se va actualizando en función del estado de la celda, la salida en el instante *t-1*, la entrada en el instante *t*, y una serie de puertas que se encargan de procesar esta información. Las puertas que determinan la información a conservar por la red en cada instante de tiempo se detallan a continuación:

- En primer lugar, se tiene una puerta de olvido (forget gate) que, recibiendo la entrada *x* en el instante *t* y la salida *h* en el instante *t-1*, decide qué información se descartará, es decir, que no formará parte del estado para ese instante. Para ello, consta de una capa con una función sigmoidal que devuelve un valor entre 0 y 1, donde 0 significa que se elimina toda la información y 1 que se conserva toda.

- En segundo lugar, se tiene una puerta de entrada, también llamada de actualización (input o update gate). Esta puerta determina qué información de la celda de estado actualizar a partir de la entrada en *x* en el instante *t* y la salida *h* en el instante *t-1*, información que pasa por una capa con función de activación sigmoidal. Por otro lado, a través de una función tangencial hiperbólica se genera la posible información a añadir al nuevo estado. Combinando estas dos salidas, se obtiene la información que pasará a la celda de estado. De esta forma, y en conjunto con el resultado de la capa de olvido, se actualiza la celda de estado desechando valores que ya no aportan a predicciones futuras y reemplazándolos por nueva información relevante.

- Por último, se tiene la puerta de salida (output gate), que generará la salida *h* para el instante *t* a partir de la información de la celda de estado. Pero no se vuelca toda la información de la celda de estado en la salida, sino que se realiza una selección. Para ello, se procesa la entrada *x* en el instante *t* y la salida *h* en el instante *t-1* con una capa con la función sigmoidal como función de activación. Ese resultado se fusiona con la información de la celda de estado pasada por la función tangente hiperbólica. De esta forma, se obtiene la salida *h* en el instante *t*, que contiene la porción de información de la celda de estado que se ha considerado importante para utilizar en el instante siguiente.

En la **Figura 14** [49] se observa la arquitectura de una red LSTM, con cada una de las partes explicadas anteriormente.

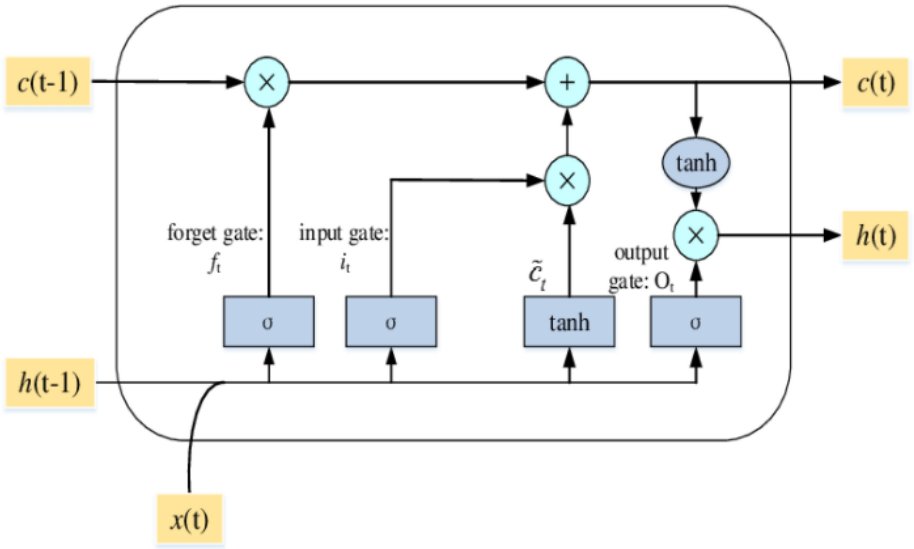

Figura 14. Arquitectura de una neurona de una red LSTM

De esta forma, el comportamiento de una red neuronal LSTM intenta replicar el aprendizaje humano, donde para procesar información que requiere de memoria a largo plazo no nos quedamos con todos los datos, sino sólo con los más relevantes, que nos serán útiles para entender lo que estamos tratando.

Esta capacidad de las redes LSTM de retener información a largo plazo, hacen a este tipo de redes la opción idónea para crear los modelos de predicción de este proyecto. Esto se debe a que los valores en las series temporales de consumo energético están correlacionados con valores en intervalos de tiempo pasados, que pueden ser días, pero también pueden ser semanas, meses o años.

Por ejemplo, en el contexto del consumo de energía eléctrica en función del consumo de sus dispositivos se puede dar el siguiente caso: Si queremos predecir el consumo total del día lunes, la puerta de olvido se encargará de desechar información sobre el consumo de dispositivos del miércoles pasado, pues deja de ser relevante. Por otro lado, la puerta de entrada añadirá datos sobre el consumo de dispositivos del lunes, ya que esta información sí es útil para predecir el consumo total de este día. De esta forma, la puerta de salida podría determinar el consumo total del día lunes combinando los datos de entrada con los datos seleccionados de la celda de estado, como el consumo del lunes pasado.

Igual que con otros tipos de redes neuronales, al realizar el modelado con redes LSTM es necesario tener en cuenta tres parámetros: el número de neuronas, el número de épocas de entrenamiento (epochs) y el tamaño de lote (batch size). El ajuste de estos parámetros es fundamental para conseguir un modelo preciso y evitar los problemas de *underfitting* y *overfitting*: El primero de ellos se da cuando la red no aprende lo suficiente y, por lo tanto, no se obtienen los resultados esperados. El segundo, por el contrario, se da cuando la red deja de aprender y empieza a memorizar, por lo que el error en la etapa de entrenamiento será bajo, pero en la fase de prueba será alto, y las predicciones no serán precisas.

2.4 Evaluación de modelos

Para evaluar la precisión de los modelos generados es necesario valorar qué tanto se ajustan las predicciones a los valores que esperamos obtener, es decir, los valores reales. Esto nos servirá para poder ajustar nuestros modelos y obtener estimaciones lo más precisas posibles. Para ello, utilizamos las siguientes métricas de calidad:

2.4.1 Error Cuadrático Medio (RMSE)

El Error Cuadrático Medio o Root Mean Square Error (RMSE) [50]-[53] mide qué tan alejados se encuentran los valores predichos de los valores reales. A la distancia entre cada punto de valor real y de predicción se le llama residuo o error, como se observa en la **Figura 15**. Para el cálculo del RMSE, se suman los residuos de cada observación, estando éstos elevados al cuadrado para evitar que los negativos contrarresten a los positivos. Posteriormente, se divide entre el número de observaciones, y se aplica la raíz cuadrada a este resultado para volver a la escala de los valores originales.

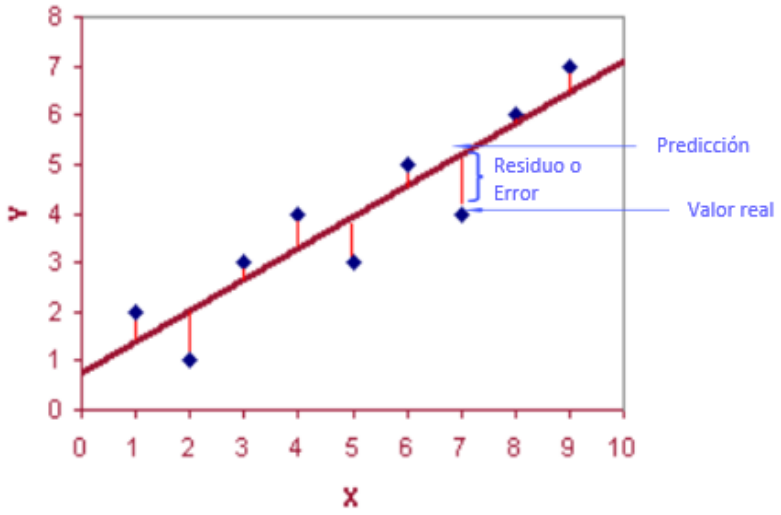

Figura 15. Residuo o Error

La expresión correspondiente al Error Cuadrático Medio es la siguiente:

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

Donde:
N es el número de observaciones,
$\hat{y}_i$ es el valor de la predicción para la observación *i*,
y es valor real para la observación *i*.

Esta métrica da una idea general de qué tan bueno es nuestro modelo, gracias a que el valor del RMSE se encuentra en las mismas unidades que los valores de las

observaciones. De esta forma, con el RMSE sabemos, de media, cuánto se desvían nuestras predicciones de los valores reales.

2.4.2 Error Porcentual Absoluto Medio (MAPE)

El Error Porcentual Absoluto Medio o Mean Absolute Percentage Error (MAPE) [54]-[56] mide el porcentaje de precisión en un modelo de predicción. Para ello, calcula la diferencia entre el valor real y el valor predicho, y la divide por el valor real, de ahí que sea un error porcentual. Además, a este resultado se le aplica el valor absoluto para evitar que las diferencias negativas contrarresten a las positivas. Este proceso se repite para cada observación, de forma que se suman todos los valores resultantes y se divide esto por el número total de observaciones.

La expresión correspondiente al MAPE es la siguiente:

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

Donde:
N es el número de observaciones,
y es valor real para la observación i,
$\hat{y}_i$ es el valor de la predicción para la observación i.

Esta métrica nos ayuda a determinar el grado de exactitud con la que nuestro modelo es capaz de realizar las predicciones de forma general. En otras palabras, nos indica en qué porcentaje los valores predichos son distintos de los valores reales.

2.4.3 R cuadrado (R^2)

R cuadrado o R squared (R^2), también llamado coeficiente de determinación [53], [57]-[59], mide en qué grado la variabilidad en las variables descriptoras puede explicar la variabilidad en la variable a predecir. En otras palabras, esta métrica indica el porcentaje de valores predichos que coinciden con los valores reales de las observaciones. Para ello, se calcula la proporción entre la variabilidad explicada y la variabilidad total. Otra forma de verlo es como el valor complementario a la variabilidad no explicada, es decir, la división entre la variabilidad no explicada y la variabilidad total, restando este resultado a 1.

Para realizar esto, debemos tener en cuenta que la variabilidad no explicada se trata de la suma de cuadrados de los residuos. Esto quiere decir que se calcula el cuadrado de la diferencia entre el valor real y el valor predicho de cada observación, para luego realizar una suma de estos resultados. Por otra parte, la variabilidad total se trata de la suma total de cuadrados. Se calcula de la misma manera que la variabilidad no explicada, pero en lugar de realizar la diferencia entre valores reales y valores predichos, se realiza la diferencia de valores reales y la media de éstos.

El coeficiente de determinación, R^2, se ve representado en las siguientes expresiones:

$$R^2 = \frac{Variabilidad\ explicada}{Variabilidad\ total}$$

$$R^2 = 1 - \frac{Variabilidad\ no\ explicada}{Variabilidad\ total}$$

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

Donde:
N es el número de observaciones,
y es valor real para la observación *i*,
$\hat{y}_i$ es el valor de la predicción para la observación *i*,
$\bar{y}$ la media de los valores reales.

De esta forma, R^2 nos ayuda a evaluar la calidad de nuestro modelo de acuerdo a cuánto afecta el cambio de las variables independientes en el cambio de la variable dependiente, y qué porcentaje de acierto obtendremos. Valores cercanos a 1 quiere decir que gran cantidad de valores serán predichos de forma precisa, mientras que valores cercanos a 0 indican que muy pocas predicciones dan el valor esperado.

Capítulo 3.
Desarrollo práctico

En este capítulo se presenta el proceso llevado a cabo para obtener modelos de predicción del consumo de energía. Para esto, el proceso se divide en cuatro secciones. De forma similar al capítulo 3, en este caso se tiene una primera sección de entendimiento y preparación de datos, en la que se detalla el análisis realizado sobre las variables del conjunto de datos seleccionado. Para ello, se aplican las técnicas vistas en la sección correspondiente del capítulo anterior, con el objetivo de dejar el modelo preparado para poder construir con él modelos de predicción, y obtener valores como la ventana temporal. En la segunda sección, ya se procede a la generación de los modelos LSTM, además de a su evaluación, puesto que es necesario valorar la precisión de cada modelo para conseguir el siguiente mediante el ajuste de sus parámetros. En este caso, se genera un modelo inicial a partir del cual se crean tres grupos de modelos, según las variables descriptoras que se consideren. En la tercera sección, se realiza una comparación entre los modelos de cada grupo. Por último, en la cuarta sección se interpretan los resultados obtenidos y, teniendo en cuenta cuáles son los mejores de cada grupo, se propone uno de ellos como la opción óptima.

3.1 Entendimiento y preparación de datos

El primer paso del proceso de un proyecto de AA es, como se ha visto anteriormente, comprender nuestros datos y prepararlos para poder usarlos en la fase de modelado. Para ello, debemos seleccionar el conjunto de datos con el que trabajaremos.

Previamente, se ha realizado una búsqueda de candidatos a nuestro conjunto de datos. Los datasets de interés son aquellos que proporcionan el consumo total de energía de viviendas, apartamentos o zonas, además del consumo de energía de los aparatos y dispositivos de los que disponen. Tras explorar los conjuntos de datos ofrecidos en fuentes abiertas [60]-[66], y examinar sus atributos y observaciones, se eligió el data set Raw_Data.csv [65].

Este conjunto de datos es una serie temporal en la que se registra el consumo energético del Ala B Este del edificio *Research Support Facility* (RSF) del

Laboratorio Nacional de Energía Renovable de Estados Unidos. Cuenta con 34 atributos:

- *Date*: Representa la fecha en que se toma cada muestra, de tipo Fecha (Date*)*, y en formato YYYY:MM:DD.

- *Month*: Representa el mes en que se toma cada muestra, de tipo entero (Integer).

- *Day*: Representa el día de la semana (*Sunday-Monday*) en que se toma cada muestra, de tipo String.

- *Time*: Representa el momento del día en que se toma la muestra, de tipo tiempo (Time), y en formato hh:mm.

- *Hour* y *Minute*: Representan la hora y el minuto de la toma de la muestra, respectivamente. Son de tipo entero (Integer).

- *Skyspark*: Representa el consumo total de energía para cada observación, de tipo Double, y en kilovatios (kW). Esta será nuestra variable target o dependiente.

- *AV.Controller, Coffee.Maker, Copier, Desktop.Server, Headset, Lamp, Laptop, Microwave, Monitor, Phone.Charger, Printer, Projector, Toaster.Oven, TV, Video.Conference.Camera, Water.Boiler, Conference.Podium.Equip, Exercise.Fans, Auto.Door.Opener, Microphone.Charger, Treadmill.and.Elliptical, AV.Controller.1, Refrigerator, Central.Monitoring.Station.TVs* y *Transformer.Loss*: Representan el consumo energético de cada uno de estos aparatos en cada observación, de tipo Double, y en kilovatios (kW). Estas serán nuestras variables descriptoras o independientes.

- *Prediction*: Representa el consumo de energía predicho para cada observación. Esta variable no será utilizada en este proyecto, ya que la predicción la realizaremos nosotros.

- *Difference*: Representa la diferencia entre el valor real de consumo energético total y el valor predicho para cada observación. Por lo tanto, esta variable tampoco será utilizada.

Además, nuestro data set cuenta con 26496 observaciones, recogidas cada 5 minutos, desde el 01-10-2017 a las 00:00 hasta el 31-12-2017 a las 23:55.

Una vez reconocidas las variables de nuestro conjunto de datos, pasamos a cargarlo uniendo los valores de los atributos *Date* y *Time*. De esta forma, podremos trabajar posteriormente con él como una serie temporal. En la **Figura 16** se muestran las cinco primeras y cinco últimas observaciones de nuestra serie temporal (por motivos de espacio no se muestran todas las columnas).

	Date_Time	Month	Day	Hour	Minute	Skyspark	AV.Controller	Coffee.Maker	Copier	Desktop.Server	...	Exercise.Fans	Auto.Door.Opener	Microphone.Charger
0	2017-10-01 00:00:00	10	Sunday	0	0	9.980	0.000000	0.001030	0.018790	0.129330	...	0	0	0
1	2017-10-01 00:05:00	10	Sunday	0	5	10.002	0.000000	0.001027	0.018790	0.129467	...	0	0	0
2	2017-10-01 00:10:00	10	Sunday	0	10	9.808	0.000000	0.001033	0.018780	0.129580	...	0	0	0
3	2017-10-01 00:15:00	10	Sunday	0	15	10.018	0.000000	0.001034	0.018780	0.130941	...	0	0	0
4	2017-10-01 00:20:00	10	Sunday	0	20	10.390	0.000000	0.001033	0.018793	0.130210	...	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26491	2017-12-31 23:35:00	12	Sunday	23	35	9.448	0.756163	0.001020	0.018710	0.130887	...	0	0	0
26492	2017-12-31 23:40:00	12	Sunday	23	40	9.760	0.756510	0.001023	0.018710	0.129637	...	0	0	0
26493	2017-12-31 23:45:00	12	Sunday	23	45	10.588	0.756377	0.001020	0.018707	0.128597	...	0	0	0
26494	2017-12-31 23:50:00	12	Sunday	23	50	11.178	0.756090	0.001020	0.018710	0.127577	...	0	0	0
26495	2017-12-31 23:55:00	12	Sunday	23	55	10.358	0.756293	0.001020	0.018717	0.130723	...	0	0	0

26496 rows × 33 columns

Figura 16. Carga inicial del dataset Raw_Data.csv

En cuanto a las variables *Month*, *Day*, *Hour* y *Minute*, no es necesario mantenerlas, ya que esta información se encuentra condensada en la variable *Date_Time*, resultado de unir los atributos iniciales *Date* y *Time*. Esta variable se establecerá como índice para poder trabajar con el data set de forma más cómoda.

También se observa que las variables *Conference.Podium.Equip, Exercise.Fans, Auto.Door.Opener, Microphone.Charger, Treadmill.and.Elliptical, AV.Controller.1, Refrigerator, Central.Monitoring.Station.TVs* y *Transformer.Loss*, es decir, las 9 últimas columnas, no aportan información, ya que todos sus valores son 0. Por lo tanto, procedemos a eliminarlas.

Por otro lado, las variables *Prediction* y *Difference* tienen valores NaN, ya que obtener esta predicción es nuestro objetivo. Estas variables también son eliminadas.

En la **Figura 17** se muestra el conjunto de datos resultante.

