Universitat Oberta de Catalunya

uoc.edu

Predicción de consumos eléctricos

Carles Figuera Penedo
Máster universitario en Ciencias de Datos
TFM – Área 5

Nombre Consultor/a: Sergio Trilles Oliver
Profesor/a responsable de la asignatura: Albert Solé Ribalta

Junio de 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Predicción de consumos eléctricos*
Nombre del autor:	*Carles Figuera Penedo*
Nombre del consultor/a:	*Sergio Trilles Oliver*
Nombre del PRA:	*Albert Solé Ribalta*
Fecha de entrega (mm/aaaa):	06/2020
Titulación::	*Máster universitario en Ciencias de Datos*
Área del Trabajo Final:	*M2.982 - TFM - Área 5*
Idioma del trabajo:	*Castellano*
Palabras clave	*energy consumption prediction, time series forecasting, deep learning.*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

Hoy en día la energía eléctrica es un recurso esencial para la mayoría de la población. Dispositivos electrónicos, calefacción, electrodomésticos, iluminación, comunicaciones y a día de hoy muchos vehículos de transporte, son claros ejemplos de que la importancia del abastecimiento eléctrico para la población.

Las distribuidoras eléctricas deben dar servicio a la población de un recurso tan importante con dos importantes problemáticas: es un recurso difícil de almacenar (en gran volumen) y difícil de producir de forma inmediata. Así que se debe generar un volumen de energía en equilibrio con el volumen de demanda, de ahí viene la necesidad de predecir constantemente cuál será el consumo de la población.

Aunque existen ciertos patrones que nos ayudan a predecir el comportamiento consumista de la población, existen otros factores como la climatología o la economía que pueden modificar dichos patrones y provocar un error en la predicción de consumo, dejando en consecuencia a parte de la población sin abastecimiento eléctrico o produciendo más energía de la demandada.

Para predecir estas necesidades de consumo, existen distintos algoritmos y modelos de predicción que nos ayudan a solventar este problema. En este proyecto trabajaremos con un *dataset* de mediciones de consumo (*time series*) para investigar y comparar las predicciones obtenidas de los distintos modelos de *deep learning* estudiados en el proyecto que se ajusten a esta tipo de problemática o necesidad.

Abstract (in English, 250 words or less):

Nowadays the electric energy is an essential resource for most of the population. Electronic devices, heating, appliances, lighting, communications and today many transport vehicles, are clear examples of the importance of electricity supply for the population.

Electricity distributors must supply the population of such an important resource with two important problems: it’s a resource difficult to store (in large volumes) and difficult to produce immediately. So a volume of energy must be generated in equilibrium with the volume of demand, hence the need to constantly predict what the population's consumption will be.

Although there are certain patterns that help us predict the consumer behavior of the population, there are other factors such as the weather or the economy that can modify these patterns and cause an error in the prediction of consumption, leaving consequently part of the population without supply electric or producing more energy than the defendant.

To predict these consumption needs, there are different algorithms and prediction models that help us solve this problem. In this project we will work with a dataset of consumption measurements (time series) to investigate and compare the predictions obtained from the different deep learning models studied in the project that fit this type of problem.

Índice

1. Introducción

1.1 Contexto y justificación del Trabajo

La electricidad es un recurso muy importante para la población. Son muchas y muy variadas las necesidades que la población cubre con la energía eléctrica (calefacción, entretenimiento, iluminación, comunicación, transporte,...). Los avances en tecnología hacen que cada día haya más demanda de este recurso, y por ello, la disponibilidad de energía eléctrica es tan importante para hogares, empresas, hospitales, escuelas, etc. Dada su importancia, se podría decir que la disponibilidad de electricidad es un claro indicador de crecimiento y desarrollo económico para una población.

Pero cuando hablamos en términos de disponibilidad, la energía eléctrica muestra dos grandes problemas para las empresas distribuidoras.

En primer lugar, la electricidad es un recurso muy costoso de almacenar en grandes volúmenes. Otros recursos como el petróleo o el gas, se pueden almacenar en depósitos a la espera de ser usados o consumidos. Pero la electricidad tiene unas características que hacen que sea muy costoso de almacenar, aunque hay empresas como *Endesa* o *Iberdrola* que empiezan a trabajar en esta línea de investigación usando baterías de gran tamaño [1][2]. Pero se sabe que las baterías pierden eficiencia con el tiempo y uso, haciendo que por el momento, sea muy costoso el almacenaje de energía. Por eso decimos que el consumo de energía eléctrica es un consumo en tiempo real, ya que se consume a la vez que se genera, sin posibilidad de almacenar sobrantes. Así que la generación y su consumo deben estar en equilibrio.

En segundo lugar, la producción de energía eléctrica no es inmediata. Es decir, que si se dispara el consumo eléctrico en un intervalo de tiempo corto, las distribuidoras tienen muy poco tiempo de reacción para producir más energía. Aunque la respuesta de las centrales eléctricas tiene que ser rápida para no desabastecer parte de la población, muchas veces, solo el hecho de incrementar la producción eléctrica puede costar entre minutos y horas, tiempo necesario para por ejemplo, abrir compuertas en una central hidroeléctrica o incrementar la cantidad de combustible de las centrales térmicas. Además hay que tener en cuenta que normalmente las fuentes de energía eléctrica con capacidad de respuesta más rápida acostumbran a ser las más contaminantes (centrales termoeléctricas de carbón, gas o gasóleo). La eficiencia de las centrales renovables dependen de recursos naturales como el viento, el caudal hidrológico o la luz solar, de las cuáles no tenemos control.

En resumen, el coste elevado de almacenar la energía y el coste económico/medioambiental de producirla de forma rápida, hacen que sea de gran valor y un tema relevante la posibilidad de realizar predicciones de consumo eléctrico de los usuarios de la red eléctrica. A continuación podemos observar un ejemplo de la predicción en tiempo real [3] de la que dispone la REE (Red Eléctrica Española).

Podemos observar como en un mismo día, la demanda de energía eléctrica (línea amarilla) es muy irregular. Desciende por las noches cuando la población descansa y presenta dos picos acentuados a primera hora de la mañana (cuando la población y las empresas empiezan su actividad diaria) y al atardecer (cuando la población vuelve a sus hogares).

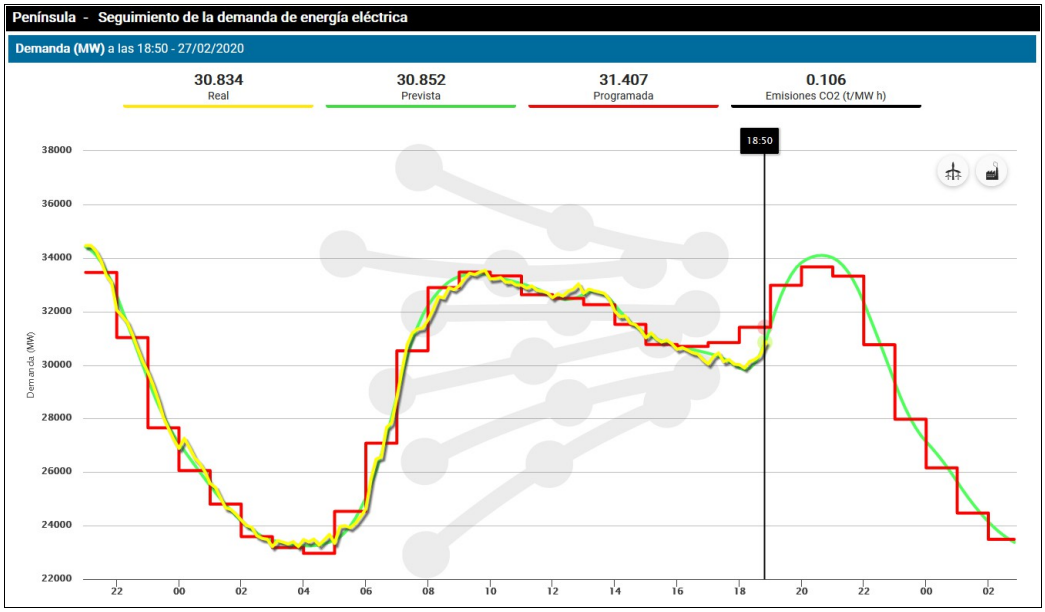


Figura 1.1. Demanda y predicción de energía eléctrica por la REE

Esta gráfica muestra cómo las predicciones de consumo pueden variar según las horas del día, aunque a la vez también nos muestra que existen ciertos patrones. Pero no solo pueden variar según la hora, sino que hay otras características o factores a tener en cuenta cuando hablamos de predicción.

A continuación se muestra un listado de todas ellas:

1. Horas del día: como ya hemos comentado, el consumo es diferente según horas del día (figura 1.1).

2. Días de la semana: los consumos eléctricos son distintos en días laborables o en fin de semana. El fin de semana acostumbra a bajar el consumo ya que muchas empresas cesan su actividad.

3. Estaciones del año: por ejemplo, el uso de calefacción en invierno y aires acondicionados en verano hace que se tenga en cuenta la época del año para predecir de forma más exacta el consumo eléctrico.

4. Eventos puntuales: una huelga general provoca la bajada del consumo de forma notoria. O bien la noche de fin de año puede que hacer que se incremente la demanda ya que los locales de ocio trabajan más horas.

5. Meteorología: existe una correlación entre el consumo de energía eléctrica y la temperatura ambiente. Así que también deberá existir una relación entre la predicción de consumo eléctrico y la predicción meteorológica.

6. Economía: la salud financiera de una zona geográfica también aplica al consumo eléctrico. Por ejemplo, durante la última crisis económica, el cierre de muchas fábricas y empresas hizo disminuir el consumo eléctrico de forma notoria.

Así que podemos hablar de dos grandes tipos de factores que afectan al consumo de electricidad: el calendario y la climatología. Estos factores tienen patrones conocidos. Por una parte, la temperatura y las lluvias siguen un patrón anual conocido (figura 1.2), y por otra parte, el factor temporal o de calendario (días festivos, días laborables, horas diurnas, horas nocturnas,...) también sigue otro patrón conocido (figura 1.3). Así que podemos trabajar las predicciones de consumos eléctricos como si fueran predicciones de series temporales.

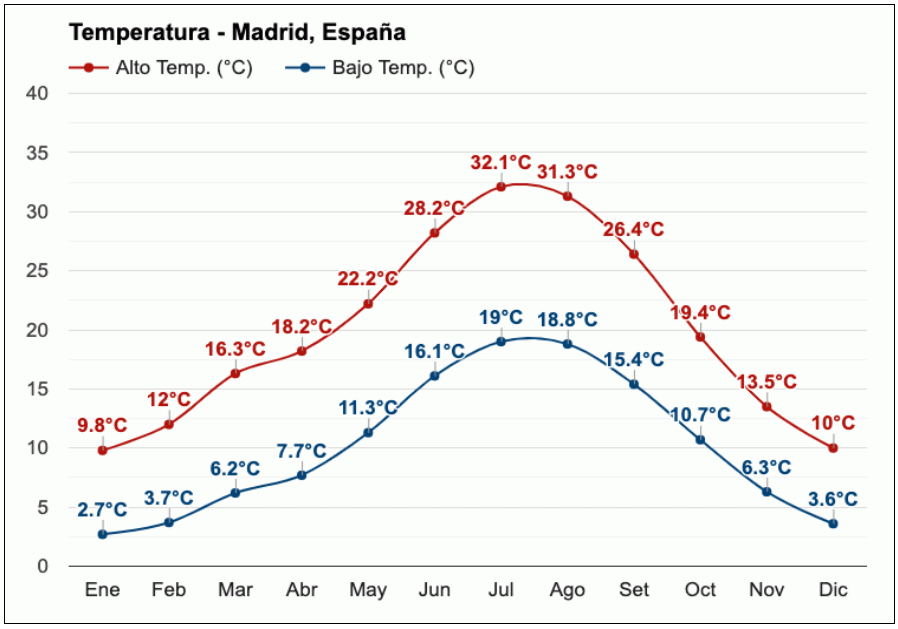


Figura 1.2. Promedio anual de temperaturas en Madrid **[4]**

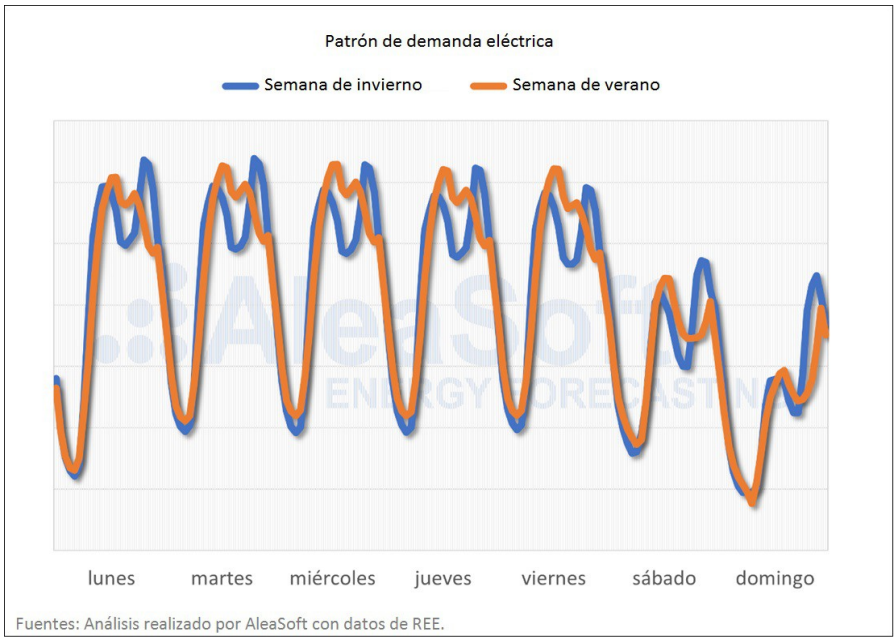


Figura 1.3. Demanda eléctrica media en España 2014-2018 según semana y estación del año

En este trabajo estudiaremos y compararemos diferentes modelos de predicción que encajan para resolver problemas del tipo series temporales como el introducido en este apartado.

1.2 Motivación personal

Creo que de todas las disciplinas del *Big Data*, las de *machine learning* y *deep learning* es la más interesante. La idea de predecir, averiguar qué va a suceder antes de que suceda de forma automática, solamente teniendo en cuenta datos de lo que ha pasado anteriormente me parece muy interesante.

Además, al licenciarme en 2011, mi primer proyecto profesional fue en un proyecto de *Smartcity* en Málaga [5], donde comprendí las dificultades que tenía administrar la energía eléctrica de toda una zona de la ciudad. El proyecto fue construir una prueba piloto de introducción de sistemas *Big Data* en la ciudadanía, creando lo que llamaban una 'ciudad inteligente'. Fue una gran experiencia, en mi departamento se trabajaba mucho en la monitorización del uso de la energía eléctrica para el ahorro energético.

Así que este trabajo trata sobre dos temas que me fascinan y me aportará mucho a nivel personal.

1.3 Objetivos del Trabajo

A continuación se muestra un listado de objetivos que se quieren realizar en este trabajo de fin de máster:

- Entender la necesidad y problemática de la predicción de consumos eléctricos.
- Estudiar el estado del arte actual de algoritmos estadísticos o de aprendizaje automático (*machine learning*) y profundo (*deep learning*) para predicción de series temporales (*time series forecasting*).
- Tomar un *dataset* de series temporales sobre consumos eléctricos, analizarlo y prepararlo para la creación de modelos predictivos.
- Crear y entrenar distintos modelos de predicción a partir del *dataset* mencionado.
- Preprocesar el *dataset* anterior para poder entrenar y evaluar de forma satisfactoria los distintos modelos predictivos creados.
- Analizar y comparar los distintos resultados obtenidos con los modelos anteriores mediante un *dataset* de evaluación.
- Evaluar la posibilidad de crear modelos híbridos o mixtos.
- Estudiar posibles líneas futuras de estudio.

1.4 Enfoque y método seguido

La estrategia a seguir en este proyecto será la de investigar y comparar los modelos predictivos usados en la actualidad, productos ya existentes. Definiremos su funcionamiento, crearemos modelos de test, los entrenaremos y los compararemos tanto en resultados como en rendimiento. También intentaremos encontrar solución a los problemas o errores encontrados en dichos modelos.

La metodología a seguir, por las características de la investigación y el número de participantes, será la de un proceso de desarrollo secuencia, es decir, desarrollo en cascada. Puede ser que en la fase de creación de modelos predictivos se pase a una metodología más *agile* del tipo *scrum* si pensamos en cada modelo predictivo a reproducir en el estudio como en un entregable.

Dichas investigaciones o experimentos serán a futuro reproducibles, ya que se almacenaran, además de los productos resultantes de la investigación (resultados, figuras, tablas, memoria, etc), los códigos fuente, los *datasets* o datos originales y los ya preprocesados, junto con las instrucciones a seguir para su repetición futura del estudio en caso de ser necesario.

Todo producto resultante se entregará en el siguiente repositorio GitHub:

https://github.com/cfiguerap/uoc-data-science-tfm

1.5 Planificación del Trabajo

A continuación, se muestra un diagrama de *Gantt* de la planificación temporal inicial que se ha establecido para el proyecto:

Implementación del trabajo - Mode...	**23/03/20**	**25/05/20**
Análisis del dataset del caso de estu...	23/03	27/03
Preprocesado del dataset del estudio	30/03	03/04
Construcción y entrenamiento de los...	06/04	24/04
Resultados obtenidos de los modelo...	27/04	06/05
Conclusiones del estudio realizado	07/05	13/05
Futuras líneas de investigación	14/05	19/05
Publicación investigación en reposit...	20/05	22/05
Fecha límite PAC3	25/05	25/05
Documentación	**25/05/20**	**24/06/20**
Redacción memoria final	25/05	09/06
Fecha límite PAC4	10/06	10/06
Presentación y defensa proyecto	08/06	15/06
Fecha límite PAC5	16/06	16/06
Defensa pública	24/06	24/06

Figura 1.4. Diagrama de *Gantt* con planificación temporal del proyecto

1.6 Breve sumario de productos obtenidos

El producto entregable del presente proyecto es una comparativa de rendimiento y resultados de los modelos predictivos más comunes de predicción de series temporales basados en un *dataset* de consumo eléctrico.