Date_Time	Skyspark	AV.Controller	Coffee.Maker	Copier	Desktop.Server	Headset	Lamp	Laptop	Microwave
2017-10-01 00:00:00	9.980	0.000000	0.001030	0.018790	0.129330	0.005530	0.014113	0.006767	0.009417
2017-10-01 00:05:00	10.002	0.000000	0.001027	0.018790	0.129467	0.005526	0.014122	0.008408	0.009430
2017-10-01 00:10:00	9.808	0.000000	0.001033	0.018780	0.129580	0.005520	0.014117	0.006767	0.009434
2017-10-01 00:15:00	10.018	0.000000	0.001034	0.018780	0.130941	0.005526	0.014126	0.007718	0.009437
2017-10-01 00:20:00	10.390	0.000000	0.001033	0.018793	0.130210	0.005530	0.014133	0.007914	0.009453
...	...	...	...	...	...	...	...	...	...
2017-12-31 23:35:00	9.448	0.756163	0.001020	0.018710	0.130887	0.005623	0.012000	0.079770	0.028673
2017-12-31 23:40:00	9.760	0.756510	0.001023	0.018710	0.129637	0.005537	0.011983	0.079760	0.028680
2017-12-31 23:45:00	10.588	0.756377	0.001020	0.018707	0.128597	0.005580	0.011993	0.079860	0.028693
2017-12-31 23:50:00	11.178	0.756090	0.001020	0.018710	0.127577	0.005650	0.012007	0.079900	0.028697
2017-12-31 23:55:00	10.358	0.756293	0.001020	0.018717	0.130723	0.005633	0.011997	0.079996	0.028720

(a)

Monitor	Phone.Charger	Printer	Projector	Toaster.Oven	TV	Video.Conference.Camera	Water.Boiler
0.015855	0.000217	0.011983	0.010495	0.0	0.030557	0.011383	0.001617
0.015840	0.000204	0.011883	0.010477	0.0	0.030654	0.011419	0.001617
0.015850	0.000147	0.012010	0.010491	0.0	0.030547	0.011377	0.001614
0.015851	0.000207	0.012523	0.010483	0.0	0.030553	0.011380	0.001617
0.015846	0.000210	0.011907	0.010487	0.0	0.030553	0.011383	0.001620
...	...	...	...	...	...	...	...
0.019158	0.000140	0.011740	0.010427	0.0	0.030480	0.357267	0.001597
0.019150	0.000157	0.011723	0.010443	0.0	0.030480	0.358143	0.001603
0.019127	0.000133	0.011732	0.010423	0.0	0.030473	0.357627	0.001603
0.019155	0.000137	0.011780	0.010437	0.0	0.030477	0.358017	0.001600
0.019138	0.000103	0.011958	0.010440	0.0	0.030483	0.357040	0.001600

(b)

Figura 17. Dataset con columnas eliminadas e índice

Una vez hecho esto, podemos pasar a estudiar la relación entre las variables de nuestro conjunto de datos. Para ello, nos apoyaremos en el coeficiente de correlación de Pearson, en modelos de regresión lineal múltiple, y en el coeficiente de correlación de Spearman, explicados en el Capítulo 3.

3.1.1 Correlación de Pearson

Para analizar las relaciones lineales existentes obtenemos los coeficientes de correlación de Pearson entre cada par de variables, representados en forma de matriz, como se muestra en la **Figura 18**.

	Skyspark	AV.Controller	Coffee.Maker	Copier	Desktop.Server	Headset	Lamp	Laptop	Microwave	Monitor	Phone.Charger	Printer	Projector	Toaster.Oven	TV	Video.Conference.C	Water.Boiler
Skyspark	1.00	0.07	0.15	0.46	0.07	-0.01	0.39	0.82	0.38	0.86	0.19	0.53	0.45	0.07	0.47	0.19	0.32
AV.Controller	0.07	1.00	0.00	0.04	0.02	0.04	-0.01	0.07	0.03	0.06	0.00	0.05	0.04	0.00	0.07	0.08	0.02
Coffee.Maker	0.15	0.00	1.00	0.09	-0.01	-0.01	0.03	0.10	0.03	0.10	0.02	0.07	0.06	0.02	0.03	0.00	0.10
Copier	0.46	0.04	0.09	1.00	0.03	0.01	0.20	0.45	0.17	0.46	0.05	0.31	0.24	0.03	0.24	0.03	0.18
Desktop.Server	0.07	0.02	-0.01	0.03	1.00	-0.01	0.10	0.06	0.03	0.07	0.12	0.04	0.03	-0.00	0.01	0.09	-0.00
Headset	-0.01	0.04	-0.01	0.01	-0.01	1.00	0.00	0.07	-0.00	0.01	-0.02	0.01	0.01	0.01	-0.05	-0.06	-0.00
Lamp	0.39	-0.01	0.03	0.20	0.10	0.00	1.00	0.43	0.13	0.43	0.14	0.25	0.24	0.02	0.23	0.03	0.11
Laptop	0.82	0.07	0.10	0.45	0.06	0.07	0.43	1.00	0.35	0.92	0.16	0.50	0.42	0.04	0.44	0.06	0.26
Microwave	0.38	0.03	0.03	0.17	0.03	-0.00	0.13	0.35	1.00	0.35	0.06	0.19	0.16	0.04	0.20	0.06	0.13
Monitor	0.86	0.06	0.10	0.46	0.07	0.01	0.43	0.92	0.35	1.00	0.18	0.53	0.45	0.04	0.50	0.10	0.27
Phone.Charger	0.19	0.00	0.02	0.05	0.12	-0.02	0.14	0.16	0.06	0.18	1.00	0.14	0.16	0.06	0.11	0.04	0.07
Printer	0.53	0.05	0.07	0.31	0.04	0.01	0.25	0.50	0.19	0.53	0.14	1.00	0.24	0.07	0.26	0.04	0.16
Projector	0.45	0.04	0.06	0.24	0.03	0.01	0.24	0.42	0.16	0.45	0.16	0.24	1.00	0.02	0.28	0.06	0.13
Toaster.Oven	0.07	0.00	0.02	0.03	-0.00	0.01	0.02	0.04	0.04	0.04	0.06	0.07	0.02	1.00	0.03	-0.01	0.05
TV	0.47	0.07	0.03	0.24	0.01	-0.05	0.23	0.44	0.20	0.50	0.11	0.26	0.28	0.03	1.00	0.06	0.13
Video.Conference.Camera	0.19	0.08	0.00	0.03	0.09	-0.06	0.03	0.06	0.06	0.10	0.04	0.04	0.06	-0.01	0.06	1.00	0.03
Water.Boiler	0.32	0.02	0.10	0.18	-0.00	-0.00	0.11	0.26	0.13	0.27	0.07	0.16	0.13	0.05	0.13	0.03	1.00

Figura 18. Matriz de coeficientes de correlación de Pearson.

Por un lado, observamos que los coeficientes de correlación de Pearson entre la variable *Skyspark* y las siguientes variables descriptoras son menores que 0.2:

- *AV.Controller*: 0.07
- *Coffee.Maker*: 0.15
- *Desktop.Server*: 0.07
- *Headset*: -0.01
- *Phone.Charger*: 0.19
- *Toaster.Oven*: 0.07
- *Video.Conference.Camera*: 0.19

Esto quiere decir que estas variables se encuentran muy poco correlacionadas con la variable target. No aportan información relevante para la generación de un modelo predictivo, ya que el comportamiento de la variable *Skyspark* no puede ser explicado por el de estas variables dependientes. Por lo tanto, podemos prescindir de ellas.

Por otro lado, las siguientes parejas de variables tienen un coeficiente mayor que 0.8:

- *Skyspark – Laptop*: 0.82
- *Skyspark – Monitor*: 0.86
- *Laptop – Monitor*: 0.92

Las variables descriptoras *Laptop* y *Monitor* se encuentran muy correlacionadas entre ellas. Además, estas mismas variables se encuentran, a su vez, muy correlacionadas con la variable target. Esto puede afectar negativamente en la etapa de modelado, por lo que debemos tenerlo en cuenta más adelante.

3.1.2 Regresión Lineal Múltiple

Otra de las técnicas que se utilizan para analizar las relaciones lineales entre variables es la generación de modelos de Regresión Lineal Múltiple. En este caso, generamos un modelo para cada una de las variables descriptoras y, de esta forma, poder detectar posibles colinealidades. Es decir, para cada variable descriptora se tiene un modelo en el que esta es la variable dependiente y las demás variables descriptoras son las variables independientes. En la **Tabla 1** se muestra el valor de la métrica R^2 ajustado y el valor-p para cada modelo generado.

Modelo	R^2 ajustado	Valor-p
AV.Controller	0.019	2.99e-99
Coffee.Maker	0.020	3.62e-106
Copier	0.230	0.00
Desktop.Server	0.031	8.48e-174
Headset	0.031	1.69e-173
Lamp	0.205	0.00
Laptop	0.848	0.00
Microwave	0.132	0.00
Monitor	0.867	0.00
Phone.Charger	0.063	0.00
Printer	0.297	0.00
Projector	0.214	0.00
Toaster.Oven	0.010	2.90e-50
TV	0.262	0.00
Video.Conference.Camera	0.036	8.81e-199
Water.Boiler	0.086	0.00

Tabla 1. Modelos de Regresión Lineal Múltiple

Se puede observar que hay dos modelos con un R^2 ajustado muy alto: los correspondientes a las variables *Laptop* y *Monitor*, de 0.848 y 0.867 respectivamente. Además, el valor-p para estos modelos es menor que 0.05. Estos resultados indican que las variables *Laptop* y *Monitor* tienen una relación lineal bastante fuerte con al menos una de las demás variables descriptoras. Esto coincide con los resultados de analizar la correlación de Pearson en el punto anterior, donde se vio que esta relación lineal en realidad estaba presente mutuamente entre esas dos variables.

3.1.3 Correlación de Spearman

Además de analizar las relaciones lineales existentes entre las variables, es necesario analizar las relaciones no lineales. Para ello, calculamos los coeficientes de correlación de Spearman, mostrados en forma de matriz en la **Figura 19**.

	Skyspark	AV.Controller	Coffee.Maker	Copier	Desktop.Server	Headset	Lamp	Laptop	Microwave	Monitor	Phone.Charger	Printer	Projector	Toaster.Oven	TV	Video.Conference.C	Water.Boiler
Skyspark	1.00	0.06	0.15	0.30	0.15	-0.06	0.32	0.56	0.06	0.62	0.17	0.51	0.26	0.15	0.40	0.18	0.04
AV.Controller	0.06	1.00	0.05	-0.03	0.08	0.13	-0.19	0.19	0.08	0.04	0.02	-0.00	0.06	0.01	-0.12	0.12	0.04
Coffee.Maker	0.15	0.05	1.00	0.34	0.09	0.00	0.15	0.09	0.67	0.11	0.28	0.16	0.22	0.04	0.12	-0.00	0.63
Copier	0.30	-0.03	0.34	1.00	0.12	-0.03	0.24	0.15	0.30	0.19	0.19	0.30	0.25	0.06	0.14	-0.00	0.30
Desktop.Server	0.15	0.08	0.09	0.12	1.00	-0.04	0.22	-0.14	0.01	-0.14	0.33	0.04	0.15	0.01	-0.13	-0.02	0.06
Headset	-0.06	0.13	0.00	-0.03	-0.04	1.00	-0.02	0.12	0.03	-0.00	-0.05	0.04	-0.11	0.02	-0.06	-0.06	-0.00
Lamp	0.32	-0.19	0.15	0.24	0.22	-0.02	1.00	0.18	0.05	0.14	0.36	0.31	0.13	0.06	0.07	-0.11	0.10
Laptop	0.56	0.19	0.09	0.15	-0.14	0.12	0.18	1.00	0.09	0.79	-0.16	0.45	0.08	0.14	0.38	0.03	0.01
Microwave	0.06	0.08	0.67	0.30	0.01	0.03	0.05	0.09	1.00	0.08	0.20	0.11	0.18	0.07	0.10	0.04	0.65
Monitor	0.62	0.04	0.11	0.19	-0.14	-0.00	0.14	0.79	0.08	1.00	-0.17	0.46	0.17	0.14	0.54	0.09	0.02
Phone.Charger	0.17	0.02	0.28	0.19	0.33	-0.05	0.36	-0.16	0.20	-0.17	1.00	0.10	0.09	0.02	-0.09	-0.05	0.24
Printer	0.51	-0.00	0.16	0.30	0.04	0.04	0.31	0.45	0.11	0.46	0.10	1.00	0.17	0.10	0.36	0.00	0.08
Projector	0.26	0.06	0.22	0.25	0.15	-0.11	0.13	0.08	0.18	0.17	0.09	0.17	1.00	0.04	0.12	0.21	0.17
Toaster.Oven	0.15	0.01	0.04	0.06	0.01	0.02	0.06	0.14	0.07	0.14	0.02	0.10	0.04	1.00	0.09	0.02	0.03
TV	0.40	-0.12	0.12	0.14	-0.13	-0.06	0.07	0.38	0.10	0.54	-0.09	0.36	0.12	0.09	1.00	0.12	0.07
Video.Conference.Camera	0.18	0.12	-0.00	-0.00	-0.02	-0.06	-0.11	0.03	0.04	0.09	-0.05	0.00	0.21	0.02	0.12	1.00	-0.02
Water.Boiler	0.04	0.04	0.63	0.30	0.06	-0.00	0.10	0.01	0.65	0.02	0.24	0.08	0.17	0.03	0.07	-0.02	1.00

Figura 19. Matriz de coeficientes de correlación de Spearman

Las correlaciones más significativas se dan entre las siguientes variables:

- *Laptop – Monitor*: 0.79
- *Coffee.Maker – Microwave*: 0.67
- Microwave – Water.Boiler: 0.65

En este caso, se observa que existe correlación no lineal entre dos parejas de variables que no estaban relacionadas linealmente (*Coffee.Maker – Microwave* y *Microwave – Water.Boiler*). Sin embargo, también se obtiene un coeficiente alto para las variables *Laptop* y *Monitor*, confirmando así que estas dos variables se encuentran correlacionadas mutuamente.

Además, las variables que tenían una correlación lineal débil con la variable target (*Skyspark*) tienen también una correlación no lineal débil (*AV.Controller, Coffee.Maker, Desktop.Server, Headset, Microwave, Phone.Charger, Toaster.Oven, Video.Conference.Camera* y *Water.Boiler*).

Una vez analizadas las relaciones entre las variables de nuestro conjunto de datos, podemos transformarlo para evitar que estas relaciones afecten a la precisión de

los modelos a generar. En cuanto a las variables que no están correlacionadas con *Skyspark*, podemos eliminarlas, ya que tienen un efecto mínimo sobre ella y, por lo tanto, no son útiles para realizar predicciones. En la **Figura 20** se muestra el nuevo conjunto de datos, ya sin estas variables.

Date_Time	Skyspark	Copier	Lamp	Laptop	Monitor	Printer	Projector	TV
2017-10-01 00:00:00	9.980	0.018790	0.014113	0.006767	0.015855	0.011983	0.010495	0.030557
2017-10-01 00:05:00	10.002	0.018790	0.014122	0.008408	0.015840	0.011883	0.010477	0.030654
2017-10-01 00:10:00	9.808	0.018780	0.014117	0.006767	0.015850	0.012010	0.010491	0.030547
2017-10-01 00:15:00	10.018	0.018780	0.014126	0.007718	0.015851	0.012523	0.010483	0.030553
2017-10-01 00:20:00	10.390	0.018793	0.014133	0.007914	0.015846	0.011907	0.010487	0.030553
...	...	...	...	...	...	...	...	...
2017-12-31 23:35:00	9.448	0.018710	0.012000	0.079770	0.019158	0.011740	0.010427	0.030480
2017-12-31 23:40:00	9.760	0.018710	0.011983	0.079760	0.019150	0.011723	0.010443	0.030480
2017-12-31 23:45:00	10.588	0.018707	0.011993	0.079860	0.019127	0.011732	0.010423	0.030473
2017-12-31 23:50:00	11.178	0.018710	0.012007	0.079900	0.019155	0.011780	0.010437	0.030477
2017-12-31 23:55:00	10.358	0.018717	0.011997	0.079996	0.019138	0.011958	0.010440	0.030483

Figura 20. Data set sin variables poco relacionadas con *Skyspark*

Por otro lado, para las variables descriptoras que se encuentran muy correlacionadas entre ellas (*Laptop* y *Monitor*), existen dos opciones. La primera de ellas consiste en realizar una selección de características, es decir, en descartar una de las variables, eliminando así la información redundante que supone esta correlación. La segunda opción consiste en realizar una extracción de características, de forma que obtengamos nuevas variables con información importante sobre cada una de las variables originales, pero sin estar correlacionadas entre ellas. Para ello, aplicamos un Análisis de Componentes Principales (PCA).

3.1.4 Análisis de Componentes Principales (PCA)

Con esta técnica buscamos realizar una reducción de la dimensionalidad de nuestra serie temporal, de forma que en cada Componente Principal (PC) tengamos un grado de información sobre cada una de las variables, sin estar correlacionados entre ellos, eliminando así el problema de la correlación entre las variables *Laptop* y *Monitor*.

Tras normalizar los valores de nuestro conjunto de datos, aplicamos PCA sobre las variables descriptoras con las que nos habíamos quedado (*Copier, Lamp, Laptop, Monitor, Printer, Projector* y *TV*), especificando como número de Componentes Principales el mismo número de variables: 7. En la **Figura 21** observamos el conjunto de datos transformado, en el que los nuevos atributos son los Componentes Principales calculados a partir de etas variables.

Date_Time	PC1	PC2	PC3	PC4	PC5	PC6	PC7
2017-10-01 00:00:00	-1.149516	-0.291675	0.255529	-0.087660	-0.122129	0.289152	-0.179384
2017-10-01 00:05:00	-1.142055	-0.293963	0.256674	-0.086361	-0.124481	0.281334	-0.169564
2017-10-01 00:10:00	-1.149035	-0.292193	0.256722	-0.087701	-0.121914	0.289727	-0.179343
2017-10-01 00:15:00	-1.139473	-0.288521	0.262416	-0.085559	-0.113571	0.291975	-0.173272
2017-10-01 00:20:00	-1.143439	-0.295435	0.259982	-0.088086	-0.124909	0.284403	-0.172609
...	...	...	...	...	...	...	...
2017-12-31 23:35:00	-1.031105	0.092535	-0.166260	-0.022208	0.106422	-0.264954	0.233805
2017-12-31 23:40:00	-1.032868	0.095190	-0.170062	-0.022025	0.107757	-0.266492	0.233767
2017-12-31 23:45:00	-1.031613	0.093634	-0.167631	-0.022136	0.107004	-0.266037	0.234478
2017-12-31 23:50:00	-1.029545	0.091782	-0.164500	-0.022273	0.106580	-0.264511	0.234629
2017-12-31 23:55:00	-1.028449	0.095251	-0.165527	-0.021345	0.110507	-0.263606	0.235436

Figura 21. Data set con Componentes Principales como atributos

Además, en la **Figura 22** podemos ver las cargas de cada PC, es decir, cuánta información sobre cada variable está contenida en los Componentes Principales. Además, el signo indica si la relación entre el componente y la variable es directa o inversa.

	Copier	Lamp	Laptop	Monitor	Printer	Projector	TV
PC1	0.318164	0.298390	0.486168	0.499520	0.350511	0.309250	0.324595
PC2	0.611761	-0.538207	0.033347	0.024401	0.366971	-0.391302	-0.215846
PC3	-0.048093	0.674872	0.055905	0.009641	0.256955	-0.492015	-0.480533
PC4	-0.320911	-0.066689	0.030654	0.068862	0.145588	-0.639390	0.675923
PC5	-0.594770	-0.295527	0.046877	0.050451	0.664432	0.237660	-0.237102
PC6	0.255683	0.272132	-0.552809	-0.445708	0.461170	0.208420	0.316544
PC7	0.009936	0.002213	0.671318	-0.737461	0.033247	0.020680	0.062023

Figura 22. Cargas de los Componentes Principales

Esto significa que el Componente Principal 1 (PC1), por ejemplo, se calcula de la siguiente manera [29]:

$$PC1 = 0.318164 * Copier + 0.298390 * Lamp + 0.486168 * Laptop + 0.499520 * Monitor + 0.350511 * Printer + 0.309250 * Projector + 0.324595 * TV$$

Los demás componentes son calculados de forma análoga.

Por otro lado, como se ha explicado en la parte teórica, los Componentes Principales se encuentran ordenados descendentemente según sus autovalores, es decir, aquellos que mejor explican la variabilidad del conjunto de variables iniciales se encuentran antes. Esto se muestra en la **Figura 23**, donde se puede ver que PC1 tiene una variabilidad explicada de 3.417665, mucho mayor a los siguientes componentes.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
0	3.417655	0.820206	0.789622	0.726443	0.67358	0.492337	0.080421

Figura 23. Variabilidad explicada de cada PC

Sin embargo, para tener una mejor idea de la proporción de variabilidad explicada de cada componente, analizamos las ratios de variabilidad explicada. De esta forma, tenemos información que puede ser interpretada más intuitivamente para determinar con qué Componentes Principales nos quedaremos.