Además, se hará entrega de la presenta memoria y resto de documentación, junto los ejecutables producto del estudio realizado, en el siguiente repositorio *GitHub*: https://github.com/cfiguerap/uoc-data-science-tfm

1.7 Breve descripción de los otros capítulos de la memoria

El siguiente esquema muestra la estructura del presente documento, dividiéndose en los siguientes capítulos:

- Capítulo 2. Estudio del estado del arte: introducción a modelos predictivos usados para la predicción de consumos u otros tipos de series temporales.

- Capítulo 3. Análisis y preprocesado del *dataset* del caso de estudio: análisis exhaustivo del *dataset* escogido para el caso de estudio y preprocesado, en caso de ser necesario, para la construcción de modelos predictivos.

- Capítulo 4. Creación y entrenamiento de los modelos predictivos: creación de diferentes modelos predictivos a partir de los datos del *dataset* antes mencionado. Resultados obtenidos tras la evaluación de dichos modelos predictivos.

- Capítulo 5. Conclusiones del estudio realizado: comparativa de los diferentes resultados obtenidos en el capítulo anterior.

2. Estado del arte

2.1 Series temporales

Una serie temporal es una secuencia de valores discretos ordenados en el tiempo. Normalmente, en los sectores industriales o de ingeniería, son el resultado de una monitorización de un proceso con cierto grado de interés. Por ejemplo, en el caso de estudio de este proyecto, el consumos eléctrico que estudiaremos es una serie temporal resultante de la continua monitorización de las distribuidoras eléctricas.

$$X = \{X_1, X_2, \ldots\} \text{ o } \{X_k\}_{k \geq 1}$$

Figura 2.1. Notación de una serie temporal

Las series temporales pueden ser regulares o irregulares. Las regulares tienen una distancia temporal constante entre sus valores (diarios, semanales,...), mientras que las irregulares tienen una distancia variable

Las series temporales son muy importantes para la predicción, para poder predecir correctamente los consumos futuros, necesitamos conocer los consumos del pasado. Pero para poder predecir con menor error, no solo necesitamos los valores de interés (el consumo), también necesitamos conocer otros factores relacionados con el consumo (meteorológicos, temporales, económicos, etc).

Las series temporales tienen una serie de características o propiedades [6]. Cada serie puede mostrar alguna o más de una de estas propiedades:

2.1.1 Estacionariedad

Esta propiedad está relacionada con la media y varianza de los valores de la serie temporal. Se dice que es estacionaria cuando la media y la varianza se mantienen constantes con el tiempo y además no presentan una tendencia.

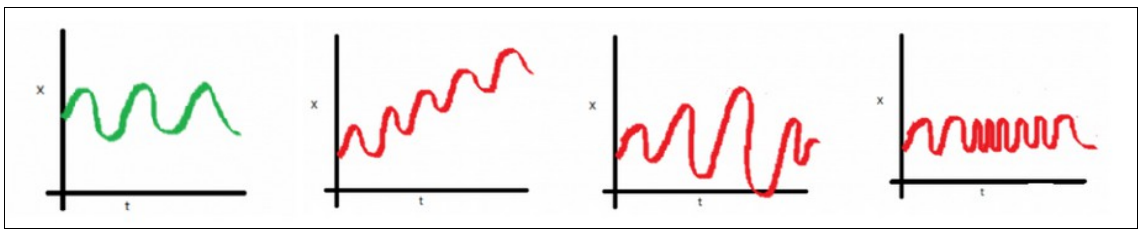


Figura 2.2. En verde, ejemplo de una serie temporal estacionaria
y en rojo distintos ejemplos de series no estacionarias **[7]**

2.1.2 Linealidad

La linealidad de una serie temporal indica que la forma que tiene depende del tiempo, es decir, los valores de la serie se pueden determinar mediante el tiempo con una función lineal. Si la serie es lineal, entonces puede ser representada por una función o ecuación lineal. En cambio, las no lineales solo pueden ser representadas mediante ecuaciones no lineales.

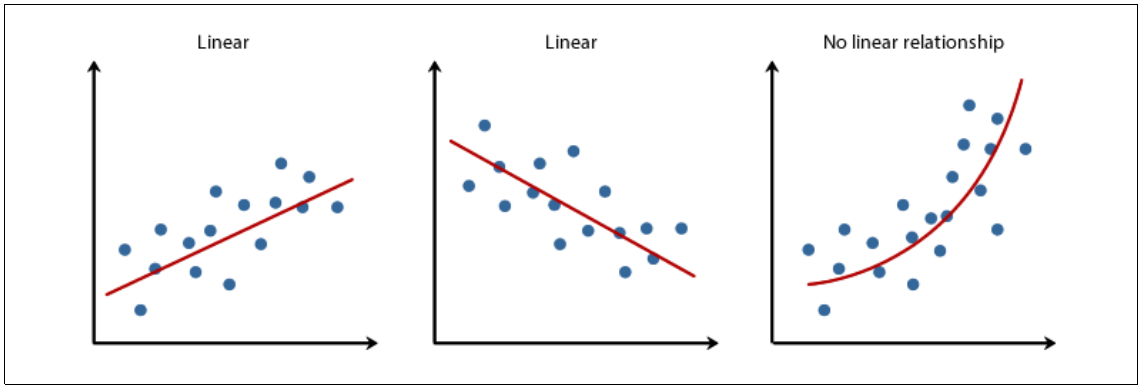


Figura 2.3. Ejemplos de series temporales lineales y no lineales

2.1.3 Tendencia

El componente de tendencia hace referencia al movimiento general de largo plazo de la serie temporal. Es decir, la tendencia describe el comportamiento de la serie a largo plazo. El análisis de tendencias es importante en el pronóstico de series de tiempo. En la práctica, se usan técnicas de regresión lineal y no lineal que ayudan a identificar el componente de tendencia no monótono en la serie temporal.

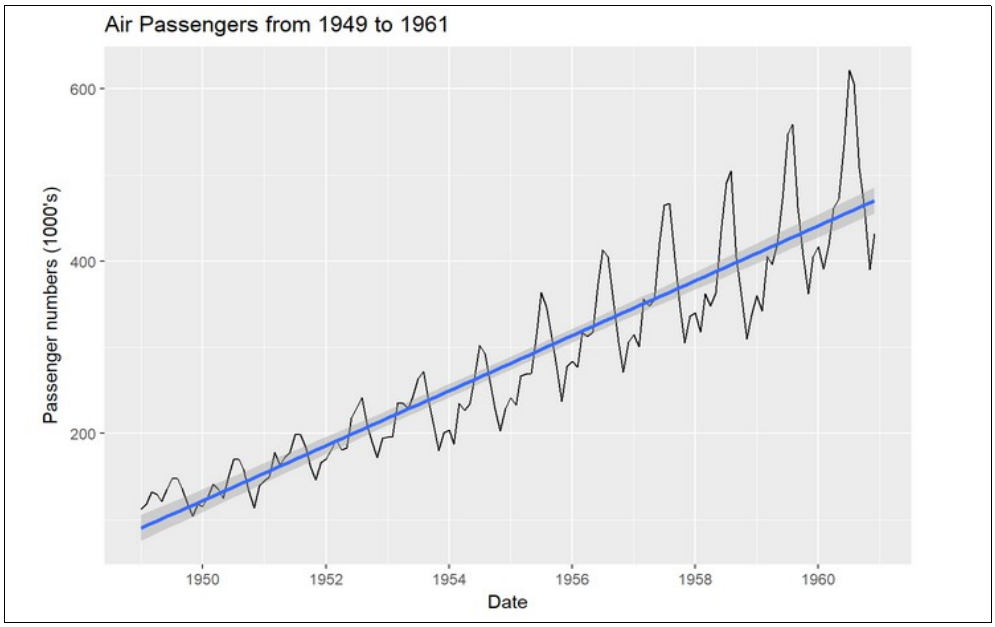


Figura 2.4. Ejemplo de serie temporal con tendencia incremental

2.1.4 Estacionalidad

Los componentes estacionales se pueden definir como oscilaciones que se producen en períodos iguales o inferiores a un año y que se reproducen de manera reconocible en los diferentes años. En otras palabras, hace referencia a cierta periodicidad de la serie o variación de cierto período, ya sea anual, semestral, trimestral, mensual, etc.

Normalmente, el objetivo principal del análisis de series temporales estacionales se centra en la detección del carácter de su fluctuaciones periódicas y sobre su interpretación. Los ejemplos típicos de series temporales estacionales, como en nuestro caso de estudio, son distribuciones de energía, gas, agua u otros sistemas, donde la predicción de las demandas de los consumidores representa el problema básico.