En la **Figura 24** observamos las ratios de variabilidad explicada acumuladas. Una forma de elegir los componentes consiste en seleccionar aquellos con los que se alcanza cierto umbral. En este caso, si fijamos como umbral el 98% de variabilidad explicada, vemos que nos podemos quedar con los 6 primeros componentes, ya que con ellos se llega a explicar el 98.85% de variabilidad.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
0	0.488218	0.605386	0.718185	0.821958	0.918181	0.988512	1.0

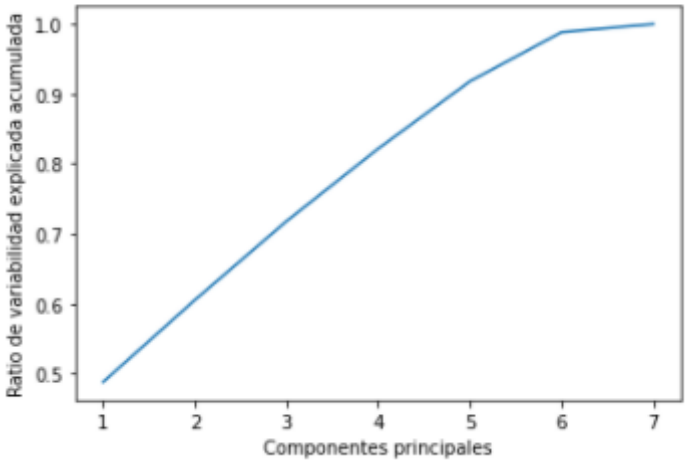

Figura 24. Ratios de variabilidad explicada acumuladas

Otra de las opciones para elegir el número de Componentes Principales consiste en aplicar el método del codo. Para ello, se visualiza de forma gráfica las ratios de variabilidad explicada por cada componente y se identifica el PC a partir del cual la curva se aplana, ya que los siguientes componentes no incluyen información relevante sobre las variables iniciales. En la **Figura 25** observamos estas ratios. Se puede ver que el valor desciende considerablemente en PC2, confirmando que PC1explica gran parte de la variabilidad total.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
0	0.488218	0.117168	0.112799	0.103774	0.096222	0.070331	0.011488

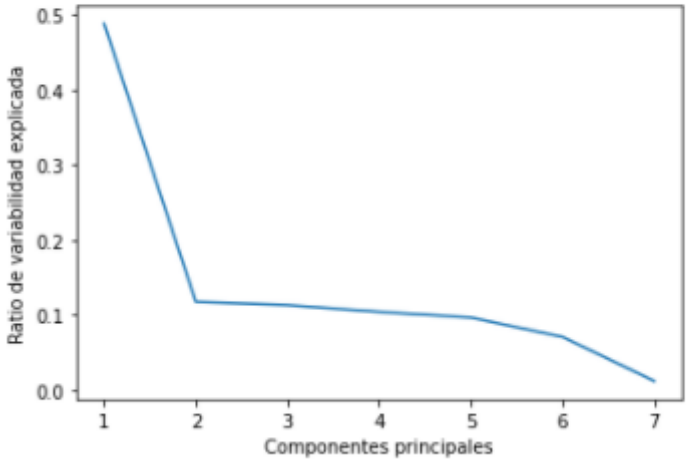

Figura 25. Ratios de variabilidad explicada por PC

Tras este análisis, se decide mantener de momento los seis primeros Componentes Principales, de acuerdo a los resultados de las ratios de variabilidad explicada acumulados. El conjunto de datos redimensionado se muestra en la **Figura 26**.

Date_Time	Skyspark	PC1	PC2	PC3	PC4	PC5	PC6
2017-10-01 00:00:00	9.980	-1.149516	-0.291675	0.255529	-0.087660	-0.122129	0.289152
2017-10-01 00:05:00	10.002	-1.142055	-0.293963	0.256674	-0.086361	-0.124481	0.281334
2017-10-01 00:10:00	9.808	-1.149035	-0.292193	0.256722	-0.087701	-0.121914	0.289727
2017-10-01 00:15:00	10.018	-1.139473	-0.288521	0.262416	-0.085559	-0.113571	0.291975
2017-10-01 00:20:00	10.390	-1.143439	-0.295435	0.259982	-0.088086	-0.124909	0.284403
...	...	...	...	...	...	...	...
2017-12-31 23:35:00	9.448	-1.031105	0.092535	-0.166260	-0.022208	0.106422	-0.264954
2017-12-31 23:40:00	9.760	-1.032868	0.095190	-0.170062	-0.022025	0.107757	-0.266492
2017-12-31 23:45:00	10.588	-1.031613	0.093634	-0.167631	-0.022136	0.107004	-0.266037
2017-12-31 23:50:00	11.178	-1.029545	0.091782	-0.164500	-0.022273	0.106580	-0.264511
2017-12-31 23:55:00	10.358	-1.028449	0.095251	-0.165527	-0.021345	0.110507	-0.263606

Figura 26. Data set tras aplicar PCA

Una vez hecho esto, podemos pasar a determinar la ventana de tiempo a utilizar para realizar las estimaciones, es decir, cuántos intervalos de tiempo anteriores de la serie temporal tendremos en cuenta en cada observación para generar los modelos de predicción. Para ello, nos apoyamos en las autocorrelaciones y en la generación de modelos ARIMA, tanto de la variable target, como de cada componente conveniente.

3.1.5 Autocorrelación

La autocorrelación nos ayudará a determinar cuántos instantes de tiempo previos afectan al consumo de energía de una observación determinada. Este número de periodos anteriores se conoce como retardo o lag, y es requerido para la obtención de modelos LSTM.

La autocorrelación simple y la autocorrelación parcial de la variable target (*Skyspark*) se muestran en la **Figura 27** y la **Figura 28**, respectivamente. El eje *x* representa los retardos, y el eje *y* el valor de la autocorrelación, un valor entre 0 y 1, como se ha visto previamente.

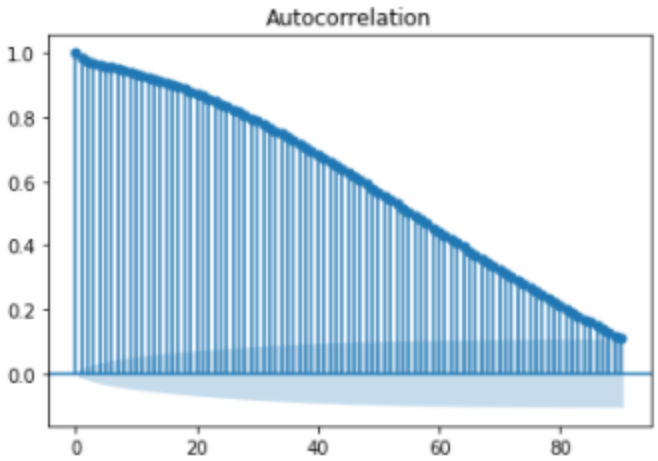

Figura 27. Autocorrelación simple de *Skyspark*

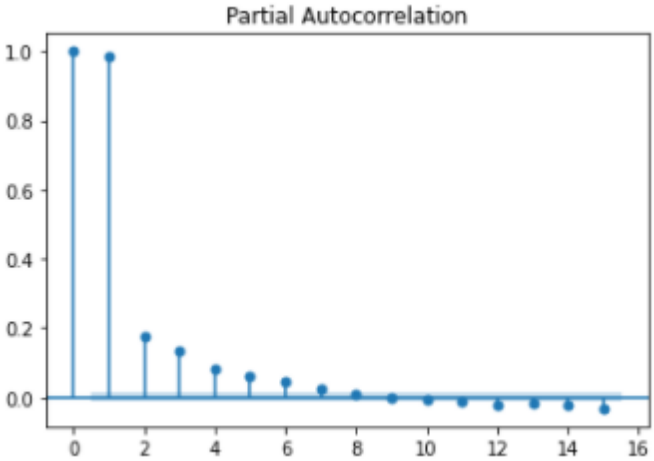

Figura 28. Autocorrelación parcial de *Skyspark*

Por un lado, según la gráfica de autocorrelación simple, el límite del intervalo de confianza de 0,05 se encuentra en el retardo 90. Esto significa que los primeros 90 retardos son estadísticamente significativos y, por lo tanto, influyen directa o indirectamente en los valores de consumo de energía de cada observación. Por otro lado, la gráfica de autocorrelación parcial indica que sólo 7 retardos son

estadísticamente significativos. Es decir, sólo 6 instantes previos producen un efecto directo en los valores de una observación determinada.

Sin embargo, la verdadera utilidad de las correlaciones simple y parcial es que sirven para la obtención de modelos ARIMA. Esto se debe a que el retardo obtenido de la autocorrelación simple corresponde al parámetro q de los modelos ARIMA, mientras que el retardo obtenido de la autocorrelación parcial corresponde al parámetro p.

De esta forma, a partir de un modelo inicial creado con los valores dados por las autocorrelaciones, se puede ajustar nuevos modelos para encontrar el más preciso, con el que se establecerá la ventana de tiempo a utilizar en nuestros modelos LSTM posteriormente. Por ejemplo, el modelo ARIMA con el que empezaríamos el ajuste sería ARIMA(90,0,6). Cabe mencionar que el parámetro d es 0 porque la serie temporal en la variable *Skyspark* es estacionaria, es decir, no se observa tendencia ni estacionalidad. Esto se muestra en la **Figura 29**, donde se tiene una representación de los valores de la variable *Skyspark* a lo largo del tiempo (primera gráfica), y su descomposición en tendencia, estacionalidad y residuos, respectivamente.

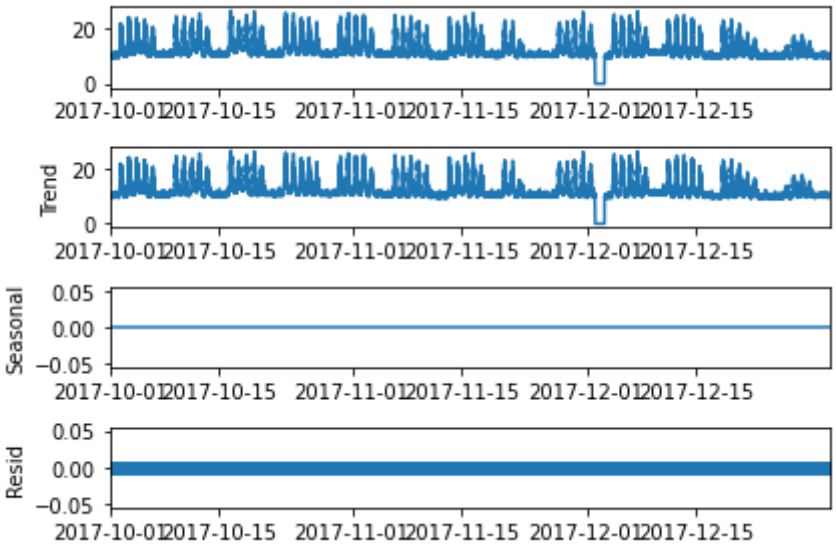

Figura 29. Descomposición de la serie temporal en *Skyspark*

A pesar de que se podría realizar el ajuste de forma manual, probando distintas combinaciones, en este proyecto se realizará de forma automática. Gracias a esto, obtendremos el modelo ARIMA óptimo para cada variable y, de acuerdo a los parámetros con los que han sido construidos, pasaremos a la fase de modelado con redes neuronales LSTM.

3.1.6 Modelos ARIMA

Como se ha mencionado, los modelos ARIMA para cada variable se generarán de forma automática. Pero antes de aplicar esto, debemos asegurarnos de que cada variable es estacionaria, ya que, en caso de ser no estacionaria, será necesario

diferenciar la serie temporal. En el punto anterior se ha realizado esto visualmente a través de las gráficas de tendencia y estacionalidad, pero para realizarlo de forma sistemática, emplearemos el test de Dickey-Fuller.

Empezando por la variable target, es decir *Skyspark*, el test de Dickey-Fuller nos devuelve un valor-p de 0.01. Este valor se encuentra por debajo del umbral de significancia estadística de 0,05, lo que indica que la serie es, en efecto, estacionaria y, por lo tanto, no necesita ser diferenciada. Una vez comprobada la estacionariedad, pasamos a obtener el modelo ARIMA para *Skyspark*, indicando que se trata de una serie estacionaria. El modelo obtenido tiene como parámetro p un valor de 5 y como parámetro q un valor de 1.

De forma similar, se realiza este proceso para las demás variables, ahora llamados Componentes Principales. Sin embargo, no se aplicará a los 6 componentes que se habían seleccionado, sino que sólo a aquellos que se encuentren más correlacionados con *Skyspark*, para no tener en cuenta retardos en variables que no tienen un efecto relevante en la variable target. Estos componentes son PC1 Y PC2, como se muestra en la matriz de correlación de Pearson de la **Figura 30**.

	Skyspark	PC1	PC2	PC3	PC4	PC5	PC6
Skyspark	1.000000	8.474843e-01	4.226491e-02	-2.407451e-02	1.475352e-02	4.591142e-02	-1.780296e-01
PC1	0.847484	1.000000e+00	5.289389e-15	-5.612004e-16	-2.389070e-15	-5.023234e-15	9.405892e-16
PC2	0.042265	5.289389e-15	1.000000e+00	-5.431375e-16	-3.181588e-15	-4.501825e-15	2.084218e-15
PC3	-0.024075	-5.612004e-16	-5.431375e-16	1.000000e+00	-5.011040e-16	2.248216e-16	7.851842e-17
PC4	0.014754	-2.389070e-15	-3.181588e-15	-5.011040e-16	1.000000e+00	2.392261e-15	-7.541371e-16
PC5	0.045911	-5.023234e-15	-4.501825e-15	2.248216e-16	2.392261e-15	1.000000e+00	-2.460730e-15
PC6	-0.178030	9.405892e-16	2.084218e-15	7.851842e-17	-7.541371e-16	-2.460730e-15	1.000000e+00

Figura 30. Correlación de Pearson (Componentes Principales)

El test de Dickey-Fuller indica que la serie temporal en ambos componentes es estacionaria, por lo que no se les aplicará diferenciación. Tras esto, se obtiene sus respectivos modelos ARIMA. Para PC1, el parámetro p tiene un valor de 5 y q, de 0, mientras que para PC2, el parámetro p tiene un valor de 5 y q, de 1.

De forma resumida, en la **Tabla 2** se muestran los valores p y q de los modelos ajustados para cada variable, que serán utilizados en el siguiente capítulo.

Modelo	p	q
Skyspark	5	1
PC1	5	0
PC2	5	1

Tabla 2. Parámetros p y q de los modelos ARIMA

3.2 Modelado y Evaluación

Para la fase de modelado se ha decidido utilizar redes neuronales LSTM, debido a las ventajas explicadas en el Capítulo 3. Para poder realizar esto, es necesario determinar qué intervalos de tiempo previos se tendrán en cuenta para las predicciones. Tras haber obtenido los valores de autocorrelación simple y autocorrelación parcial para cada una de las variables a utilizar, se decide formar tres grupos de modelos.

El primer grupo consiste en modelos de predicción en los que se tiene en cuenta sólo PC1 para la predicción de *Skyspark*, ya que se trata de la variable más correlacionada con la variable target. El segundo grupo incluye, además de PC1, a PC2, por ser la segunda variable más correlacionada con *Skyspark*. No se incluyen más variables, ya que tienen una correlación muy poco significativa con la variable a predecir. Por último, en el tercer grupo de modelos se realiza la estimación de *Skyspark* a partir de las variables originales, para determinar si la extracción de características ha supuesto alguna ventaja frente a utilizar las variables de forma directa.

Dado que para las tres variables se obtuvo como parámetro p de los modelos ARIMA un valor de 5, siendo éste en los tres casos mayor que el valor del parámetro q, para los tres grupos de modelos se probará con retardos de *t-5*. Es decir, que se considerará los valores de consumo energético en *t*, además de los valores en desde *t-1* hasta *t-5*

El proceso de modelado sigue los siguientes pasos:

- En primer lugar, se normaliza los valores y se separa las variables predictoras de la variable target. Las variables predictoras corresponden al conjunto de datos de entrada, mientras que la variable target se trata del dato de salida. Las redes neuronales LSTM requieren que el conjunto de datos de entrada sea un array de tres dimensiones: número de observaciones, número de intervalos de tiempo y número de variables. De esta forma, para cada observación se tiene un array por cada intervalo de tiempo a tener en cuenta, incluyendo el instante actual y los retardos establecidos. Es decir, el conjunto de datos de entrada transformado para *t-i* contendrá los valores en *t* y los valores en *t-1*, *t-2*, hasta *t-i*. Por ejemplo, en la Figura 31 se muestra el conjunto de variables predictoras para el primer grupo de modelos, donde se incluyen 5 retardos (valores desde *t* hasta *t-5*).

- Una vez preparado nuestro conjunto de datos según la ventana de tiempo determinada, es necesario dividir el total de observaciones en datos de entrenamiento y datos de prueba. Se selecciona el 70% de los datos para entrenamiento y el 30% para pruebas.

- Ya con nuestros datos de entrenamiento, procedemos a crear y entrenar nuestro modelo LSTM. En todos los modelos, por cuestiones de rendimiento, se ha dejado que el tamaño de lote o *batch size* se calcule de

forma automática durante el entrenamiento. El número de neuronas y de épocas se irá ajustando de acuerdo a los resultados. En cuanto a la función de pérdida, se utiliza la métrica de Mínimos Errores Cuadrados.

- Para poder medir la precisión de nuestro modelo, realizamos las predicciones con el conjunto de datos de prueba y calculamos los valores de RMSE, MAPE y R^2 entre las predicciones y los valores reales. Finalmente, reconstruimos nuestro conjunto de datos para invertir la normalización y comparar gráficamente los resultados predichos con los reales. Los resultados obtenidos en este último paso se utilizan en la fase de evaluación de modelos para realizar una comparación entre ellos.

```
[[[0.08809926]
  [0.08867936]
  [0.08813668]
  [0.08888007]
  [0.08857178]
  [0.08867663]]

 [[0.08867936]
  [0.08813668]
  [0.08888007]
  [0.08857178]
  [0.08867663]
  [0.09058979]]

 [[0.08813668]
  [0.08888007]
  [0.08857178]
  [0.08867663]
  [0.09058979]
  [0.08790316]]

 ...
```

Figura 31. Datos de entrada: PC1 con 5 retardos

Antes de crear los tres grupos de modelos previamente mencionados, se decide generar un modelo de predicción inicial que servirá como punto de partida para ajustar, tanto el número de neuronas como de épocas, en los siguientes grupos de modelos. Para producir este primer modelo se relaciona la ventana de tiempo establecida con el número de neuronas y de épocas de entrenamiento de la red LSTM. Por lo tanto, el modelo se genera con una red compuesta por 5 neuronas en la capa oculta y entrenada en 5 épocas.

La **Figura 32** muestra la función de pérdida para los conjuntos de datos de entrenamiento y de prueba. Se puede observar que los valores para el conjunto de datos de prueba no llegan a alcanzar los valores para el conjunto de datos de entrenamiento, manteniéndose superior a este en todos los puntos.