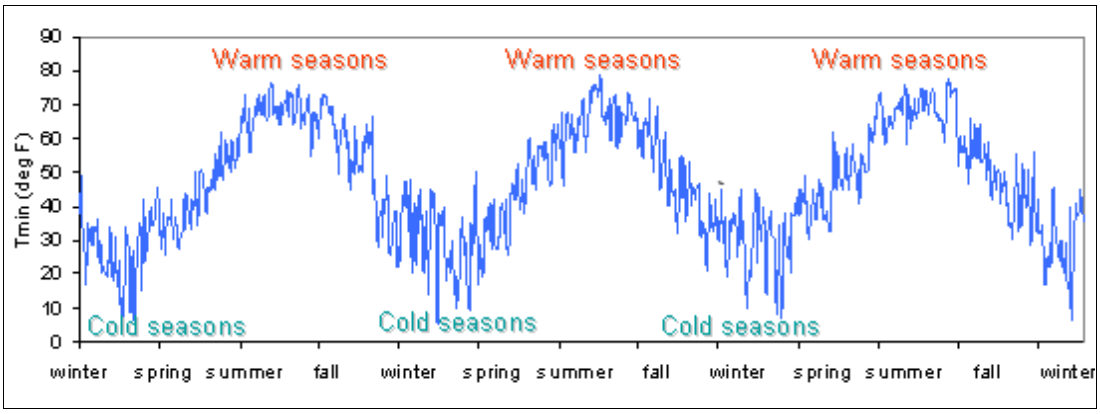


Figura 2.5. Serie temporal estacional, podemos apreciar como el período oscila de forma anual.

2.2 Componentes de una serie temporal

Las series temporales se pueden descomponer en cuatro componentes diferenciados que son: tendencia, estacionalidad, ciclo y ruido [8]. Cuando hablamos de análisis de series temporales, dichos componentes, en combinación los unos con los otros, definen el comportamiento de la serie temporal. A continuación definimos los componentes [8]:

- Tendencia: movimiento general que describe el comportamiento de la serie temporal a largo plazo.

- Ciclo: fluctuaciones alrededor de la tendencia que se repiten de forma más o menos periódica y de amplitud superior al año.

- Estacionalidad: oscilaciones que se producen en períodos iguales o inferiores a un año (diaria, semanal, mensual, trimestral, etc).

- Ruido: oscilaciones que no responden a ningún patrón de comportamiento conocido. Suelen ser debidos a factores fortuitos.

Las series temporales pueden llegar a representar una variedad de patrones infinitos. Y si se descompone la serie en estos distintos componentes que la forman, podemos llegar a descubrir distintos patrones en los componentes. Este hecho, nos permite poder conocer el comportamiento de las series temporales de una forma más clara, ya que estudiando cada componente por separado nos aporta información más útil y más fácil de conseguir que estudiando directamente la serie temporal original.

Por ejemplo, cuando analizamos el volumen de pedidos de una compañía para predecir sus futuras ventas, podemos analizar por separado cada componente de la serie temporal de ventas anteriores. En este caso, la tendencia nos ofrecería información valiosa sobre el ritmo de ventas sin vernos afectados por las fluctuaciones derivadas de los meses de vacaciones o Navidad. Mientras que la estacionalidad nos daría una idea de cuáles son los mejores y peores meses del año de cara a las ventas de la compañía.

En la práctica, la distinción entre tendencia y ciclo es problemática ya que, cuando se tienen pocas observaciones, es fácil confundir estos dos componentes. Así que cuando descomponemos una serie temporal en sus componentes, normalmente se combina la tendencia y el ciclo en un solo componente. De esta forma, podemos pensar que una serie temporal consta de solo tres componentes: un componente de tendencia, un componente estacional y un componente de ruido [10].

En la siguiente figura (2.6) podemos observar los tres distintos componentes que forman una serie temporal:

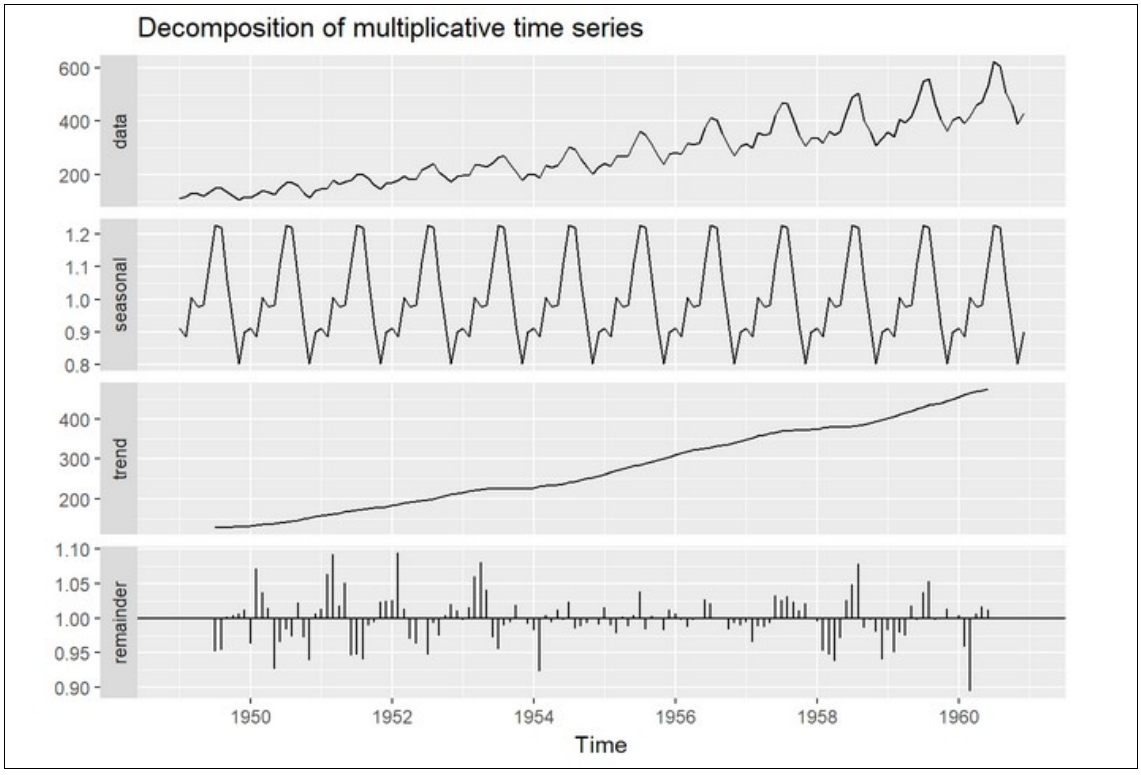


Figura 2.6. Ejemplo de descomposición de serie temporal.

2.3 Clasificación de una serie temporal

Como hemos visto en los apartados anteriores, dependiendo de los valores que forman las series temporales, las podemos clasificar en:

- Estacionarias o no estacionarias
- Estacionales o no estacionales
- Lineales o no lineales
- Caóticas (valores aleatorios sin periodicidad)

Pero además existen otros dos niveles más de clasificación [6]:

- Univariables o Multivariables:

  Los valores de las series univariables solo dependen del tiempo (unidimensional). Por ejemplo, datos recopilados de un sensor que mide temperatura.

  Las series multivariables dependen del tiempo y otros factores (multidimensional). Por ejemplo, en nuestro caso de estudio, el consumo eléctrico depende del factor temporal pero también de la meteorología.

  Las multivariadas, al depender de más factores, tienden a tener por probabilidad más *outliers*. Además, normalmente estos distintos factores pueden presentar una correlación que da forma al patrón de la serie temporal.

- Aditivas o Multiplicativas:

  Las series aditivas son las formadas por la suma de sus componentes (tendencia, estacionalidad y ruido).

$$x_t = T_t + E_t + C_t + R_t$$

Figura 2.7. Ecuación de una serie temporal aditiva (tendencia, estacionalidad, ciclo y residuo)

  En cambio, las multiplicativas son las formadas por el producto de sus componentes.

$$x_t = T_t \cdot E_t \cdot C_t \cdot R_t$$

Figura 2.8. Ecuación de una serie temporal multiplicativa (tendencia, estacionalidad, ciclo y residuo)

2.4 Predicción de series temporales

La predicción de series temporales es un tema que hoy en día interesa mucho en distintos sectores: financiero, comercial, energético, sanitario,... Desde predicciones bursátiles (IBEX35) hasta predicciones de contagios virales (COVID19), pasando por la predicción de demanda energética como sucede en nuestro caso de estudio.

El propósito de la predicción de series temporales suele ser doble: comprender las causas que dan lugar a una serie temporal (análisis) y predecir sus valores futuros a partir de sus valores pasados o conocidos (predicción).

Existen distintas técnicas de pronóstico o predicción de series temporales, algunas de ellas más modernas que otras. Existen desde técnicas puramente estadísticas (AR, MA o ARIMA) hasta técnicas más modernas de *Deep Learning* (ANN o SVM). Pero también existen casos de modelos o técnicas híbridas donde se usan más de un tipo de técnicas conjuntamente.