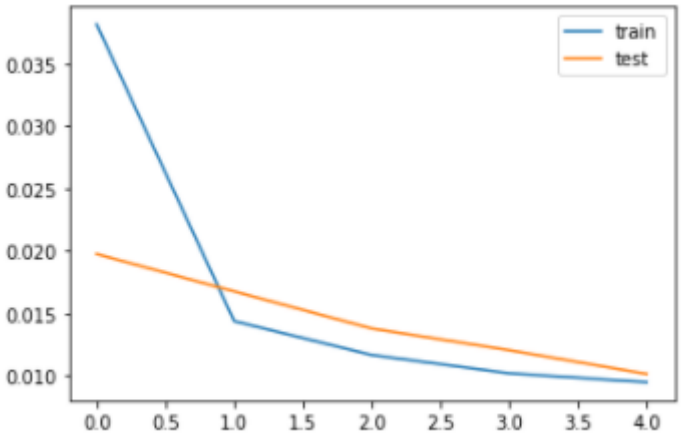

Figura 32. Función de pérdida para modelo de 5 neuronas y 5 épocas de entrenamiento

Por otra parte, las métricas de calidad de las predicciones no son lo suficientemente óptimas. En cuanto a los errores obtenidos, se tiene un RMSE de 0.10 y un MAPE de 0.17, y en cuanto a la precisión de las predicciones, se tiene una R^2 de 0.40. Esta baja precisión se puede ver gráficamente en la **Figura 33**, donde en pocos casos los valores predichos, marcados en rojo, se encuentran sobrepuestos a los valores reales, marcados en azul. Así, es bastante visible el desfase inferior de las predicciones en el eje *Y* con respecto a las salidas reales.

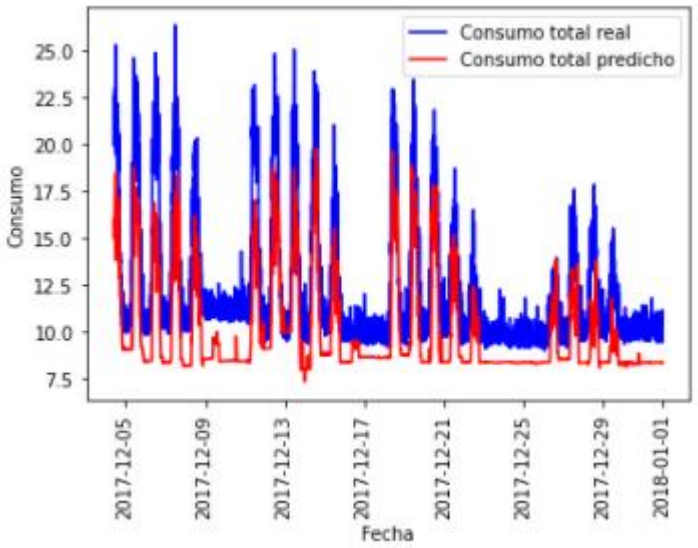

Figura 33. Consumo real y predicho con un modelo de 5 neuronas y épocas de entrenamiento

Ante estos resultados tan poco precisos, se decide probar con mayor número de neuronas y de épocas de entrenamiento en cada uno de los siguientes grupos de modelos, con el objetivo de conseguir mejores predicciones.

3.2.1 Grupo de modelos 1: PC1 y *Skyspark*

Se ha generado un primer modelo del que partir y, de acuerdo a la función de pérdida y de las métricas de calidad, se han generado los subsecuentes modelos. Este primer modelo cuenta con una red neuronal LSTM con 50 neuronas y se han establecidos 25 épocas de entrenamiento.

Modelo 1: 50 neuronas y 25 épocas de entrenamiento

Como se observa en la **Figura 34**, la función de pérdida para el conjunto de datos de prueba decrece lentamente, alcanzando a los valores de pérdida del conjunto de datos de entrenamiento ya en los últimos ciclos. Esto puede significar que le faltan épocas de entrenamiento para poder aprender lo suficiente sobre los datos de entrada.

Además, tras hacer las predicciones, se tiene unos valores de RMSE y MAPE de 0.09 y 0.17, respectivamente, mientras que según R^2 se da un 54% de acierto. Por esta razón, se decide probar un segundo modelo con el mismo número neuronas, pero incrementando el número de épocas a 40.

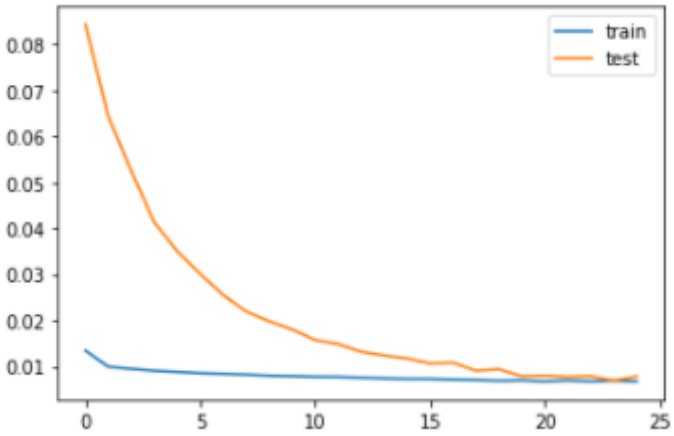

Figura 34. Funciones de pérdida para el modelo 1 del grupo 1

Modelo 2: 50 neuronas y 40 épocas de entrenamiento

En este modelo se observa que, a partir de la época 25, la función de pérdida para el conjunto de datos de prueba se mantiene en valores bastante bajos. Estos valores rondan el 0.1, prácticamente igual que los valores de la función de pérdida para el conjunto de datos de entrenamiento, como se observa en la **Figura 35**.

En cuanto a las métricas de calidad, se obtienen mejores resultados de predicción que en el caso anterior: un RMSE de 0.08, un MAPE de 0.13 y un R^2 de 0.65. Dados estos resultados, se decide aumentar el número de neuronas a 100 para observar si aporta mejoras al proceso de aprendizaje, aunque manteniendo el número de épocas.

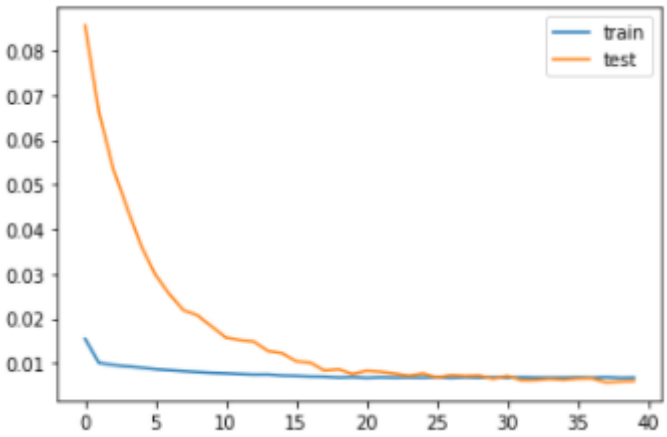

Figura 35. Funciones de pérdida para el modelo 2 del grupo 1

Modelo 3: 100 neuronas y 40 épocas de entrenamiento

En este caso, la función de pérdida para el conjunto de datos de entrenamiento presenta valores mínimos desde el principio, manteniéndose durante las siguientes épocas (ver **Figura 36**). Esto se debe al incremento del número de neuronas. Sin embargo, se observa un comportamiento inestable en la función de pérdida del conjunto de datos de prueba. Aproximadamente, en la época 15 alcanza los valores de pérdida de los datos de entrenamiento, pero en las subsecuentes épocas estos valores varían, ascendiendo y descendiendo continuamente. Esto nos lleva a pensar que el número de épocas establecido para entrenar una red con un número de neuronas elevado no es suficiente.

Por otro lado, con este modelo se obtiene un RMSE de 0.1, un MAPE de 0.18 y un R2 de 0.44, indicando menor precisión predictiva que el modelo anterior. Debido a esto, se decide crear un nuevo modelo con el mismo número de neuronas, pero mayor número de épocas de entrenamiento.

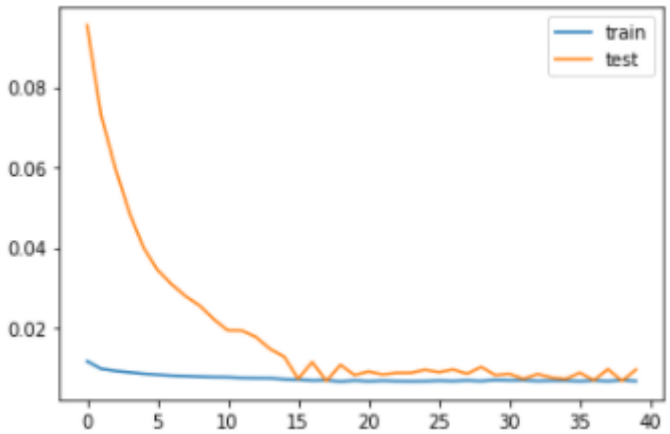

Figura 36. Funciones de pérdida para el modelo 3 del grupo 1

Modelo 4: 100 neuronas y 100 épocas de entrenamiento

En el último modelo de este grupo, se ha decidido incrementar el número de épocas de entrenamiento a 100, basados en los resultados del modelo anterior. En la **Figura 37** se observa que, en este caso, la función de pérdida para los datos de prueba también alcanza los valores de pérdida de los datos de entrenamiento antes de la época 20. A partir de entonces, sufre pequeñas fluctuaciones, pero estas van desapareciendo gradualmente.

Una vez hechas las predicciones, las métricas de calidad indican una mejora frente a los anteriores modelos, con un RMSE de 0.07, un MAPE de 0.10 y un R^2 de 0.74. Aun así, es posible que incrementando el número de neuronas en la capa oculta se pueda obtener mejores resultados, dada la inestabilidad de la función de pérdida en las primeras épocas, aunque no se muestra la comprobación de esto en este trabajo.

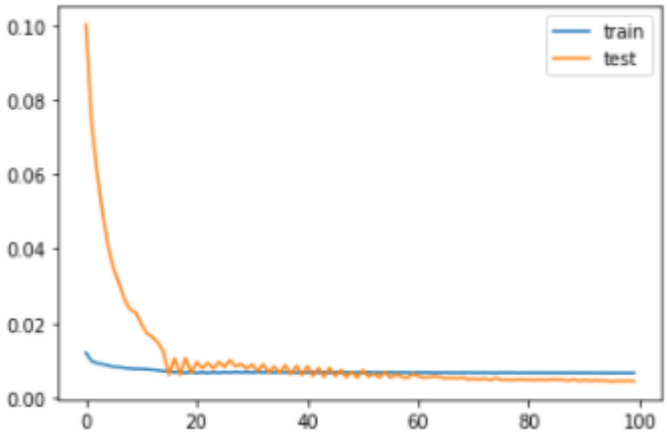

Figura 37. Funciones de pérdida para el modelo 4 del grupo 1

3.2.2 Grupo de modelos 2: PC1, PC2 y *Skyspark*

De forma similar al primer grupo de modelos, en este apartado también partiremos de un modelo base, a partir del cual se irá ajustando el número de neuronas y de épocas de entrenamiento. La diferencia, sin embargo, radica en que para la generación de estos modelos se tendrá en cuenta también la variable PC2 en el conjunto de datos de entrada.

Partiendo de los resultados anteriores, se decide generar un modelo con los mismos parámetros que el modelo 2 del primer grupo de modelos: 50 neuronas y 40 épocas de entrenamiento.

Modelo 1: 50 neuronas y 40 épocas de entrenamiento

Las funciones de pérdida de este modelo, mostradas en la **Figura 38**, nos indican que alrededor de la época 20 los valores de pérdida del conjunto de datos de prueba se reducen hasta alcanzar los valores de pérdida del conjunto de datos de entrenamiento. A partir de entonces, presenta pequeñas variaciones.

Por otro lado, se obtienen los siguientes valores para las métricas de calidad para las predicciones: un RMSE de 0.08, un MAPE de 0.15 y un R^2 de 0.59. De acuerdo a estos resultados, se decide probar un segundo modelo, en el que se incrementará tanto el número de neuronas, como el de épocas de entrenamiento. En concreto, se establecerán 75 neuronas y 50 épocas.

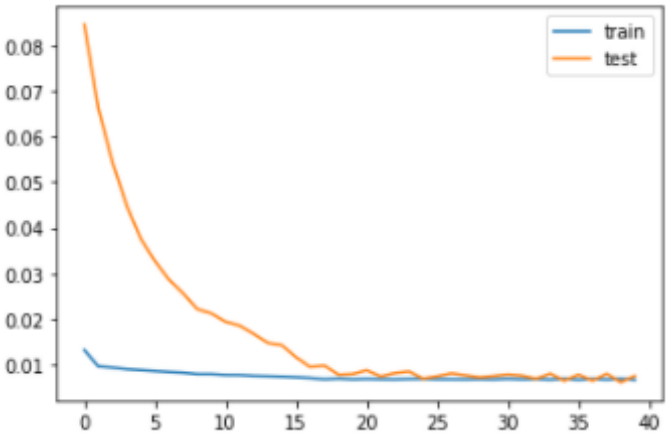

Figura 38. Funciones de pérdida para el modelo 1 del grupo 2

Modelo 2: 75 neuronas y 50 épocas de entrenamiento

En este modelo se tiene unas funciones de pérdida peculiares. Como se muestra en la **Figura 39**, la función de pérdida para los datos de entrenamiento se mantiene en valores mínimos, mientras que la función de pérdida para el conjunto de datos de prueba alcanza esos valores mínimos en torno a la época 30. Sin embargo, a partir de ese punto presenta variaciones sin llegar a estabilizarse.

Además, tras realizar las predicciones, se obtiene un RMSE de 0.09 y un MAPE de 0.16, iguales a los del modelo anterior. No obstante, R^2 presenta un valor de 0.55, indicando menor precisión que el modelo 1. Por ello, se genera un nuevo modelo reduciendo el número de neuronas a 50, pero incrementando el número de épocas de entrenamiento a 100, para poder ver si con más épocas la función de pérdida logra estabilizarse.

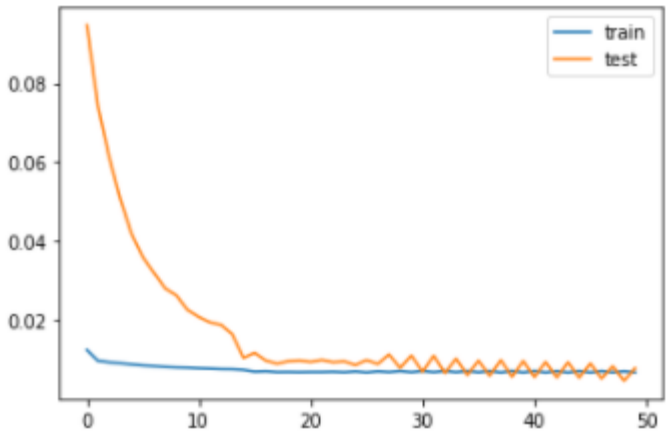

Figura 39. Funciones de pérdida para el modelo 2 del grupo 2

Modelo 3: 50 neuronas y 100 épocas de entrenamiento

En este caso, la función de pérdida del conjunto de datos de prueba alcanza los valores del conjunto de datos de entrenamiento poco después de la época 20, como se ve en la **Figura 40**. Aunque desde ese punto se producen pequeñas variaciones, la función tiende a estabilizarse eventualmente.

Con respecto a las métricas de calidad, se tiene un RMSE de 0.07, un MAPE de 0.11 y un R^2 de 0.72, mejorando los resultados de predicción de los modelos anteriores. Sabiendo esto, se decide comprobar si estos resultados pueden ser superados. Por ello, se crea otro modelo con los mismos parámetros que el último modelo del grupo 1: 100 neuronas y 100 épocas.

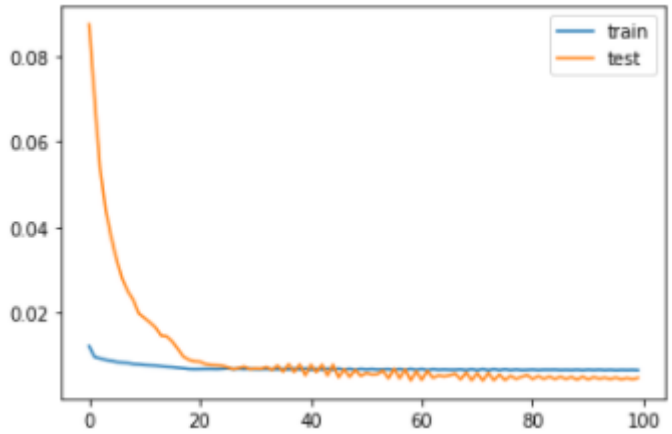

Figura 40. Funciones de pérdida para el modelo 3 del grupo 2

Modelo 4: 100 neuronas y 100 épocas de entrenamiento

Este último modelo presenta las funciones de pérdida mostradas en la **Figura 41**. Como se puede observar, los valores de pérdida para el conjunto de datos de entrenamiento también se mantienen bajos en este caso, pero los valores de

pérdida para el conjunto de datos de prueba presentan alteraciones que continúan hasta la última época, sin llegar a estabilizarse en ningún punto.

Por otro lado, las métricas de calidad de predicción indican que este modelo no ha supuesto ninguna mejora frente al anterior, ya que tanto el RMSE (0.08) como el MAPE (0.13) son superiores. Además, R*2* indica una menor precisión, al tener un valor de 0.66.

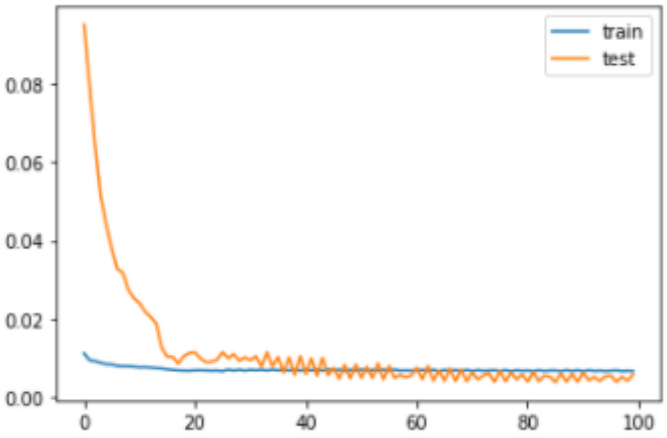

Figura 41. Funciones de pérdida para el modelo 4 del grupo 2

3.2.3 Grupo de modelos 3: Variables originales y *Skyspark*

Por último, se sigue el mismo proceso de generación de modelos seguido en los dos primeros grupos. A diferencia de los grupos anteriores, estos modelos no serán generados teniendo como datos de entrada los Componentes Principales. En su lugar, se utilizarán las variables originales de las que está formado PC1, por ser éste el componente más correlacionado con *Skyspark*. Es decir, el conjunto de datos de entrada para los modelos de este grupo corresponde a las variables originales antes de aplicar PCA, y sus respectivos retardos.

El objetivo de tener estos últimos modelos es posteriormente realizar una comparación con los resultados de los modelos en los que se usan los Componentes Principales. Por esta razón, se generarán modelos con los mismos parámetros (número de neuronas y número de épocas de entrenamiento) del primer grupo, donde se consideraba sólo PC1.

Modelo 1: 50 neuronas y 25 épocas

La **Figura 42** muestra las funciones de pérdida para los conjuntos de datos de entrenamiento y de prueba de este modelo. Como se puede observar, los valores de pérdida de los datos de prueba alcanzan los valores de pérdida de los datos de entrenamiento ya en las últimas épocas. Por lo tanto, es en este caso también es necesario aumentar el número de épocas para obtener mejores resultados.

Además, las predicciones con este modelo presentan un RMSE de 0.08, un MAPE de 0.14 y un R^2 de 0.65. Es bueno señalar, que los resultados de estas métricas son peores en el modelo obtenido con PC1 como entrada para esta configuración.