A continuación se introducen estos dos grupos importantes mencionados:

1) Técnicas tradicionales

Entre las distintas técnicas tradicionales destacan los modelos estadísticos, como el modelo autoregresivo (*AR*) o promedio móvil (*MA*), y en particular el modelo *ARIMA* (combinación de los dos modelos anteriores) que ha demostrado un mejor rendimiento en precisión y exactitud entre las predicciones de estas técnicas tradicionales.

Pero este tipo de modelos presentan una serie de desventajas o limitaciones a la hora de modelar problemas reales [11]:

- Requieren datos completos: valores perdidos u *outliers* pueden afectar considerablemente a la predicción resultante.

- Se basan en relaciones lineales: sus suposiciones tienen una base lineal que empeoran la predicción con series temporales complejas no lineales.

- Se basan en series univariables: no se obtiene una buena predicción cuando los valores a predecir dependen de más de un *input* o entrada.

- Pronostican a corto tiempo: por lo general, funcionan bien en predicciones a corto plazo pero no a largo.

2) Técnicas de aprendizaje automático

Con el avance en los últimos años en tecnología y desarrollo de algoritmos de aprendizaje automático, se han desarrollado nuevos modelos que parece que obtienen mejores predicciones que los modelos tradicionales. Entre ellos destacan dos modelos de aprendizaje supervisado con redes neuronales: las *LSTM (long short-term memory)* y CNN *(convolutional neural network).*

Este tipo de técnicas tienen una serie de ventajas frente a los modelos tradicionales:

- Capacidad para aproximar funciones no lineales.
- Capacidad para tratar el ruido y *outliers*.
- Capacidad para aceptar entradas multivariables.
- Capacidad para realizar pronósticos de varios pasos.

En los siguientes apartados vamos a entrar más en detalle de los modelos más usados en la actualidad para la predicción de series temporales. Se pueden usar muchos modelos distintos pero nos centraremos en los más usados.

2.5 Modelo ARIMA

El modelo ARIMA (*Autoregressive Integrated Moving Average*) es un método que se utiliza tanto para el análisis como la predicción de series temporales. Es un tipo de modelo estadístico que nos da una ecuación sobre una serie temporal en función de sus propios valores pasados, de modo que dicha ecuación se puede utilizar para pronosticar valores futuros. Como ya hemos dicho antes, es una combinación de los modelos autoregresivos (*AR*), promedio móvil (*MA*). Sus siglas se forman por los siguientes componentes:

- AR (Autoregresión): Un modelo que usa la relación dependiente entre una observación y un número de observaciones pasadas.

- I (Integrado): El uso de diferenciación entre los valores para ayudar a hacer la serie estacionaria. La diferenciación es un proceso que nos permite eliminar la tendencia de una serie temporal restando a cada valor de la serie el valor anterior.

- MA (Promedio móvil): Un modelo que usa la dependencia entre un valor y un error residual de un modelo de promedio móvil aplicado a valores anteriores.

Cada una de estos componentes forma parte en el modelo como un parámetro. Se utiliza una notación estándar de ARIMA *(p, d, q)* donde los parámetros se sustituyen por valores enteros para indicar un modelo ARIMA específico.

Los parámetros del modelo ARIMA se definen de la siguiente manera:

P: Número de observaciones pasadas (llamado orden de retraso).
D: Número de veces que los valores son diferenciadas (grado de diferencia).
Q: Tamaño de la ventana de promedio móvil (orden de promedio móvil).

Se construye un modelo de regresión lineal que incluye el número y tipo de términos especificados, y los datos se preparan mediante un grado de diferenciación para hacerlo estacionario, es decir, para eliminar las estructuras estacionales y de tendencia que afectan negativamente a la precisión del modelo.

Se puede usar un valor de 0 en un parámetro para indicar que no se debe usar ese componente del modelo. De esta forma, el modelo ARIMA se puede configurar para realizar la función de un modelo ARMA, e incluso un modelo AR, I o MA simple.

2.6 Modelos ANN

Las ANN (a*rtificial neural network)* [13-14] *son modelos* inspirados en las redes neuronales biológicas del cerebro humano. Están constituidas por elementos que se comportan de forma similar a una neurona biológica en sus funciones mas comunes. Las redes neuronales, como profundizaremos más adelante, pueden formarse por una o más capas de neuronas.

Al margen de "parecerse" al cerebro presentan una serie de características propias del cerebro. Por ejemplo las *ANN* aprenden de la experiencia, generalizan de ejemplos previos y abstraen las características principales de una serie de datos.

Las redes neuronales son capaces de:

- Aprender: *a*dquirir el conocimiento por medio de la experiencia. Se les muestra un conjunto de entradas y ellas mismas se ajustan para producir unas salidas consistentes.
- Generalizar: extender o ampliar un conocimiento. Generalizan automáticamente debido a su propia estructura y naturaleza. De este modo pueden mitigar el efecto que produce en los datos los *outliers* o el ruido.
- Abstraer: son capaces de abstraer la esencia de un conjunto de entradas que aparentemente no presentan aspectos comunes o relativos.

Cada neurona tiene un conjunto de entradas y cada una de estas entradas está ponderada por una serie de pesos. Cada peso se entiende como la importancia del valor de entrada que llega a la neurona.

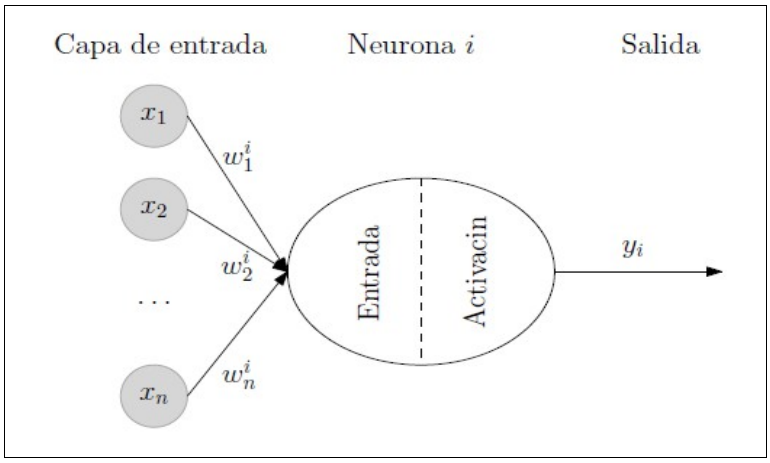


Figura 2.9. Nodo de una red neural

Cada neurona combina los valores de entrada, aplicando sobre ellos una función de entrada. La función de entrada puede ser una función de máximos, mínimos,... aunque la suma ponderada suele ser la función más usada.

$$z(x) = \sum_{j=1}^{n} x_j w_j^i$$

Figura 2.10. Función de entrada más usada en una red neuronal (suma ponderada)

El valor resultante es procesado por una función de activación, que modula el valor de las entradas para generar el valor de salida de la neurona. La función de activación suelen ser la escalón, lineal, sigmoide o hiperbólica. Destacar que las funciones sigmoide e hiperbólica presentan un comportamiento no lineal.

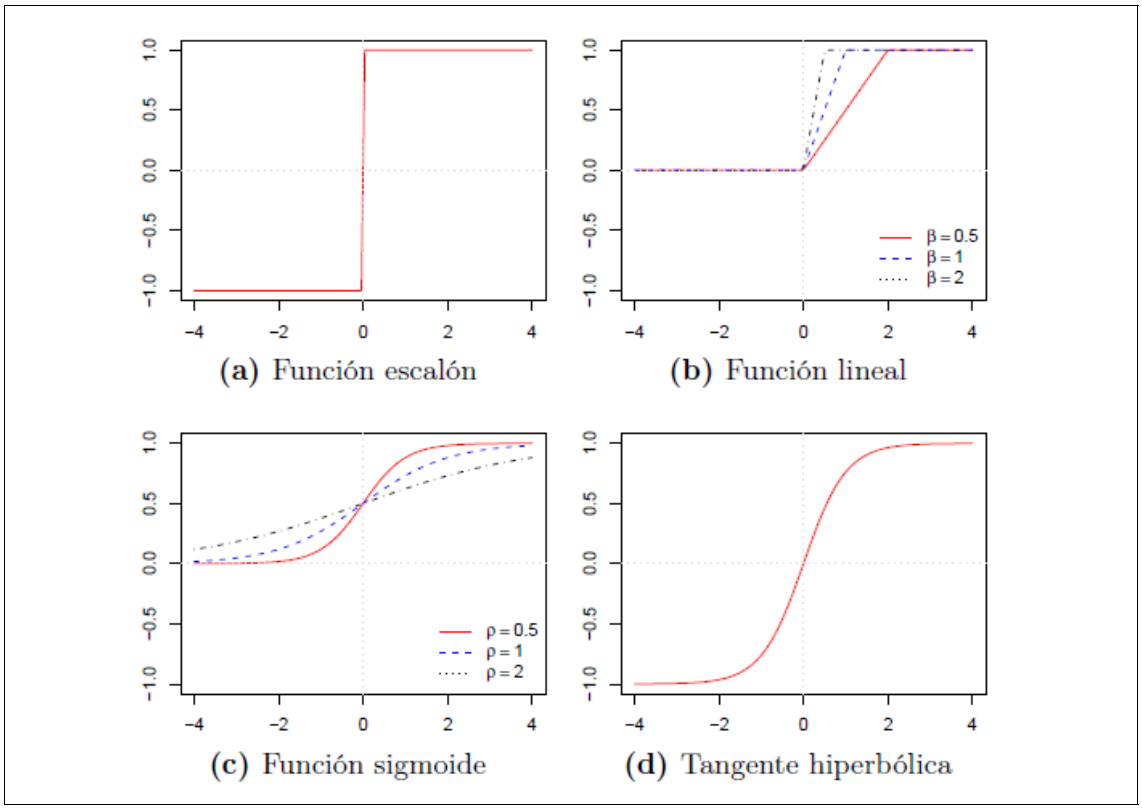


Figura 2.11. Funciones de activación: escalón (a), lineal (b), sigmoide (c) e hiperbólica (d).

El valor de salida de la función de activación, generalmente se propaga a la siguiente capa de conexiones neuronales o puede ser el valor definitivo resultante de la red neuronal.

Como ya hemos dicho antes, las redes neuronales pueden formarse por más de una capa. Cada capa puede tener su propia configuración en cuanto a pesos, función de entrada y función de activación. Podemos encontrar redes con más de una capa o más de una salida. Cuantas más capas mejor predicción obtendremos pero necesitaremos también más coste computacional.

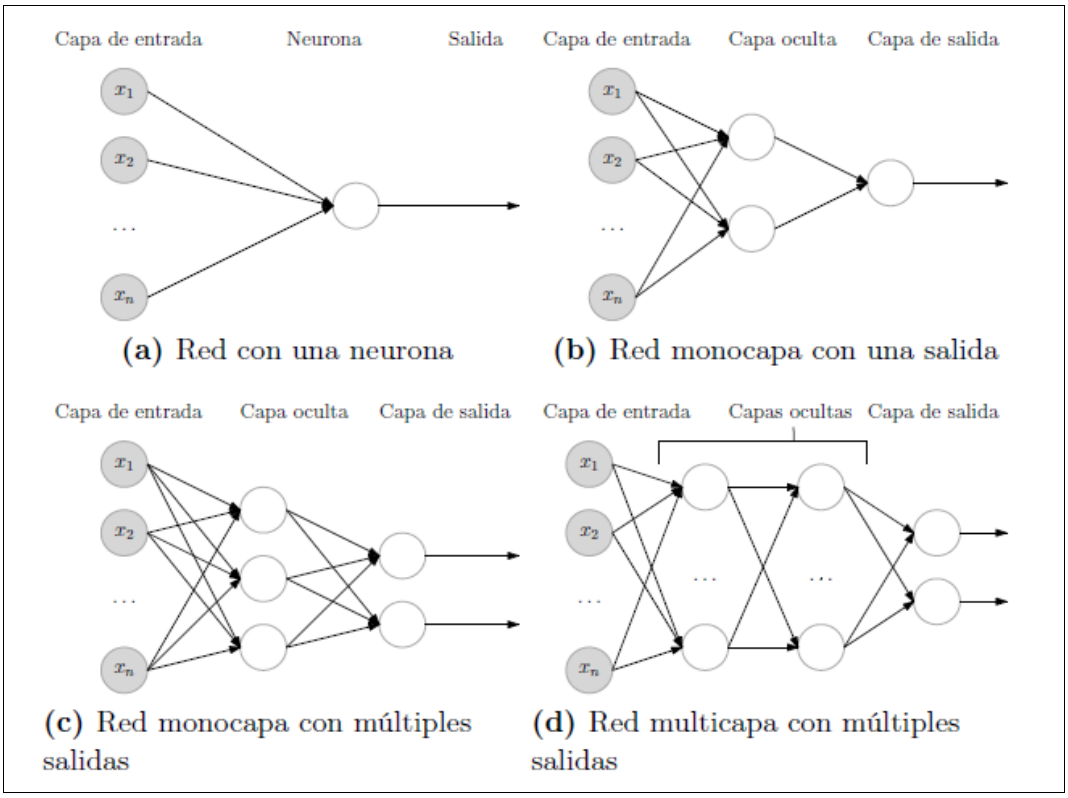


Figura 2.12. Diferentes arquitecturas de redes neuronales FNN.

Estos ejemplos de la figura anterior (2.12) muestran distintas arquitecturas de redes neuronales del tipo FNN (*feedforward neural network*). Este tipo de redes no producen bucles, la información siempre avanza y no se retroalimenta. De aquí destaca la CNN (*convolutional neural network*) usada sobretodo para clasificación de imágenes.

Pero hay otro tipo de arquitecturas con bucles retroalimentados, donde salidas de capas neuronales se reutilizan como entrada del sistema. Son las llamadas RNN (*recurrent neural network*) y son capaces de tener conocimiento de los datos en base al tiempo y el orden. Entre ellas destaca la LSTM (*long short-term memory*), usadas en reconocimiento de texto y predicción de series temporales, como es en nuestro caso de estudio.

Pero no todo son bondades en este tipo de modelos automáticos, cuando entrenamos nuestros modelos intentamos obtener una salida conocida a una entrada de valores conocidos. Así que estos modelos pueden tender a sobreentrarse, aprender los casos particulares que han adquirido en el entrenamiento y ser incapaces de reconocer nuevos datos de entrada. Este problema se llama también *overfitting*.

2.6.1 Red neuronal CNN

Las redes neuronales del tipo CNN (*convolution neural network*) son una extensión de las FNN (*feedforward neural network*), muy eficaces para procesar datos visuales y otros datos bidimensionales, son muy similares a las redes neuronales ordinarias multicapa, lo que hace que tengan un entrenamiento más corto al no ser retroalimentadas. Cada uno de los nodos de una capa no están conectados con las anteriores, sino que cada capa funciona como un filtro. Los filtros trabajan como una ventana que se desliza por la imagen en busca de patrones.

Las redes neuronales CNN se forman de las siguientes capas o etapas [15]:

- Capa de convolución: un filtro o ventana llamada *kernel* que se desliza sobre la imagen, visualizando unos pocos *píxeles* a la vez (por ejemplo, 3x3 o 5x5). La operación de convolución es un producto de los valores de *píxeles* originales con pesos definidos en el filtro. Los resultados se resumen en un número que representa todos los *píxeles* que observó el filtro.

- Capa de activación: la capa de convolución genera una matriz que es mucho más pequeña en tamaño que la imagen original. Esta matriz se ejecuta a través de una capa de activación, normalmente con una función de activación ReLu.

- Capa de *Pooling*: es el proceso de disminución de muestras y reducción del tamaño de la matriz. Se pasa un filtro sobre los resultados de la capa anterior y selecciona un número de cada grupo de valores. Esto permite que la red entrene mucho más rápido, enfocándose en la información más importante en cada característica de la imagen.

- Capa *Fully connected*: una estructura multicapa tradicional. Su entrada es un vector unidimensional que representa la salida de las capas anteriores. Su salida es una lista de probabilidades para diferentes etiquetas posibles adjuntas a la imagen. La etiqueta que recibe la mayor probabilidad es la decisión de clasificación.

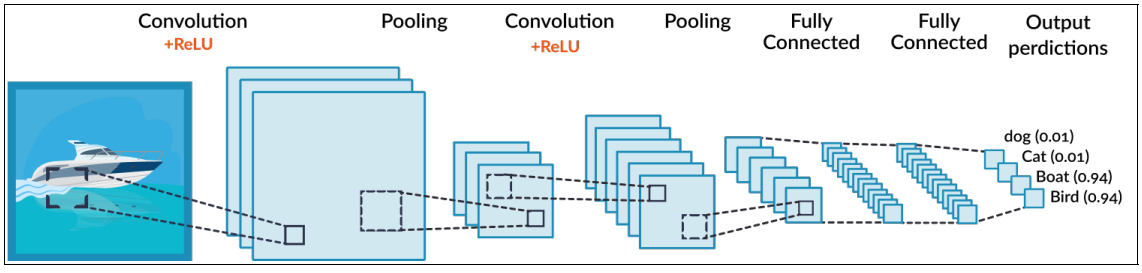


Figura 2.13. Capas de una red neuronal del tipo convolucional (CNN)

Aunque su función más conocida es la de clasificar imágenes, también son útiles para la predicción de series temporales. Las imágenes no son más que matrices de *píxeles*, así que si la red neuronal CNN es capaz de trabajar con imágenes, si transformamos la matriz 2D a un vector 1D, la red también será capaz de trabajar con un vector que represente una serie temporal [16]. Así que los filtros que trabajan como ventanas deslizantes en busca de patrones en una imagen, también pueden buscar patrones en nuestra serie temporal.