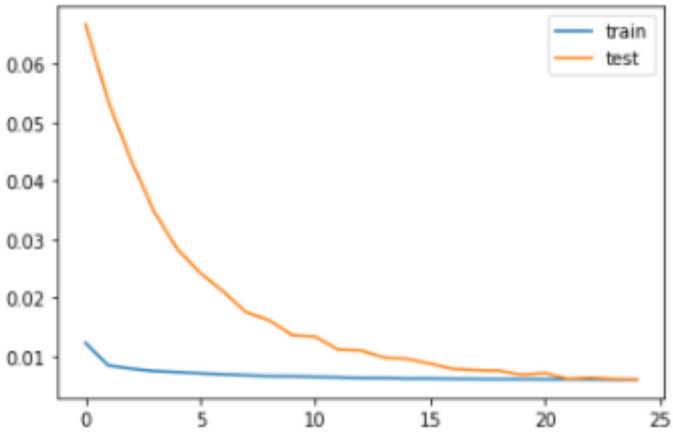

Figura 42. Funciones de pérdida para el modelo 1 del grupo 3

Modelo 2: 50 neuronas y 40 épocas

En este caso, la función de pérdida para el conjunto de datos de prueba se presenta estable, descendiendo hasta alcanzar los valores de pérdida del conjunto de datos de entrenamiento alrededor de la época 20. Este se puede ver en la **Figura 43.**

En cuanto a las métricas de calidad de predicción, los valores indican que este modelo es más preciso que el anterior, teniendo un RMSE de 0.07, un MAPE de 0.12 y un R^2 de 0.74. De nuevo, los valores obtenidos en las métricas en este caso son mejores al modelo construido con PC1 para esta configuración de la red.

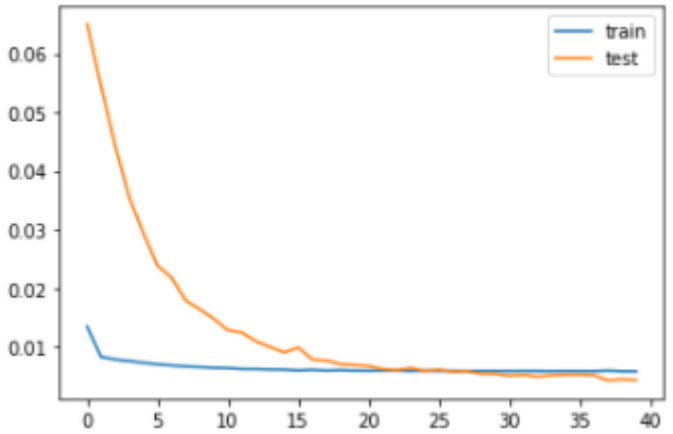

Figura 43. Funciones de pérdida para el modelo 2 del grupo 3

Modelo 3: 100 neuronas y 40 épocas

Este modelo presenta una función de pérdida para el conjunto de datos que alcanza los valores de pérdida para el conjunto de entrenamiento aproximadamente en la época 30, como se muestra en la **Figura 44**. Presenta ciertas variaciones, aunque se van reduciendo a medida que se completan épocas.

Por otro lado, tras realizar las predicciones, se obtiene un RMSE de 0.07, un MAPE de 0.13 y un R^2 de 0.70, sin suponer mejoras con respecto al modelo anterior. Ahora, presenta una mejora considerable con respecto a los resultados obtenidos con PC1 para esta configuración de la red.

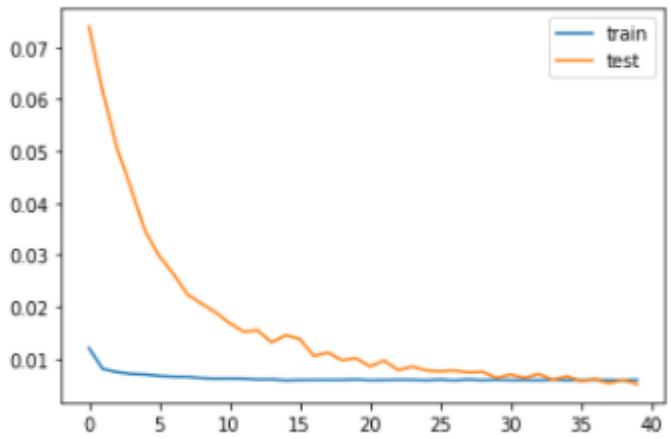

Figura 44. Funciones de pérdida para el modelo 3 del grupo 3

Modelo 4: 100 neuronas y 100 épocas

Finalmente, para este modelo se tiene valores para el conjunto de datos de prueba cercanos a los valores de pérdida del conjunto de datos de entrenamiento a partir de la época 30, aproximadamente. A partir de entonces, estos valores van variando, pero se mantienen en un rango en torno a los valores de pérdida de los datos de entrenamiento. La **Figura 45** ilustra este comportamiento.

De forma similar al caso del grupo 1, se obtiene errores de predicción menores que con el modelo anterior. En concreto, un RMSE de 0.07 y un MAPE de 0.12. El valor de R^2 es de 0.73, el cual es mayor que el del modelo anterior, lo que indica mayor precisión. Ahora bien, con respecto al modelo basado en PC1 como entrada, es en el único caso en el que las métricas no son mejores. De acuerdo a los resultados obtenidos, al considerar un mayor número de variables, aumentar la cantidad de neuronas y de épocas de entrenamiento mejora la capacidad de aprendizaje de la red. Esto podría significar que, si se usan todas las variables, una red con bastantes neuronas resulte ser una configuración óptima, ajustando un número de épocas adecuado para ello. Ahora bien, para el caso del uso de los PCs, con redes con pocas neuronas se obtuvieron resultados ligeramente mejores (por ejemplo, para el modelo 2 con 50 neuronas).

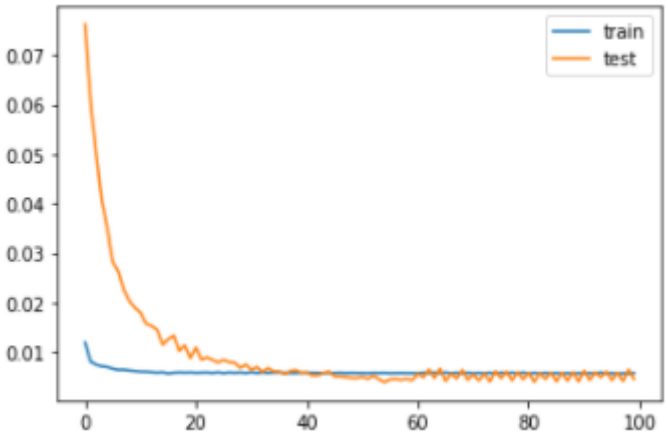

Figura 45. Funciones de pérdida para el modelo 4 del grupo 3

3.3 Comparación de modelos

Una vez generados los modelos de predicción, y tras haberlos evaluado en el proceso, nos disponemos a realizar una comparación entre ellos, presentando en mayor detalle los resultados obtenidos.

Tras haber generado cada uno de estos modelos con el conjunto de datos de entrenamiento, se ha realizado la estimación del consumo energético total para el conjunto de datos de prueba. Estos valores predichos, se han comparado posteriormente con los valores reales de consumo energético total para dichas observaciones. De esta forma, se ha obtenido las métricas de calidad que nos dan una idea de la precisión de nuestros modelos (RMSE, MAPE y R^2). Además, se generan gráficas en las que se muestra los valores predichos y los valores reales, para poder compararlos visualmente.

El primer grupo de modelos, en los que sólo se tiene como variable descriptora a PC1, cuenta con 4 candidatos. En la **Tabla 3** se resume los parámetros con los que fueron generados cada uno de los modelos, así como los valores de las métricas de calidad obtenidas con ellos.

De todos los modelos, el que presenta errores más altos, tanto de RMSE como de MAPE, es el tercero, con valores de 0.10 y 0.18, respectivamente. Además, el valor de R^2 indica que sólo un 44% de los resultados predichos coincide con los resultados reales. Este comportamiento puede ser explicado por la incorporación de un elevado número de neuronas, sin aplicar los suficientes ciclos de entrenamiento. Por esta razón, a pesar de contar con más neuronas que el primer modelo, del que se partió para generar los demás, no se superan esos resultados. Este primer modelo, sin embargo, tampoco presenta resultados lo suficientemente buenos, ya que el número de neuronas es bajo y el número de épocas también. A partir de este modelo se generó el segundo modelo, aumentando el número de épocas de entrenamiento. Este modelo sí supuso mejoras, obteniendo errores de predicción más bajos y una precisión del 65%. Tras las experiencias anteriores, para el cuarto modelo se incrementó tanto el número de neuronas como de épocas. Con él, se obtuvieron los mejores resultados de primer grupo de modelos, alcanzando una precisión de predicción del 74% y errores más bajos que en los casos anteriores.

Modelo	No. de neuronas	No. de épocas	RMSE	MAPE	R^2
1	50	25	0.09	0.17	0.54
2	50	40	0.08	0.13	0.65
3	100	40	0.10	0.18	0.44
4	100	100	0.07	0.10	0.74

Tabla 3. Resumen de modelos del grupo 1

Adicionalmente, comparamos los resultados reales y los predichos a través de una gráfica para cada modelo. Comprobamos así, que con el cuarto modelo se obtienen predicciones más acertadas, ya que el desfase en el eje *y* es menor que en los

demás casos, habiendo más puntos en los que los valores predichos coinciden con los valores reales (se encuentran superpuestos). Esto se observa en la **Figura 46**.

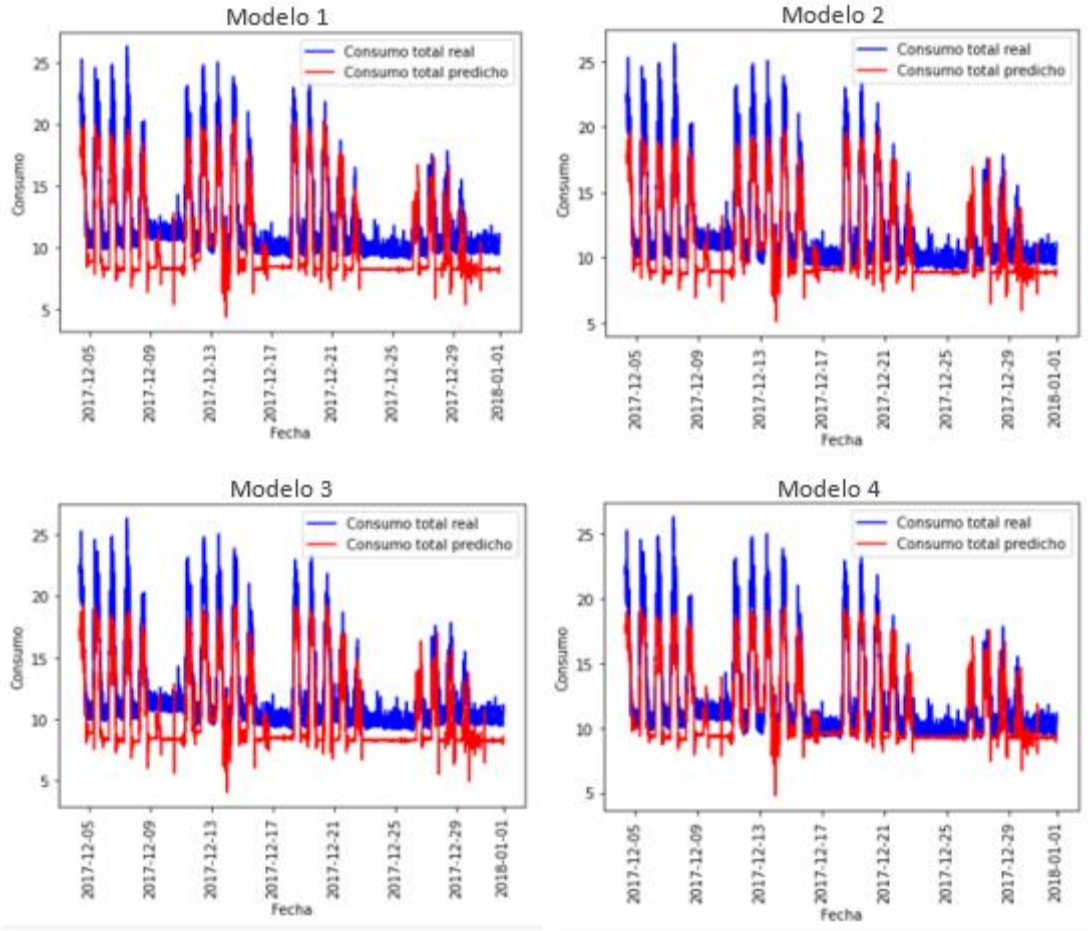

Figura 46. Valores reales y predichos de los modelos del grupo 1

Con respecto al segundo grupo de modelos, en los que las variables descriptoras son PC1 y PC2, los resultados obtenidos se encuentran resumidos en la **Tabla 4**. Para generar estos modelos, se tomó como punto de partida el segundo modelo del primer grupo, ya que dio mejores resultados que el modelo 1 de ese grupo. Se observa que, a pesar de tener los mismos parámetros, presenta un RMSE de 0.09 y un MAPE de 0.16 frente a un RMSE de 0.08 y un MAPE de 0.13 del modelo 2 del grupo 1. Además, presenta un 9% menos de precisión en cuanto a predicción. Esto nos lleva a pensar que la incorporación de PC2 como variable descriptora no aporta ventajas, sino que produce el efecto contrario.

A partir de este primer modelo, se ajusta un segundo modelo incrementando el número de neuronas y de épocas de entrenamiento. Sin embargo, este modelo no logra superar los resultados del primero, ya que se tienen los mismos errores de predicción, pero una precisión menor. Con el tercer modelo, en el que se reduce el número de neuronas, pero se incrementa el número de épocas de entrenamiento se consigue errores menores y mayor precisión, siendo estos los mejores resultados de este grupo de modelos. Por último, se probó también un modelo con los mismos

parámetros que el modelo que dio mejores resultados en el primer grupo (100 neuronas y 100 épocas). En este caso, estos parámetros no consiguieron un resultado tan bueno como en el primer grupo. Esto puede deberse también a que incluir PC2 como variable descriptora afecta a la capacidad de aprendizaje de la red neuronal, dado que no aporta información útil.

Modelo	No. de neuronas	No. de épocas	RMSE	MAPE	R^2
1	50	40	0.09	0.16	0.56
2	75	50	0.09	0.16	0.55
3	50	100	0.07	0.11	0.72
4	100	100	0.08	0.13	0.66

Tabla 4. Resumen de modelos del grupo 2

En la **Figura 47** observamos cómo de cercanos son los valores estimados a los valores reales tras realizar la predicción con el conjunto de datos de prueba. Confirmamos así, que el mejor resultado para este grupo de modelos se obtiene con el modelo 3, con el que ve un menor desfase en el eje *y* de los valores predichos con respecto a los valores reales.

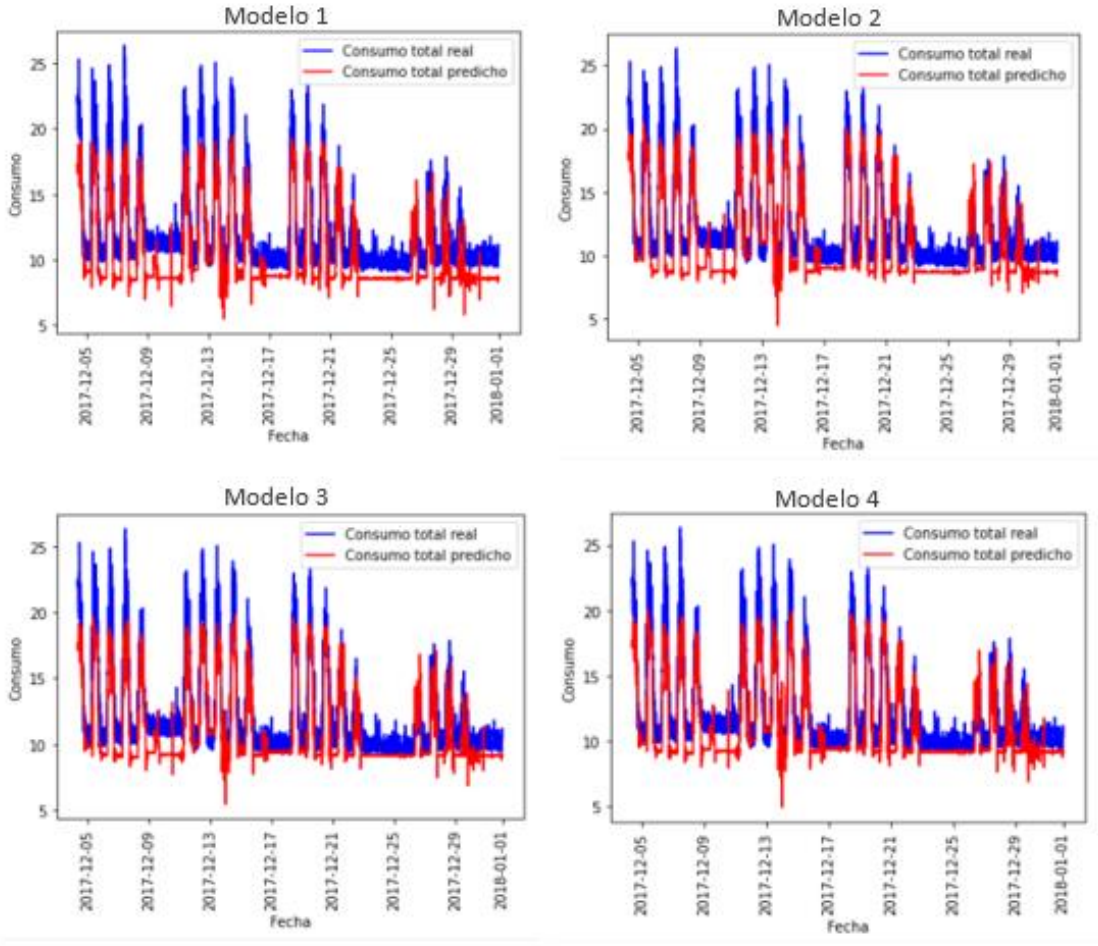

Figura 47. Valores reales y predichos de los modelos del grupo 2

Para terminar, en el tercer grupo se incluye como variables descriptoras las variables originales a partir de las cuales se generó PC1 a través de la técnica PCA. Por ello, este grupo también cuenta con 4 modelos, construido con los mismos parámetros que los modelos del grupo 1 para poder realizar una comparación y evaluar si el uso de las variables originales presenta mejores resultados que el uso de sólo PC1. En la **Tabla 5** se resume los resultados obtenidos con los modelos de este grupo.

Comparando los resultados de este grupo de modelos con los del primer grupo, podemos ver que los tres primeros modelos presentan menores errores y mayor precisión en este caso. Sin embargo, al incrementar el número de neuronas y el número de épocas de entrenamiento, aunque no hay mucha variación con respecto a las métricas de los demás modelos, no se trata del mejor. En este grupo, los mejores resultados se consiguen con el segundo modelo. La razón de que ocurra esto podría ser el hecho de que, al considerar todas las variables sin haber realizado extracción de características, la red neuronal tiene más información de la que aprender. No obstante, a diferencia de PC2, estas variables sí aportan información útil al proceso de aprendizaje. A pesar de esto, se observa un comportamiento similar a los casos anteriores, es decir, que un número de épocas de entrenamiento no lo suficientemente alto para el número de neuronas establecido reduce la precisión del modelo.

Modelo	No. de neuronas	No. de épocas	RMSE	MAPE	R^2
1	50	25	0.08	0.14	0.65
2	50	40	0.07	0.12	0.74
3	100	40	0.07	0.13	0.70
4	100	100	0.07	0.12	0.73

Tabla 5. Resumen de modelos del grupo 3

La exactitud de las predicciones con respecto a los valores reales para los modelos de este grupo se puede observar en la **Figura 48**.

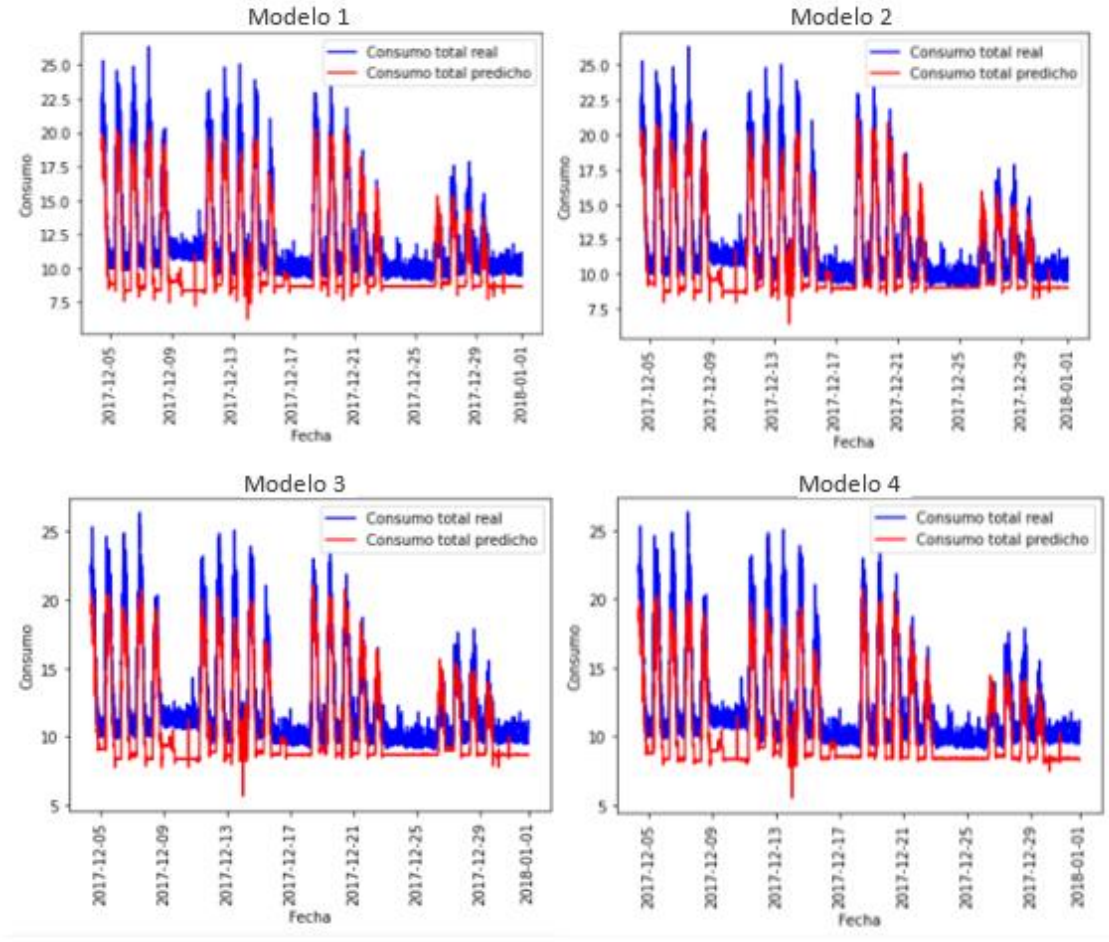

Figura 48. Valores reales y predichos del grupo de modelos 3

3.4 Interpretación de resultados

Tras haber evaluado los modelos de cada uno de los tres grupos propuestos, podemos finalmente proponer uno de ellos, puesto que, precisamente, ese es el objetivo de un proceso de obtención de modelos de predicción. Para ello, tenemos en cuenta tanto los resultados obtenidos y valorados en el punto anterior, como el proceso llevado a cabo para conseguirlos.

Partiendo de los mejores modelos obtenidos en cada grupo, resumidos en la Tabla 6, realizamos una comparación entre ellos, para determinar cuál es el más adecuado. En el primer grupo de modelos se obtuvo mejores resultados con un modelo de 100 neuronas y 100 épocas de entrenamiento, mientras que en el segundo grupo los mejores resultados se obtuvieron con un modelo de 50 neuronas y 100 épocas de entrenamiento.

Sin embargo, el modelo del primer grupo presenta errores menores que el del segundo, así como una mayor precisión. De esto, podemos decir que la inclusión de PC2 como variable descriptora no aporta ventajas a la generación de un modelo óptimo. Incluso, se puede decir que afecta negativamente al proceso de aprendizaje, ya que provoca que la red neuronal tenga en cuenta mayor cantidad de datos de entrada, datos que no aportan información relevante puesto que PC2 no tiene una correlación significativa con la variable target (*Skyspark*). Por lo tanto, de esta primera comparación se puede seleccionar al modelo del primer grupo como la mejor opción.

Por otro lado, en el tercer grupo de modelos se obtuvo mejores resultados con el segundo modelo, compuesto por 50 neuronas y entrenado en 40 épocas. Este resultado difiere de lo que se había obtenido en el primer grupo, donde se crearon modelos con exactamente los mismos parámetros para evaluar la diferencia entre aplicar extracción de características y usar las variables de forma directa. Este modelo permite obtener predicciones tan precisas como el mejor modelo del primer grupo y con el mismo valor de RMSE, aunque con un MAPE superior. Esto ocurre porque se han considerado las variables a las que se le ha aplicado PCA, obteniendo con ello PC1. Por esta razón, estas variables están lo suficientemente correlacionadas con *Skyspark* para aportar información relevante en el proceso de aprendizaje. Además, es posible que, al incluir una mayor cantidad de variables, se haya necesitado menos cantidad de neuronas y de ciclos de entrenamiento para conseguir un modelo con un desempeño similar al mejor modelo del primero grupo. No obstante, al incrementar el número de neuronas y de épocas se puede producir overfitting, provocando predicciones menos precisas. Esto se refleja en los casos en los que la función de pérdida para el conjunto de datos de prueba presenta valores por debajo de los de la función de pérdida para el conjunto de datos de entrenamiento (ver **Figuras 37, 40, 41, 43**) [67], [92].

Así, este modelo del tercer grupo parece una opción viable. Sin embargo, si revisamos la función de pérdida para este modelo, vemos que sufre variaciones para el conjunto de datos de prueba, lo que significa que no es realmente un modelo estable. Es decir, que en ciertos casos puede dar buenas predicciones, y en otros casos no. De esto podemos decir que, si bien considerar las variables iniciales como

descriptoras puede dar buenos resultados, no nos asegura que esto ocurra en todos los casos y, por lo tanto, es preferible generar modelos con un conjunto de variables analizadas y tratadas previamente.

Una vez comparados los mejores modelos de cada grupo, podemos proponer como el más adecuado al modelo del primer grupo. A pesar de que con este modelo se obtienen los mejores resultados, aún se podría intentar optimizarlo más. Además, la precisión según la métrica R^2, en este caso de un 74%, puede verse afectada por la variabilidad de los datos y el ancho del intervalo de precisión, por lo que el uso de otras métricas de evaluación también ayudaría en el análisis de la calidad de los resultados.

Grupo - Modelo	No. de neuronas	No. de épocas	RMSE	MAPE	R^2
1 - 4	100	100	0.07	0.10	0.74
2 - 3	50	100	0.07	0.11	0.72
3 - 2	50	40	0.07	0.12	0.74

Tabla 6. Resumen de mejores modelos de cada grupo

Cabe también mencionar que se ha notado que no por incorporar más neuronas en un modelo, la capacidad de predicción de este mejorará. Como se ha visto, si no se establece un número de épocas de entrenamiento adecuado para el número de neuronas elegido, se conseguirá el efecto contrario. Si establecen muy pocas épocas, la red no aprende lo suficiente, produciéndose un underfitting, como en el caso del modelo inicial de 5 neuronas y 5 épocas de entrenamiento, donde los valores de la función de pérdida para el conjunto de datos de prueba quedan muy por encima de los valores para el conjunto de datos de entrenamiento [67]. Por el contrario, si se establecen muchas épocas, el modelo empieza a memorizar y deja de aprender, produciéndose un overfitting, como se observa en los casos en los que la función de pérdida para el conjunto de datos de prueba va por debajo de la del conjunto de datos de entrenamiento [67], [92].

Capítulo 4.
Conclusiones y líneas futuras

En este último capítulo se expresan las reflexiones finales acerca del desarrollo del proyecto. Por ello, se tiene dos secciones. En la primera de ellas, se presentan las conclusiones a las que se ha llegado tras el proceso de análisis de datos y obtención de modelos de predicción utilizando el AA, destacando los aspectos más notables del desarrollo del proyecto. Por otro lado, la segunda sección presenta algunas opciones para conseguir mejores resultados, así como de qué manera se podría extender este proyecto y su aplicación.

4.1 Conclusiones

El cambio vertiginoso que se está produciendo actualmente en las sociedades, en gran parte gracias a las nuevas tecnologías, acarrea también una serie de problemas, y qué mejor que aprovechar esta tecnología para intentar solucionarlos. Uno de los escenarios que se está viviendo, y de la que todos deberíamos ser conscientes, es el incremento del consumo de energía eléctrico en los últimos tiempos. Este consumo masivo conlleva, por un lado, un mayor gasto económico por parte de los usuarios y, por otro lado, una repercusión en estado medioambiental. Ante esta situación, se han tomado diversas medidas e iniciativas, como el cambio de hora, la "Hora del Planeta", y el uso de fuentes de energía renovables. Sin embargo, gracias a la evolución de la tecnología y de la informática, se han desarrollado otras técnicas que pueden ayudar a abordar este problema desde otro enfoque.

Es en este contexto donde se presentan los Edificios Inteligentes, en los cuales se puede realizar una gestión del consumo energético gracias a su estructura provista de sensores, actuadores, y sistemas adaptados para ello. Esta gestión, sin embargo, debe realizarse de forma eficiente para que se pueda conseguir un verdadero ahorro de energía, así como un servicio que ofrezca comodidad a los usuarios. Una de las opciones para lograr esto implica la aplicación del *Soft Computing*, rama de la IA, y en concreto, un método bastante utilizado: AA.

Gracias a este proyecto nos hemos introducido en el mundo del AA, conociendo cada parte de un estudio de estas características, y entendiendo lo que implica. Teniendo en cuenta que el AA puede ser aplicado en multitud de ámbitos, como se está viendo en la actualidad, uno de los campos en los que se le está dando mayor cabida es, precisamente, el de la predicción del consumo energético. Como hemos

visto en el desarrollo de este proyecto, el AA intenta replicar el proceso de aprendizaje humano, reconociendo características de los datos recibidos. Otra razón por la que usar AA, es que permite realizar predicciones con series temporales, como es el caso de este proyecto.

Desde el punto de vista de organización, a lo largo de este trabajo se ha visto la importancia de usar una metodología para el desarrollo de un proyecto. En este caso, al tratarse de un proyecto de análisis de datos, la elección de CRISP-DM ha resultado ser una decisión acertada. Debido a la estructura de esta metodología, se ha podido diferenciar de forma clara cada uno de los pasos a seguir en el desarrollo del proyecto, así como planificar las tareas necesarias. Gracias a esto, ha sido posible seguir un orden, permitiéndonos trabajar de manera eficiente. Además, se ha comprobado la flexibilidad que aporta esta metodología en cuanto al seguimiento de las etapas, ya que en ocasiones ha sido necesario volver a una fase anterior. Por ejemplo, al momento de ajustar los parámetros de nuestros modelos era necesario verificar las métricas de calidad y evaluar los resultados, para poder determinar los parámetros para un nuevo modelo.

En cuanto al desarrollo concreto a llevar a cabo con el objetivo de obtener y proponer un modelo de predicción, se ha podido explorar diferentes técnicas de AA, así como las técnicas estadísticas en las que se apoyan. Cada uno de estos procedimientos se ha explicado, primeramente, de forma teórica, para luego ser aplicados en el análisis con el conjunto de datos seleccionado. De esta forma, se tiene claro el concepto de cada técnica y la razón de ser empleada, lo que nos permite interpretar los resultados obtenidos en el caso práctico.

Por otro lado, y gracias a haber comprendido nociones que nos ayudan a analizar las relaciones y dependencias entre variables, notamos la importancia de entender nuestros datos, para poder prepararlos antes de pasar a la fase de modelado. Esto nos ayuda a conseguir un conjunto de datos limpio y de esta forma, obtener mejores modelos. Como hemos visto, gracias a los coeficientes de correlación y los modelos de regresión lineal analizamos las relaciones entre nuestras variables, y así poder determinar cuáles son realmente útiles y cuáles no. A partir de esto, hemos realizado extracción de características aplicando PCA. Los beneficios de esta técnica se han reflejado al comparar los modelos en los que se usaba como variable descriptora un Componente Principal con los modelos en los que se usaban de forma directa las variables originales. Como era de esperarse, al aplicar PCA se obtuvieron modelos más estables. Otro de los aspectos a tener en cuenta es la relación temporal de las variables consigo mismas. Para esto, nos apoyamos en los conceptos de autocorrelación y en modelos ARIMA, gracias a los cuales obtuvimos la ventana de tiempo óptima para realizar las predicciones.

Con respecto a los tipos de modelo a generar, se exploró las ventajas de usar Redes Neuronales Recurrentes y, en concreto, redes LSTM. En la fase de modelado se puso en práctica el uso de estas redes, así como el ajuste de sus parámetros. En este proceso obtuvimos resultados que nos han permitido entender el comportamiento de los modelos de acuerdo a sus configuraciones, y así, poder extraer conclusiones. Por ejemplo, se notó que realizar el entrenamiento con un número elevado de neuronas, pero con pocas épocas de entrenamiento, provoca que la red no ajuste sus parámetros correctamente. Por el contrario, si el número de neuronas es pequeño,

pero se le aplica muchas épocas de entrenamiento, se produce el fenómeno de overfitting, es decir, la red empieza a memorizar y deja de aprender. En ambos casos, la capacidad de predicción del modelo se reduce.

Cabe mencionar que para poder evaluar estos modelos se han utilizado tres métricas de calidad: RMSE, MAPE y R^2, ya que con una única métrica la comparación entre modelos habría sido menos acertada. Se realizó, en primer lugar, una comparación entre los modelos de cada uno de los tres grupos propuestos, para luego contrastar los mejores modelos de cada grupo. De esta forma, se propuso como mejor opción un modelo de 100 neuronas con 100 épocas de entrenamiento, en el que se tenían como variable descriptora únicamente el primer Componente Principal (PC1).

Para terminar, se ha de destacar que al desarrollar este proyecto se ha podido contemplar algunas de las ventajas que aporta el AA, como la capacidad de tratar gran cantidad de datos, poder procesar eventos ordenados en el tiempo, y poder generar modelos de predicción de forma casi autónoma sin nuestra participación en el proceso de entrenamiento. A pesar de esto, la calidad y la precisión de los modelos obtenidos dependen también de otros factores como los recursos disponibles, tanto tecnológicos como temporales, sobre todo en casos reales y cuando se trata conjuntos de datos de mayor tamaño. Además, el ajuste de parámetros puede ser un proceso largo, ya que se trata de ir realizando multitud de pruebas hasta dar con el modelo óptimo. Esto se refleja en el hecho de que el mejor modelo en este estudio tiene un 74% de precisión, valor que podría ser mejorado, como se propone en el apartado de líneas futuras.

4.2 Líneas futuras

Un proyecto de AA puede dejar diferentes vías abiertas en las que seguir trabajando, dependiendo del entorno donde se desarrolle. Por ejemplo, si es un proyecto desarrollado para una empresa y que se va a poner en producción, será necesario definir una serie de planes para realizar una monitorización y llevar un seguimiento de los resultados, comprobar si el sistema se comporta según lo esperado, o realizar un mantenimiento del mismo. Aunque ese no es el caso de este trabajo, a continuación, se proponen posibles mejoras y ampliaciones al proyecto desarrollado.

En primera instancia, dado que la precisión del modelo propuesto podría mejorarse, se propone la aplicación de otras técnicas para el preparado de datos. De esta forma, se podría tener una visión más clara de la información a tratar y transformar el conjunto de datos según sea necesario para pasar a la fase de modelado. Por ejemplo, se podría realizar una correlación cruzada o cross-correlation, para determinar la correlación temporal entre variables.

Por otro lado, las redes neuronales LSTM no son la única opción para la generación de modelos de predicción. Se propone realizar una investigación sobre otro tipo técnicas de obtención de modelos que puedan ser utilizados con series temporales o, incluso, otro tipo de redes neuronales basadas en LSTM, como LSTM-CNN. Tras explorar las distintas opciones y las ventajas que ofrecen, se producirían un conjunto de modelos de estimación, para ser posteriormente valorados y ver si se obtienen mejores resultados.

Otro de los aspectos que puede ser estudiado es la inclusión de variables climatológicas en nuestro conjunto de datos. Al considerar este tipo de variables, se tendría más información de la que nuestro modelo pueda aprender. Esto se debe a que factores climatológicos, como la temperatura o la humedad, afectan a nuestro consumo de energía, pues en función de esos valores se hace uso en mayor o menor medida de aparatos como la calefacción o el aire acondicionado. Además, estas variables tienen una temporalidad marcada, ya que dependen fuertemente de la estación y época del año. Sin embargo, habría que estudiar si estas variables aportan realmente información útil.

Por último, una vez estudiadas y realizadas, si se da el caso, las mejoras sugeridas previamente, se plantea la aplicación del modelo que se termine por proponer en un sistema de un entorno real. En este caso sí se realizaría un plan de monitorización de los resultados para, de esta forma, poder evaluar el rendimiento y calidad de nuestro modelo cuando se realiza una ingesta de datos en tiempo real. Además, para agilizar esta tarea, se podría desarrollar una aplicación que nos permita llevar ese control sobre el sistema, pudiendo visualizar valores, gráficas y estadísticas.

Bibliografía

[1] Building Efficiency Initiative, "What is a Smart Building?", buildingefficiencyinitiative.org. [Online]. Disponible en: https://buildingefficiencyinitiative.org/articles/what-smart-building [Último acceso 18/sept./2021].

[2] V. Advani, "What is Machine Learning? How Machine Learning Works and future of it?", mygreatlearning.com. [Online]. Disponible en: https://www.mygreatlearning.com/blog/what-is-machine-learning [Último acceso 18/sept./2021].

[3] IBM Cloud Education, "Machine Learning", ibm.com. [Online]. Disponible en: https://www.ibm.com/es-es/cloud/learn/machine-learning [Último acceso 18/sept./2021].

[4] Peixeiro, "The Complete Guide to Time Series Analysis and Forecasting", towardsdatascience.com. [Online]. Disponible en: https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775 [Último acceso 18/sept./2021].

[5] A. Hayes, "What Is a Time Series?", investopedia.com. [Online]. Disponible en: https://www.investopedia.com/terms/t/timeseries.asp [Último acceso 18/sept./2021].

[6] G. Li *et al.*, "Data partitioning and association mining for identifying VRF energy consumption patterns under various part loads and refrigerant charge conditions", *Applied Energy*, vol. 185, no. 1, pp. 846-86, Enero., 2017. [Online]. Disponible en: https://doi.org/10.1016/j.apenergy.2016.10.091

[7] A. González-Vidal, A. P. Ramallo-González, F. Terroso-Sáenz, y A. Skarmeta. "Data driven modeling for energy consumption prediction in smart buildings", in *2017 IEEE International Conference on Big Data (Big Data)*, pp. 4526-4569. [Online]. Disponible en: 10.1109/BigData.2017.8258499

[8] T. Le, M. T. Vo, T. Kieu, E. Hwang, S. Rho, y S. W. Baik, "Multiple Electric Energy Consumption Forecasting Using a Cluster-Based Strategy for Transfer Learning in Smart Building", *Sensors*, vol. 20, no. 9, pp. 20092668-1–20092668-17, Mayo, 2020. [Online]. Disponible en: https://doi.org/10.3390/s20092668

[9] Y. Quintero, D. Ardila, E. Camargo, F. Rivas, y J. Aguilar, "Machine learning models for the prediction of the SEIRD variables for the COVID-19 pandemic based on a deep dependence analysis of variables", *Computers in Biology and Medicine*, vol. 134, no. 104500, pp. 1-19, Mayo, 2021. [Online]. Disponible en: https://doi.org/10.1016/j.compbiomed.2021.104500

[10] Data Science Process Alliance, "What is CRISP-DM?", datascience-pm.com. [Online]. Disponible en: https://www.datascience-pm.com/crisp-dm-2 [Último acceso: 18/sept./2021]

[11] P. Chapman *et al.*, "The CRISP-DM Process Model", keithmccormick.com. [Online]. Disponible en: https://keithmccormick.com/wp-content/uploads/CRISP-DM%20No%20Brand.pdf [Último acceso: 18/sept./2021].