2.6.2 Red neuronal LSTM

Las redes neuronales del tipo LSTM (l*ong-short term memory*) son una extensión de las RNN (*recurrent neural network*), que básicamente amplían su memoria para aprender de experiencias importantes que han pasado a lo largo del tiempo. Las LSTM permiten a las RNN recordar sus entradas durante un período de tiempo. Esto se debe a que LSTM contiene información en una memoria.

Esta memoria se puede ver como una “celda” bloqueada, donde “bloqueada” significa que la neurona decide si almacenar o eliminar información dentro, en función de la importancia que asigna a la información que está recibiendo. La asignación de importancia se decide a través de los pesos, que también se aprenden mediante el algoritmo. Esto lo podemos ver como que aprende con el tiempo qué información es importante y cuál no.

Las celdas LSTM tiene tres puertas o *gates* que dan acceso a esta información: puerta de entrada (*input gate*), puerta de olvido (*forget gate*) y puerta de salida (*output gate*). Estas puertas determinan si se permite o no una nueva entrada, se elimina la información porque ya no es importante o se deja que afecte a la salida en el paso de tiempo actual.

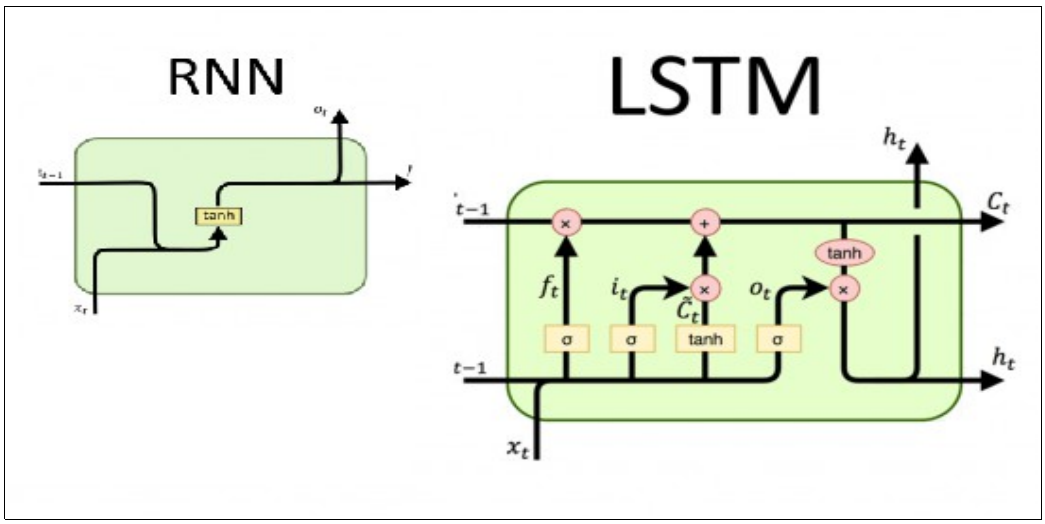


Figura 2.14. Esquema de una celda RNN y una LSTM

Las puertas en una LSTM son análogas a una forma *sigmoide*, lo que significa que van de 0 a 1. El hecho de que sean análogas a una función de activación *sigmoide*, permite incorporarlas al proceso de retroalimentación.

Existen otras implementaciones RNN parecidas como las GRU (g*ated recurrent unit).* Las capas GRU aparecieron en el 2014 [17] y usan el mismo principio que LSTM, pero están simplificadas de manera que tienen menos puertas o *gates*, su rendimiento es parecido al de las LSTM pero son más eficientes en cuanto a coste computacional.

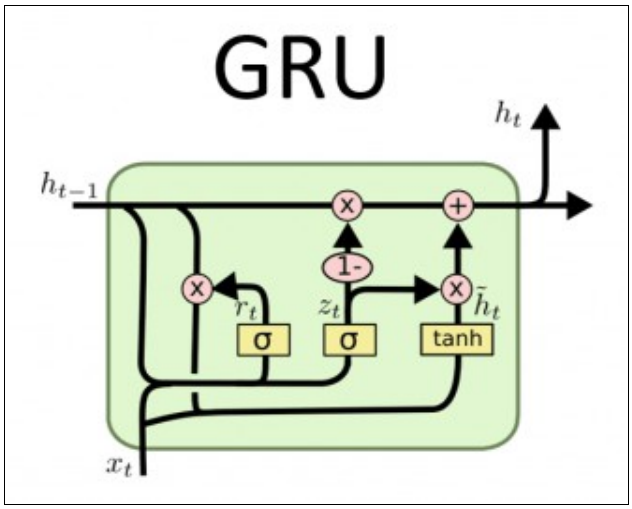


Figura 2.15. Esquema de una celda GRU

3. Análisis y preprocesado del dataset del caso de estudio

3.1 Obtención de los datos

Para la obtención de datos de consumos eléctricos necesarios para este proyecto, he decidido usar un *dataset* ya existente de los consumos de la ciudad de Londres llamada "Smart Meter London" de Jean Michel Daignan. El *dataset* se encuentra publicado en la plataforma *Kaggle* en la siguiente dirección:

https://www.kaggle.com/jeanmidev/smart-meters-in-london

El gobierno británico tiene como objetivo que todo hogar en Inglaterra, Gales y Escocia sustituya los medidores analógicos de consumo eléctrico por un dispositivo *smart meter*, un total de 26 millones de hogares según el gobierno. Dichos dispositivos inteligentes recogen datos sobre el consumo eléctrico de cada hogar y lo registra en un *datastore* disponible para todo el mundo en la siguiente dirección web:

https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households.

Figura 3.1. *Smart meter* instalado por la distribuidora *EDF Energy* en Londres (modelo SGM1312)

Este despliegue de nuevos medidores estaba liderado por la UE (Unión Europea), que pidió a todos sus gobiernos miembros que consideren el uso de medidores inteligentes como parte de las medidas para mejorar nuestro suministro de energías y hacer frente al cambio climático. En España también se empezaron a sustituir los medidores en el año 2010, justo después de la compra de la distribuidora italiana *Enel* sobre *Endesa*, que ya había empezado a sustituirlos en Italia con anterioridad [18][19].

Aunque el *Brexit* haya llegado para los británicos en el año 2020, la mayoría de dichos dispositivos ya se encuentran instalados y funcionando con normalidad, por lo que ya se recogen algunos datos desde finales de 2011.

El *dataset* está creado a partir de estos datos gubernamentales mencionados y recoge datos de un total de 5566 hogares desde noviembre 2011 hasta febrero 2014. Pero dichos datos solo contienen consumos eléctricos, así que el autor del *dataset* ha añadido otros datos que se creen relativos al consumo como las festividades en la capital inglesa, sus datos meteorológicos, perfiles de usuarios (edad, género, economía,…). Estos datos han sido extraídos de los servicios nacionales británicos de estadística y de meteorología [19].

Dicho *dataset* está formado por distintos ficheros, algunos de ellos representan los mismos datos pero en distintos formatos, vayamos a ver con más detalle los ficheros usados para el preprocesado de nuestro dataset:

- Ficheros con el consumo eléctrico de la ciudad

En el directorio *halfhourly_dataset* encontramos 112 ficheros *block_xxx.csv* (de un tamaño entre 60-70 MB) que contienen los siguientes campos:

LCLid	*texto*	Identificador del medidor inteligente
tstp	*fecha/hora*	Fecha en la que se ha tomado la lectura
energy (KWh/hh)	*decimal*	Consumo eléctrico de los últimos 30 minutos en kWh/hh

Con estos ficheros disponemos del consumo eléctrico de cada hogar del *dataset* con una frecuencia de 30 minutos. En resumen disponemos de:

(a) 48 lecturas diarias/hogar
(b) Total de 5.566 hogares
(c) Datos desde 11/2011 hasta 02/2014

```
LCLid,tstp,energy(kWh/hh)
MAC000002,2012-10-12 11:30:00.0000000, 0.143
MAC000002,2012-10-12 12:00:00.0000000, 0.663
MAC000002,2012-10-12 12:30:00.0000000, 0.256
MAC000002,2012-10-12 13:00:00.0000000, 0.155
MAC000002,2012-10-12 13:30:00.0000000, 0.199
MAC000002,2012-10-12 14:00:00.0000000, 0.125
MAC000002,2012-10-12 14:30:00.0000000, 0.165
MAC000002,2012-10-12 15:00:00.0000000, 0.14
MAC000002,2012-10-12 15:30:00.0000000, 0.148
```

Figura 3.2. Muestra del contenido del fichero *block_0.csv*

- Ficheros con festividades en la ciudad

En el fichero *uk_bank_holidays.csv* encontramos una relación de los días festivos de los bancos londinenses. Hay que tener en cuenta que este documento no refleja los días no laborables (sábado y domingo), solo los días festivos.