[12] N. Leaper, "A visual guide to CRISP-DM methodology", exde.wordpress.com. [Online]. Disponible en: https://exde.files.wordpress.com/2009/03/crisp_visualguide.pdf [Último acceso: 18/sept./2021].

[13] J. Villena Román, "CRISP-DM: La metodología para poner en orden los proyectos", sngular.com. [Online]. Disponible en: https://www.sngular.com/es/data-science-crisp-dm-metodologia [Último acceso: 18/sept./2021].

[14] Laerd Statistics, "Pearson Product-Moment Correlation", statistics.laerd.com. [Online]. Disponible en: https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide-2.php [Último acceso: 18/sept./2021].

[15] J. Brownlee, "How to Calculate Correlation Between Variables in Python", machinelearningmastery.com. [Online]. Disponible en: https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables [Último acceso: 18/sept./2021].

[16] Minitab, LLC., "A comparison of the Pearson and Spearman correlation methods", minitab.com. [Online]. Disponible en: https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/supporting-topics/basics/a-comparison-of-the-pearson-and-spearman-correlation-methods [Último acceso: 18/sept./2021].

[17] QuestionPro, "¿Qué es el coeficiente de correlación de Pearson?", questionpro.com. [Online]. Disponible en: https://www.questionpro.com/blog/es/coeficiente-de-correlacion-de-pearson [Último acceso: 18/sept./2021].

[18] A. Hayes, "Multiple Linear Regression (MLR)", investopedia.com. [Online]. Disponible en: https://www.investopedia.com/terms/m/mlr.asp [Último acceso: 18/sept./2021].

[19] J. A. Rodrigo, "Introducción a la Regresión Lineal Múltiple", cienciadedatos.net. [Online]. Disponible en: https://www.cienciadedatos.net/documentos/25_regresion_lineal_multiple.html [Último acceso: 18/sept./2021].

[20] J. Frost, "Spearman's Correlation Explained", statisticsbyjim.com. [Online]. Disponible en: https://statisticsbyjim.com/basics/spearmans-correlation [Último acceso: 18/sept./2021].

[21] I. Weir, "Spearman's correlation", statstutor.ac.uk. [Online]. Disponible en: https://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf [Último acceso: 18/sept./2021].

[22] Laerd Statistics, "Spearman's Rank-Order Correlation", statistics.laerd.com. [Online]: Disponible en: https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php [Último acceso: 18/sept./2021].

[23] QuestionPro, "¿Qué es el coeficiente de correlación de Spearman?", questionpro.com. [Online]. Disponible en: https://www.questionpro.com/blog/es/coeficiente-de-correlacion-de-spearman [Último acceso: 18/sept./2021].

[24] L. Li, "Principal Component Analysis for Dimensionality Reduction", towardsdatascience.com. [Online]. Disponible en: https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad [Último acceso: 18/sept./2021].

[25] J. I. Bagnato, "Comprende Principal Component Analysis", aprendemachinelearning.com. [Online]. Disponible en: https://www.aprendemachinelearning.com/comprende-principal-component-analysis [Último acceso: 18/sept./2021].

[26] C. Albon, “Feature Extraction With PCA”, chrisalbon.com. [Online]. Disponible en: https://chrisalbon.com/code/machine_learning/feature_engineering/feature_extraction_with_pca [Último acceso: 18/sept./2021].

[27] B. Mikulski, “PCA – how to choose the number of components?”, mikulskibartosz.name. [Online]. Disponible en: https://www.mikulskibartosz.name/pca-how-to-choose-the-number-of-components [Último acceso: 18/sept./2021].

[28] G. Toth, “Principal Components Analysis”, datasklr.com. [Online]. Disponible en: https://www.datasklr.com/principal-component-analysis-and-factor-analysis/principal-component-analysis [Último acceso: 18/sept./2021].

[29] J. A. Rodrigo, “PCA con Python”, cienciadedatos.net. [Online]. Disponible en: https://www.cienciadedatos.net/documentos/py19-pca-python.html [Último acceso: 18/sept./2021].

[30] J. M. Cortés Patiño, “Aplicación de series temporales en el monitoreo estructural”, Tesis de Grado, Instituto de Ingeniería, UNAM, México D.F., 2011. [Online]. Disponible en: http://www.ptolomeo.unam.mx:8080/jspui/bitstream/132.248.52.100/363/5/A5.pdf

[31] C. M. López, “Análisis de series temporales con R(III): Autocorrelación”, finanzaszone.com. [Online]. Disponible en: https://finanzaszone.com/analisis-y-prediccion-de-series-temporales-con-r-iii-autocorrelacion [Último acceso: 18/sept./2021].

[32] Z. West, “Autocorrelation of Time Series Data in Python”, alpharithms.com. [Online]. Disponible en: https://www.alpharithms.com/autocorrelation-time-series-python-432909 [Último acceso: 18/sept./2021].

[33] J. Brownlee, “A Gentle Introduction to Autocorrelation and Partial Autocorrelation”, machinelearningmastery.com. [Online]. Disponible en: https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation [Último acceso: 18/sept./2021].

[34] S. Prabhakaran, “ARIMA Model – Complete Guide to Time Series Forecasting in Python”, machinelearningplus.com. [Online]. Disponible en: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python [Último acceso: 18/sept./2021].

[35] R.J. Hyndman and G. Athanasopoulos, “ARIMA models” in *Forecasting: Principles and Practice*, 2nd edition. Victoria, Australia: O’Texts.com, 2018, ch.8, sec.8.5. [Online]. Disponible en: https://otexts.com/fpp2/non-seasonal-arima.html

[36] Ritvikmath, *Time Series Talk: ARIMA Model*, Jul. 12, 2019. [Video file]. Disponible en: https://www.youtube.com/watch?v=3UmyHed0iYE [Último acceso: 18/sept./2021].

[37] J. Brownlee, “11 Classical Time Series Forecasting Methods in Python (Cheat Sheet)”, machinelearningmastery.com. [Online]. Disponible en: https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet [Último acceso: 18/sept./2021].

[38] J. Brownlee, “How to Create an ARIMA Model for Time Series Forecasting in Python”, machinelearningmastery.com. [Online]. Disponible en: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python [Último acceso: 18/sept./2021].

[39] R.J. Hyndman and G. Athanasopoulos, "ARIMA models" in *Forecasting: Principles and Practice*, 2nd edition. Victoria, Australia: O'Texts.com, 2018, ch.8, sec.8.3. [Online]. Disponible en: https://otexts.com/fpp2/AR.html

[40] R.J. Hyndman and G. Athanasopoulos, "ARIMA models" in *Forecasting: Principles and Practice*, 2nd edition. Victoria, Australia: O'Texts.com, 2018, ch.8, sec.8.1. [Online]. Disponible en: https://otexts.com/fpp2/stationarity.html

[41] R.J. Hyndman and G. Athanasopoulos, "ARIMA models" in *Forecasting: Principles and Practice*, 2nd edition. Victoria, Australia: O'Texts.com, 2018, ch.8, sec.8.4. [Online]. Disponible en: https://otexts.com/fpp2/MA.html

[42] IBM Cloud Education, "Recurrent Neural Networks", ibm.com. [Online]. Disponible en: https://www.ibm.com/cloud/learn/recurrent-neural-networks [Último acceso: 18/sept./2021].

[43] Codificando Bits, *Introducción a las Redes Neuronales Recurrentes*, Jun. 8, 2019. [Video file]. Disponible en: https://www.youtube.com/watch?v=bKkjQx_PS_M [Último acceso: 18/sept./2021].

[44] A. Amidi and S. Amidi "Recurrent Neural Networks cheatsheet", stanford.edu. [Online]. Disponible en: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks [Último acceso: 18/sept./2021].

[45] J. Torres, "Redes Neuronales Recurrentes", torres.ai. [Online]. Disponible en: https://torres.ai/redes-neuronales-recurrentes [Último acceso: 18/sept./2021].

[46] Codificando Bits, *¿Qué es una red LSTM?*, Jul. 20, 2019. [Video file]. Disponible en: https://www.youtube.com/watch?v=1BubAvTVBYs [Último acceso: 18/sept./2021].

[47] M. Sotaquirá, "¿Qué son las redes LSTM?", codificandobits.com. [Online]. Disponible en: https://www.codificandobits.com/blog/redes-lstm [Último acceso: 18/sept./2021].

[48] C. Olah, "Understanding LSTM Networks", colah.github.io. [Online]. Disponible en: https://colah.github.io/posts/2015-08-Understanding-LSTMs [Último acceso: 18/sept./2021].

[49] X. Yuan, L. Li, and Y. Wang, "Nonlinear Dynamic Soft Sensor Modeling With Supervised Long Short-Term Memory Network", presented at IEEE Transactions on Industrial Informatics, pp. 1-1. [Online]. Disponible en: https://www.researchgate.net/publication/331421650_Nonlinear_Dynamic_Soft_Sensor_Modeling_With_Supervised_Long_Short-Term_Memory_Network

[50] J. Moody, "What does RMSE really mean?", towardsdatascience.com [Online]. Disponible en: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e [Último acceso: 18/sept./2021].

[51] S. Gupta, "RMSE: What does it mean?", medium.com. [Online]. Disponible en: https://medium.com/@mygreatlearning/rmse-what-does-it-mean-2d446c0b1d0e [Último acceso: 18/sept./2021].

[52] S. Gunjal, "What is Root Mean Square Error (RMSE)", Ene., 2021. [Discussion topic]. Disponible en: https://www.kaggle.com/general/215997 [Último acceso: 18/sept./2021].

[53] Aprende IA, "Evaluando el error en los modelos de regresión", aprendeia.com. [Online]. Disponible en: https://aprendeia.com/evaluando-el-error-en-los-modelos-de-regresion [Último acceso: 18/sept./2021].

[54] S. Glen, "Mean Absolute Percentage Error (MAPE)", statisticshowto.com. [Online]. Disponible en: https://www.statisticshowto.com/mean-absolute-percentage-error-mape [Último acceso: 18/sept./2021].

[55] GEO Tutoriales, "Error Porcentual Absoluto Medio (MAPE) en un Pronóstico de Demanda", gestiondeoperaciones.net. [Online]. Disponible en: https://www.gestiondeoperaciones.net/proyeccion-de-demanda/error-porcentual-absoluto-medio-mape-en-un-pronostico-de-demanda [Último acceso: 18/sept./2021].

[56] Z. Bobbitt, "What is Considered a Good Value for MAPE?", statology.org. [Online]. Disponible en: https://www.statology.org/what-is-a-good-mape [Último acceso: 18/sept./2021].

[57] J. Fernando, "R-Squared", investopedia.com. [Online]. Disponible en: https://www.investopedia.com/terms/r/r-squared.asp [Último acceso: 18/sept./2021].

[58] D. Bhalla, "Difference Between Adjusted R-Squared and R-Squared", listendata.com. [Online]. Disponible en: https://www.listendata.com/2014/08/adjusted-r-squared.html [Último acceso: 18/sept./2021].

[59] S. Glen, "Coefficient of Determination (R Squared): Definition, Calculation", statisticshowto.com. [Online]. Disponible en: https://www.statisticshowto.com/probability-and-statistics/coefficient-of-determination-r-squared [Último acceso: 18/sept./2021].

[60] J. M. Ashfaque, *Electricity France*, UK: Kaggle, 2020. [Online]. Disponible en: https://www.kaggle.com/ukveteran/electricity-france [Último acceso: 18/sept./2021].

[61] S. Houidi, D. Fourer, H. Ben Attia Sethom, F. Auger, and L. Miègeville, *Powers_time_series*, FR: Mendeley Data, 2019. [Online]. Disponible en: https://data.mendeley.com/datasets/wpv89cgrm4/1 [Último acceso: 18/sept./2021].

[62] J. Langevin, *Long-term data on 3 office Air Handling Units*, CA, USA: OpenEI, 2015. [Online]. Disponible en: https://data.openei.org/submissions/636 [Último acceso: 18/sept./2021].

[63] J. Langevin, *Long-term energy consumption & outdoor air temperature for 11 commercial buildings*, CA, USA: OpenEI, 2015. [Online]. Disponible en: https://data.openei.org/submissions/603 [Último acceso: 18/sept./2021].

[64] Erik, *Home Energy Consumption*: data.world, inc, 2016. [Online]. Disponible en: https://data.world/ewood/home-energy-consumption [Último acceso: 18/sept./2021].

[65] B. Doherty y K. Trenbath, *Raw_Data*, CO, USA: Mendeley Data, 2019. [Online]. Disponible en: https://data.mendeley.com/datasets/g392vt7db9/1 [Último acceso: 18/sept./2021].

[66] M. Hamdy, S. Attia y S. S. Karlsen, *Tariff_computation_template*, Norway: Mendeley Data, 2019. [Online]. Disponible en: https://data.mendeley.com/datasets/45psvj6h3m/1 [Último acceso: 18/sept./2021].

[67] J. Brownlee, "How to Diagnose Overfitting and Underfitting of LSTM Models", machinelearningmastery.com. [Online]. Disponible en: https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/ [Último acceso: 27/sept./2021].

[68] The pandas development team, "User Guide", pandas.pydata.org. [Online]. Disponible en: https://pandas.pydata.org/docs/user_guide/index.html [Último acceso: 18/sept./2021].

[69] The NumPy community, "NumPy user guide", numpy.org. [Online]. Disponible en: https://numpy.org/doc/stable/user/index.html [Último acceso: 18/sept./2021].

[70] F. Pedregosa *et al*., "Scikit´learn: Machine Learning in Python", *JMLR*, vol. 12, no. 85 pp. 2825-2830, Oct., 2011. [Online]. Disponible en: https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf [Último acceso: 18/sept./2021].

[71] F. Pedregosa *et al*., "API Reference", scikit-learn.org. [Online]. Disponible en: https://scikit-learn.org/stable/modules/classes.html [Último acceso: 18/sept./2021].

[72] J. Perktold, S. Seabold, J. Taylor and statsmodels-developers, "API Reference", statsmodels.org. [Online]. Disponible en: https://www.statsmodels.org/stable/api.html#graphics [Último acceso: 18/sept./2021].

[73] T. G. Smith, "API Reference", alkaline-ml.com. [Online]. Disponible en: http://alkaline-ml.com/pmdarima/modules/classes.html#api-ref [Último acceso: 18/sept./2021].

[74] Keras team, "About Keras", keras.io. [Online]. Disponible en: https://keras.io/about [Último acceso: 18/sept./2021].

[75] Keras team, "Keras API reference", keras.io. [Online]. Disponible en: https://keras.io/api [Último acceso: 18/sept./2021].

[76] Jupyter Team, "The Jupyter Notebook", jupyter-notebook.readthedocs.io. [Online]. Disponible en: https://jupyter-notebook.readthedocs.io/en/stable/notebook.html [Último acceso: 18/sept./2021].

[77] Agilis IT Australia, "Online syntax hightlighting for the masses!". [Online]. Disponible en: https://tohtml.com

[78]] A. C. Müller y S. Guido, *Introduction to Machine Learning with Python: A guide for Data Scientists*, CA, USA: O'Reilly Media, Inc., 2016.

[79] Bobain y a_k_v, "Something like:", Nov. 15, 2018. [Comment in a forum]. Disponible en: https://stackoverflow.com/a/53318677 [Último acceso: 18/sept./2021]

[80] S. Ranjan, "Python | Pandas dataframe.corr()", geeksforgeeks.org. [Online]. Disponible en: https://www.geeksforgeeks.org/python-pandas-dataframe-corr [Último acceso: 18/sept./2021].

[81] M. Stojiljković, "NumPy, SciPy, and Pandas: Correlation With Python", realpython.com. [Online]. Disponible en: https://realpython.com/numpy-scipy-pandas-correlation-python [Último acceso: 18/sept./2021].

[82] The pandas development team, "pandas.DataFrame.corr", pandas.pydata.org. [Online]. Disponible en: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html [Último acceso: 18/sept./2021].

[83] M. Peixeiro, "The Complete Guide to Linear Regression in Python", towardsdatascience.com. [Online]. Disponible en: https://towardsdatascience.com/the-complete-guide-to-linear-regression-in-python-3d3f8f06bf8 [Último acceso: 18/sept./2021].

[84] F. Pedregosa *et al*, "Linear Models", scikit-learn.org. [Online]. Disponible en: https://scikit-learn.org/stable/modules/linear_model.html [Último acceso: 18/sept./2021].

[85] J. Perktold, S. Seabold, J. Taylor and statsmodels-developers, "Ordinary Least Squares", www.statsmodels.org. [Online]. Disponible en: https://www.statsmodels.org/stable/examples/notebooks/generated/ols.html [Último acceso: 18/sept./2021].

[86] F. Pedregosa *et al*, "sklearn.decomposition.PCA", scikit-learn.org. [Online]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html [Último acceso: 18/sept./2021].

[87] J. Perktold, S. Seabold, J. Taylor and statsmodels-developers, "statsmodels.graphics.tsaplots.plot_acf", www.statsmodels.org. [Online]. Disponible en: https://www.statsmodels.org/stable/generated/statsmodels.graphics.tsaplots.plot_acf.html [Último acceso: 18/sept./2021].

[88] J. Perktold, S. Seabold, J. Taylor and statsmodels-developers, "statsmodels.graphics.tsaplots.plot_pacf", www.statsmodels.org. [Online]. Disponible en: https://www.statsmodels.org/stable/generated/statsmodels.graphics.tsaplots.plot_pacf.html [Último acceso: 18/sept./2021].

[89] T. G. Smith, "pmdarima.arima.auto_arima", alkaline-ml.com. [Online]. Disponible en: https://alkaline-ml.com/pmdarima/1.0.0/modules/generated/pmdarima.arima.auto_arima.html [Último acceso: 18/sept./2021].

[90] S. Pulagam, "Time Series forecasting using Auto ARIMA in Python", towardsdatascience.com. [Online]. Disponible en: https://towardsdatascience.com/time-series-forecasting-using-auto-arima-in-python-bb83e49210cd [Último acceso: 18/sept./2021].

[91] J. Brownlee, "How to Develop Convolutional Neural Network Models for Time Series Forecasting", machinelearningmastery.com. [Online]. Disponible en: https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/ [Último acceso: 18/sept./2021].

[92] J. Brownlee, "Multivariate Time Series Forecasting with LSTMs in Keras", machinelearningmastery.com. [Online]. Disponible en: https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/ [Último acceso: 18/sept./2021].

[93] J. Brownlee, "Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras", machinelearningmastery.com. [Online]. Disponible en: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ [Último acceso: 18/sept./2021].

[94] V. Lendave, "How To Do Multivariate Time Series Forecasting Using LSTM", analyticsindiamag.com. [Online]. Disponible en: https://analyticsindiamag.com/how-to-do-multivariate-time-series-forecasting-using-lstm [Último acceso: 18/sept./2021].

[95] Keras team, "Models API", keras.io. [Online]. Disponible en: https://keras.io/api/models [Último acceso: 18/sept./2021].

Apéndice A. Tecnologías utilizadas

Para desarrollar este proyecto nos hemos valido de los siguientes recursos:

Recursos Hardware:

- Un ordenador y los periféricos necesarios (monitor y ratón).

Recursos Software:

- Sistema operativo *Windows 10*.

- Fuentes de datos abiertas [60]-[66]: *Kaggle*, *Mendeley*, *OpenEI* y data.*world*.

- Lenguaje de programación Python 3. Se ha elegido este lenguaje debido a que cuenta con bibliotecas específicas para el análisis de datos, en las que se proporciona los tipos de datos necesarios y se implementan algoritmos de técnicas estadísticas y de Aprendizaje Automático. A continuación, se describen las bibliotecas utilizadas:

    - ***pandas*** [68]: Esta biblioteca proporciona una estructura en la que se representan los conjuntos de datos a analizar, así como métodos para trabajar con ella. Esta estructura se llama *DataFrame* y algunas funciones de las librerías estadísticas requieren que sus datos de entrada sean de este tipo de datos.

    - ***numpy*** [69]: Esta biblioteca cuenta con un tipo de datos llamado *array*, así como los métodos necesarios para operar con ellos. El *array* de *numpy* es la estructura base para la biblioteca *scikit-learn*, explicada más adelante.

    - ***scikit-learn*** [70], [71]: Esta biblioteca provee un gran número de funciones que implementan algoritmos estadísticos y de Aprendizaje Automático. De esta biblioteca se ha usado el módulo *preprocessing* para la normalización de datos, el módulo *decomposition* para aplicar Principal Component Analysis (PCA), y el módulo *metrics* para calcular las métricas de calidad una vez hechas las predicciones.

      sStatsmodels [72]: Esta biblioteca proporciona funciones que implementan algoritmos de distintas técnicas estadísticas, como la estimación de modelos, la ejecución de pruebas y el análisis estadístico de datos. En este caso, hemos hecho uso del módulo *api* para la

generación de modelos de Regresión Lineal Múltiple, el módulo *graphics* para las gráficas de autocorrelaciones, y el módulo *tsa* para la descomposición de la serie temporal y observar la tendencia y la estacionalidad.

- ***pmdarima*** [73]: Esta biblioteca también pone a nuestra disposición un conjunto de funciones estadísticas, en concreto, para tratar series temporales. En este proyecto se ha usado el módulo *arima* de esta librería para efectuar el test de Dickey-Fuller aumentado y para la construcción de los modelos ARIMA.

- ***keras*** [74], [75]: Se trata de una API de la biblioteca de código abierto TensorFlow, que proporciona gran cantidad de soluciones de Aprendizaje Automático. En este caso, hemos utilizado los módulos *models* y *layers* para construir y entrenar nuestros modelos de estimación, así como para realizar predicciones.

- Entorno de trabajo *Jupyter Notebook* [76]: Se trata de tipo de documento que permite incorporar de forma conjunta documentación y código fuente, así como la ejecución del mismo, a través de una aplicación web. Estos notebooks pueden ser compartidos entre usuarios y descargados en distintos formatos. Además, soporta varios lenguajes de programación, entre ellos, *Python*.

- Procesador de Texto *Microsoft Word*.

- Resaltador de sintaxis (syntax highlighting) online [77] para insertar código fuente con formato en el presente documento.

Cabe mencionar que, antes de iniciar el desarrollo del proyecto, se ha revisado el libro de A.C. Müller y S. Guido [78], en el que se explora la aplicación de distintas técnicas de Aprendizaje Automático con Python 3 y usando Jupyter Notebooks. De esta forma, nos hemos familiarizado con algunas de las librerías previamente explicadas.

Apéndice B. Código fuente

A continuación, se adjunta el código fuente utilizado para desarrollar este proyecto, presentado en distintos puntos y en el orden en el que se ha ejecutado.

- Importación de bibliotecas.

```
#Import libraries
%pylab inline
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import r2_score
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plotpacf
from statsmodels.tsa.seasonal import seasonal_decompose
from pmdarima.arima import ADFTest
from pmdarima.arima import auto_arima
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.core import Dropout
from keras.layers.core import Dropout
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import r2_score
from keras.layers.core import Dropout
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import r2_score
```

- Carga inicial del dataset.

```
# Load de dataset joining Date and Time attributes
df_ini = pd.read_csv('Raw_Data.csv', sep=',', header=0,
parse_dates=[['Date', 'Time']])
display(df_ini)
```

- Comprobación de columnas con valores 0.

```
# Check if the following columns values are all 0
display(df_ini.loc[df_ini['Conference.Podium.Equip'] != 0])
display(df_ini.loc[df_ini['Exercise.Fans'] != 0])
display(df_ini.loc[df_ini['Auto.Door.Opener'] != 0])
display(df_ini.loc[df_ini['Microphone.Charger'] != 0])
display(df_ini.loc[df_ini['Treadmill.and.Elliptical'] != 0])
display(df_ini.loc[df_ini['AV.Controller.1'] != 0])
display(df_ini.loc[df_ini['Refrigerator'] != 0])
display(df_ini.loc[df_ini['Central.Monitoring.Station.TVs'] != 0])
display(df_ini.loc[df_ini['Transformer.Loss'] != 0])
```

- Eliminación de columnas innecesarias.

```
# Delete colums that will not be used
df = df_ini.drop(columns=['Month','Day','Hour','Minute'])
df = df.drop(columns = ['Prediction','Difference'])

# Delete columns with all values equal to 0
df = df.drop(columns=['Conference.Podium.Equip',
                      'Exercise.Fans','Auto.Door.Opener',
                      'Microphone.Charger',
                      'Treadmill.and.Elliptical',
                      'AV.Controller.1',
                      'Refrigerator',
                      'Central.Monitoring.Station.TVs',
                      'Transformer.Loss'])
df = df.set_index(['Date_Time'])

# Display cleaned dataset
display(df)
```

- Función que da formatea la visualización del dataset [79] para una mejor visualización de las matrices de coeficientes de correlación.

```
# ---------------------------------------------------------
# Function that rotates the header labels and limit the
# number of decimals to 2 for a better visualization
# ---------------------------------------------------------
def format_view(dataframe):
    return dataframe.round(2).style.set_table_styles(
    [dict(selector="th", props=[('width', '10px')]),
            dict(selector="th.col_heading",
                   props=[("writing-mode", "vertical-rl"),
                          ('transform', 'rotateZ(180deg)'),
                          ('height', '100px'),
                          ('vertical-align', 'top')])]
    ).format("{:.2f}")
```

- Correlación de Pearson [80]-[82].

```
# Calculate Pearson's correlation coefficients and format the view
pearsons_corr = df.corr(method='pearson', min_periods=1)
format_view(pearsons_corr)t("{:.2f}")
```

- Modelos de Regresión Lineal Múltiple [19], [83]-[85].

```
# ---------------------------------------------------------
# Function that builds a Multiple Linear Regression model
# for the variable indicated as dependent in the argument
# ---------------------------------------------------------
def multiple_linear_reg(var_dep):
    X=df.drop(columns=['Skyspark',var_dep])
    X=np.asarray(X)
    y=df[var_dep]
    X2 = sm.add_constant(X)
    ols_model = sm.OLS(y,X2)
    ols_res = ols_model.fit()
    display(ols_res.summary())
```

```
# Build a Linear Regression Model for each variable of the data set
multiple_linear_reg('AV.Controller')
multiple_linear_reg('Coffee.Maker')
multiple_linear_reg('Copier')
multiple_linear_reg('Desktop.Server')
multiple_linear_reg('Lamp')
multiple_linear_reg('Laptop')
multiple_linear_reg('Microwave')
multiple_linear_reg('Monitor')
multiple_linear_reg('Phone.Charger')
multiple_linear_reg('Printer')
multiple_linear_reg('Projector')
multiple_linear_reg('Toaster.Oven')
multiple_linear_reg('TV')
multiple_linear_reg('Video.Conference.Camera')
multiple_linear_reg('Water.Boiler')
```

- Correlación de Spearman [80]-[82].

```
# Calculate Spearman's correlation coefficients and format the view
spearman_corr = df.corr(method='spearman', min_periods=1)
format_view(spearman_corr)
```

- Eliminación de columnas no correlacionadas con la variable target (*Skyspark*).

```
# Delete variables which are no correlated to the Skyspark
df = df.drop(columns=['AV.Controller','Coffee.Maker',
                      'Desktop.Server','Headset','Microwave',
                      'Phone.Charger','Toaster.Oven',
                      'Video.Conference.Camera','Water.Boiler'])

# Display cleaned dataset
display(df)
```

- Principal Component Analysis (PCA) [24]-[29], [86].

```
# Normalize data
X = df.drop(columns=['Skyspark'])
scaler = StandardScaler()
scaled_X = scaler.fit_transform(X)

# Get Principal Components
pca = PCA(n_components = 7)
pca_X = pca.fit_transform(scaled_X)
pca_X = pd.DataFrame(pca_X, columns =
['PC1','PC2','PC3','PC4','PC5','PC6','PC7'])
pca_X = pd.concat([df_ini['Date_Time'], pca_X], axis = 1)
pca_X = pca_X.set_index(['Date_Time'])
print("Dataset con Componentes Principales")
display(pca_X)
```

```
# Get loadings by PC
print("Cargas de cada Componente Principal")
display(
    pd.DataFrame(
        pca.components_,columns = X.columns,
        index = ['PC1','PC2','PC3','PC4','PC5','PC6','PC7']
    )
)
```

```
# Get Explained variance by PC
exp_var = pca.explained_variance_
print("Variabilidad explicada por cada Componente Principal")
display(
    pd.DataFrame(
        [exp_var],
        columns = ['PC1','PC2','PC3','PC4','PC5','PC6','PC7']
    )
)
```

```
# Get Cumulative Explained Variance ratio
ratio_exp_var = pca.explained_variance_ratio_
cum_exp_var = np.cumsum(ratio_exp_var)
pc_order = [1,2,3,4,5,6,7]
print("Ratio de variabilidad explicada acumulada")
display(
    pd.DataFrame(
        [cum_exp_var],
        columns = ['PC1','PC2','PC3','PC4','PC5','PC6','PC7']
    )
)
pylab.plot(pc_order, cum_exp_var)
pylab.xlabel('Componentes principales')
pylab.ylabel('Ratio de variabilidad explicada acumulada')
pylab.show()
```

```
# Get Explained Variance ratio by PC
print("Ratio de variabilidad explicada por cada Componente Principal")
display(
    pd.DataFrame(
        [ratio_exp_var],
        columns = ['PC1','PC2','PC3','PC4','PC5','PC6','PC7']
    )
)
pylab.plot(pc_order, ratio_exp_var)
pylab.xlabel('Componentes principales')
pylab.ylabel('Ratio de variabilidad explicada')
pylab.show()
```

```
# Redimension data set after PCA - delete PC7
pca_df = pca_X.drop(columns = ['PC7'])
pca_df = pd.concat([df['Skyspark'], pca_df], axis = 1)
display(pca_df)
```

- Autocorrelaciones [32], [33], [87], [88].

```
# Select variable Skyspark
df_skyspark = pca_df[['Skyspark']]

# Simple autocorrelaction of Skyspark
print("Autocorrelación simple")
plot_acf(df_skyspark, lags = 90)
pylab.show()

# Partial autocorrelation of Skyspark
print("Autocorrelación parcial")
plot_pacf(df_skyspark, lags = 15)
pylab.show()
```

- Correlación de Pearson para los Componentes Principales.

```
# Calculate Pearson's correlation coefficients for PCs
pca_pearsons_corr = pca_df.corr(method='pearson', min_periods=1)
display(pca_pearsons_corr)
```

- Modelos ARIMA [34], [89], [90].

    - Test Dickey-Fuller aumentado para verificar estacionariedad de *Skyspark*.

```
# Perform Augmented Dickey-Fuller test for Skyspark
adf_test = ADFTest(alpha = 0.05)
adf_test_result = adf_test.should_diff(df_skyspark)
display(adf_test_result)
```

    - Modelo ARIMA generado automáticamente para *Skyspark*.

```
# Best ARIMA model for Skyspark generated automatically
arima_model_skyspark = auto_arima(df_skyspark, start_p = 0,
                                  start_q = 0, stationary=True
                                 )
display(arima_model_skyspark.summary())
```

    - Test Dickey-Fuller aumentado y modelos ARIMA para los CPs más correlacionados con *Skyspark* (PC1 y PC2).

```
# Perform Augmented Dickey-Fuller for PC1
df_pc1 = pca_df[['PC1']]
adf_test_result_pc1 = adf_test.should_diff(df_pc1)
print(adf_test_result_pc1)
```

```
# Best ARIMA model for PC1 generated automatically
arima_model_pc1 = auto_arima(df_pc1, start_p = 0,
                             start_q = 0, stationary=True
                            )
display(arima_model_pc1.summary())
```

```
# Perform Augmented Dickey-Fuller test for PC2
df_pc2 = pca_df[['PC2']]
adf_test_result_pc2 = adf_test.should_diff(df_pc2)
print(adf_test_result_pc2)
```

```
# Best ARIMA model for PC2 generated automatically
arima_model_pc2 = auto_arima(df_pc2, start_p = 0, start_q = 0,
stationary=True)
display(arima_model_pc2.summary())
```

- Modelado con redes neuronales LSTM [91]-[95].

    - Función para preparar los datos de entrada.

```
# ------------------------------------------------------------
# Function that splits output variables from input variables
# and adds previous observations values to the input
# ------------------------------------------------------------
def setup_input_output(data, lags):
    n_obs = lags + 1
    X, y = list(), list()
    for i in range (len(data)):
        end_ix = i + n_obs
        if end_ix > len(data):
            break
        data_x, data_y = data[i:end_ix, :-1], data[end_ix-1, -1]
        X.append(data_x)
        y.append(data_y)
    return array(X), array(y)
```

    - Función para construir, entrenar y probar el modelo LSTM, así como para mostrar resultados y métricas de calidad de las predicciones.

```
# ------------------------------------------------------------------------------
# Function that perfoms the process of building, training and testing a LTSM
# model with the number data, input variables, number of previous observations,
# number of neurons and number of epochs indicated as arguments
# ------------------------------------------------------------------------------
def generate_model(data, n_input, lags, n_neurons, n_epochs):
    n_obs = lags + 1

    # Normalize data
    MM_scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = MM_scaler.fit_transform(data)

    # Setup input and output data
    X, y = setup_input_output(scaled_data, lags)

    # Split into train and test data
    train_size = int(len(X) * 0.7)
    X_train, X_test = X[0:train_size,:], X[train_size:,:]
    y_train, y_test = y[0:train_size], y[train_size:]

    # Create and fit the LSTM model
    model = Sequential()
    model.add(LSTM(n_neurons, input_shape=(n_obs, n_input)))
    model.add(Dropout(0.2))  Droptout 20% of neurons randomly to prevent overfitting
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    history = model.fit(X_train, y_train, epochs=n_epochs, verbose=2,
                        validation_data=(X_test, y_test), shuffle=False)

    # Visualize Loss
    pylab.plot(history.history['loss'], label='train')
    pylab.plot(history.history['val_loss'], label='test')
    pylab.legend()
    pylab.show()

    # Make predictions
    y_test_predicted = model.predict(X_test)
    print('Predictions: ')
    print(y_test_predicted[:,0])
    print('Real data:')
    print(y_test)
    print(len(y_test))

    # Calculate RMSE, MAPE and R2
    rmse = math.sqrt(mean_squared_error(y_test, y_test_predicted[:,0]))
    mape = mean_absolute_percentage_error(y_test, y_test_predicted[:,0])
    r2 = r2_score(y_test, y_test_predicted[:,0])
    print('RMSE: %.2f' % (rmse))
    print('MAPE: %.2f' % (mape))
    print('R SQUARED: %.2f' % (r2))

    # Reconstruct data
    y_test_predicted_aux = y_test_predicted.reshape(len(y_test_predicted), 1)
    X_test_aux = X_test[:,-1:].reshape(len(X_test), n_input)
    data_test_predicted = concatenate((X_test_aux, y_test_predicted_aux), axis = 1)
    data_test = scaled_data[train_size+lags:,:]

    # Inverse normalization
    data_test_predicted_inv = MM_scaler.inverse_transform(data_test_predicted)
    data_test_inv = MM_scaler.inverse_transform(data_test)

    # Plot results
    dates = df_ini[['Date_Time']].to_numpy()
    dates = dates[train_size+lags:,0]
    pylab.plot(dates, data_test_inv[:,-1:][:,0],'b',label='Consumo total real')
    pylab.plot(dates, data_test_predicted_inv[:,-1:][:,0],'r',
               label='Consumo total predicho')
    pylab.ylabel('Consumo')
    pylab.xlabel('Fecha')
    pylab.legend()
    pylab.xticks(rotation=90)
    pylab.show()
```

- Modelo inicial de 5 neuronas y 5 épocas de entrenamiento con PC1 como variable descriptora y *Skyspark* como variable target.

```
# Data for group 1 of models
data_g1 = pca_df[['PC1','Skyspark']]
n_input_g1 = 1
```

```
# Model with data_g1, 5 lags, 5 neurons and 5 epochs
generate_model(data_g1, n_input_g1, 5, 5, 5)
```

- Primer grupo de modelos: PC1 como variable descriptora, *Skyspark* como variable target.

```
# Model with data_g1, 5 lags, 50 neurons and 25 epochs
generate_model(data_g1, n_input_g1, 5, 50, 25)
```

```
# Model with data_g1, 5 lags, 50 neurons and 40 epochs
generate_model(data_g1, n_input_g1, 5, 50, 40)
```

```
# Model with data_g1, 5 lags, 100 neurons and 40 epochs
generate_model(data_g1, n_input_g1, 5, 100, 40)
```

```
# Model with data_g1, 5 lags, 100 neurons and 100 epochs
generate_model(data_g1, n_input_g1, 5, 100, 100)
```

- Segundo grupo de modelos: PC1 y PC2 como variables descriptoras, *Skyspark* como variable target.

```
# Data for group 2 of models
data_g2 = pca_df[['PC1','PC2','Skyspark']]
n_input_g2 = 2
```

```
# Model with data_g2, 5 lags, 50 neurons and 40 epochs
generate_model(data_g2, n_input_g2, 5, 50, 40)
```

```
# Model with data_g2, 5 lags, 75 neurons and 50 epochs
generate_model(data_g2, n_input_g2, 5, 75, 50)
```

```
# Model with data_g2, 5 lags, 50 neurons and 100 epochs
generate_model(data_g2, n_input_g2, 5, 50, 100)
```

```
# Model with data_g2, 5 lags, 100 neurons and 100 epochs
generate_model(data_g2, n_input_g2, 5, 100, 100)
```

- Tercer grupo de modelos: Variables originales como variables descriptoras, *Skyspark* como variable target.

```
# Data for group 3 of models
data_g3 = df[['Copier','Lamp','Laptop','Monitor','Printer',
              'Projector','TV','Skyspark']]
n_input_g3 = 7
```

```
# Model with data_g3, 5 lags, 50 neurons and 25 epochs
generate_model(data_g3, n_input_g3, 5, 50, 25)
```

```
# Model with data_g3, 5 lags, 50 neurons and 40 epochs
generate_model(data_g3, n_input_g3, 5, 50, 40)
```

```
# Model with data_g3, 5 lags, 100 neurons and 40 epochs
generate_model(data_g3, n_input_g3, 5, 100, 40)
```

```
# Model with data_g3, 5 lags, 100 neurons and 100 epochs
generate_model(data_g3, n_input_g3, 5, 100, 100)
```