El fichero lo conforman los siguientes campos:

Bank holidays	*fecha*	Fecha de la festividad
Type	*texto*	Nombre de la festividad

En resumen disponemos de:

(a) 25 días festivos en Londres
(b) Datos desde 12/2012 hasta 01/2014

```
Bank holidays,Type
2012-12-26,Boxing Day
2012-12-25,Christmas Day
2012-08-27,Summer bank holiday
2012-05-06,Queen?s Diamond Jubilee (extra bank holiday)
2012-04-06,Spring bank holiday (substitute day)
2012-07-05,Early May bank holiday
2012-09-04,Easter Monday
2012-06-04,Good Friday
2012-02-01,New Year?s Day (substitute day)
2013-12-26,Boxing Day
```

Figura 3.3. Muestra del contenido del fichero *uk_bank_holidays.csv*

- Ficheros con datos meteorológicos de la ciudad

En el fichero *weather_hourly_darksky.csv* encontramos una relación de datos meteorológicos de la ciudad de Londres con una frecuencia horaria (1h). Los datos son extraídos de la API de Darksky (https://darksky.net/dev/docs) y sus campos son los siguientes:

time	*fecha/hora*	Fecha en la que se ha tomado la lectura
visibility	*decimal*	Distáncia visible (Km)
wind bearing	*decimal*	Velocidad del viento (m/s)
temperature	*decimal*	Temperatura (ºC)
dewpoint	*decimal*	Temperatura de rocío (ºC)
pressure	*decimal*	Presión atmosférica (hPa)
apparent temperature	*decimal*	Sensación térmica (ºC)
wind speed	*decimal*	Dirección del viento (grados)
precipType	*texto*	Descripción del tipo de lluvia [rain, snow, sleet]
icon	*texto*	Resumen meteorológico (*machine-readable*)
humidity	*decimal*	Humedad relativa [0-1]
summary	*texto*	Resumen meteorológico (*human-readable*)

En resumen disponemos de:

(a) 21.165 lecturas meteorológicas de Londres
(b) 24 lecturas diarias
(c) Datos desde 11/2011 hasta 02/2014

Hay que tener en cuenta que estos datos meteorológicos han sido tomados cada 60 minutos, mientras que los consumos eléctricos han sido tomados cada 30 minutos.

```
visibility,windBearing,temperature,time,dewPoint,pressure,apparentTemperature,windSpeed,precipType,icon,humidity,summary
5.97,104,10.24,2011-11-11 00:00:00,8.86,1016.76,10.24,2.77,rain,partly-cloudy-night,0.91,Partly Cloudy
4.88,99,9.76,2011-11-11 01:00:00,8.83,1016.63,8.24,2.95,rain,partly-cloudy-night,0.94,Partly Cloudy
3.7,98,9.46,2011-11-11 02:00:00,8.79,1016.36,7.76,3.17,rain,partly-cloudy-night,0.96,Partly Cloudy
3.12,99,9.23,2011-11-11 03:00:00,8.63,1016.28,7.44,3.25,rain,fog,0.96,Foggy
1.85,111,9.26,2011-11-11 04:00:00,9.21,1015.98,7.24,3.7,rain,fog,1.0,Foggy
1.96,115,9.33,2011-11-11 05:00:00,8.87,1015.91,7.19,3.97,rain,fog,0.97,Foggy
1.3,118,9.31,2011-11-11 06:00:00,8.82,1015.7,7.1,4.1,rain,fog,0.97,Foggy
1.22,114,8.85,2011-11-11 07:00:00,8.69,1016.08,6.48,4.23,rain,fog,0.99,Foggy
1.4,120,9.13,2011-11-11 08:00:00,8.75,1016.33,6.84,4.2,rain,fog,0.97,Foggy
1.38,121,9.23,2011-11-11 09:00:00,8.7,1016.57,7.07,3.96,rain,fog,0.97,Foggy
1.35,115,9.21,2011-11-11 10:00:00,8.76,1016.26,6.96,4.16,rain,fog,0.97,Foggy
1.72,127,9.78,2011-11-11 11:00:00,9.23,1016.17,7.68,4.14,rain,fog,0.96,Foggy
```

Figura 3.4. Muestra del contenido del fichero *weather_hourly_darksky.csv*

3.2 Preprocesamiento de los datos

Para preprocesar los datos del *dataset* original comentado en el apartado anterior nos apoyaremos en el uso de *R*, un lenguaje bajo licencia *GNU* usado para problemas de análisis estadísticos, minería de datos y *machine learning* [21].

Se han realizado los siguientes pasos:

1) Fichero con datos de consumo

(a) Carga de ficheros

En el apartado anterior hemos visto que para los dato de consumo eléctrico se disponen de hasta 112 ficheros con un tamaño medio de 60-70MB. Al tratarse de un tamaño total de datos tan considerable, se ha optado por usar los 10 primeros ficheros, con un total de casi 15 millones de registros de consumo.

(b) Datos nulos y outliers

Se comprueba que no existan datos nulos tras la carga de los documentos. Nos encontramos que existen algunos registros de consumo eléctrico sin valor, así que debemos limpiarlos del *dataframe*.

También existen una serie de *outliers* o valores fuera de lo común, sobretodo en las lecturas de consumo de los primeros meses del *dataset (*nov-dic 2011*),* donde nos encontramos valores nulos o muy superiores a la media (figura 3.5).

Estos *outliers* se deben a que aún había pocos medidores instalados o funcionando con normalidad, así que también nos tendremos que desprender de estos primeros valores y trabajar con valores a partir del año natural 2012.

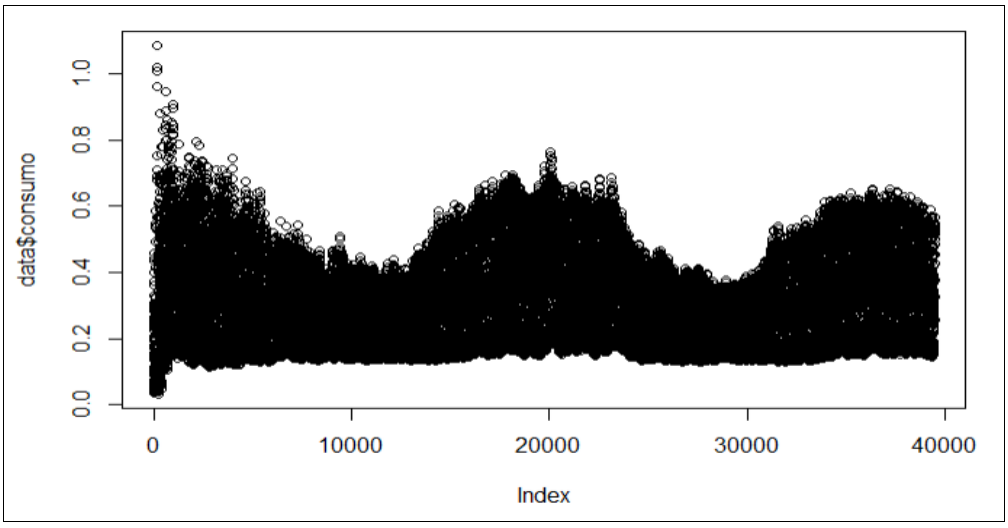


Figura 3.5. D*atos de consumo eléctrico cargados (kWh/hh). Podemos apreciar una serie de valores nulos (o cero) y outliers al principio de la serie.*

(c) Conversión de tipos de datos

Un paso importante para el preprocesado de datos es comprobar si los tipo de datos son correctos. Muchas veces se pueden cargar datos numéricos o fechas como datos de tipo texto. Viendo la figura anterior (figura 3.5) podemos observar como por error la columna de fecha (*tstp*) y la de consumo (*energy.Kwh.hh.*) se representan como texto. Así que debemos convertir estos datos a tipo temporal (*POSIX*) y decimal (*double*) respectivamente. Además aprovecharemos para renombrar las columnas con nombres más claros.

(d) Agrupado por fecha de lecturas

Al tener de un mismo instante distintas lecturas por cada hogar, para conseguir una serie temporal única realizaremos una agrupación de los datos por la columna temporal (*fecha*) con la media aritmética de los valores. También eliminaremos la columna de identificador de hogar (*id*) ya que no nos será de utilidad.

2) Fichero de días festivos

(a) Carga de ficheros

El fichero de festividades tiene un tamaño muy menor con apenas 25 registros. Pero hay que tener en cuenta que en nuestro *dataset* no solo tendremos que registrar los días festivos, sino que también tendremos que registrar los fines de semana.

(b) Valores nulos y outliers

No se encuentran valores nulos o fuera de lo común.

(c) Conversión de tipos de datos

Igual que nos sucedió con la carga de datos de consumos eléctricos, se nos han cargado los datos con tipo de datos *texto*. Deberemos modificar el tipo de dato de fecha a *POSIX* y podemos también eliminar la columna de nombre de festividad ya que no nos será de interés para el *dataset*.

(d) Añadir nueva columna al dataset

A partir del *dataframe* anterior de consumos eléctricos, añadiremos una nueva columna “festivo” donde marcaremos si el día de la fecha de la lectura era día festivo o no laborable.

Por defecto, marcaremos la columna con valor 0 indicando que se trata de un día laborable. Luego emparejamos las fechas del *dataframe* de festividades con el *dataframe* de consumos indicando con valor 1 aquellas fechas coincidentes o que hacen *match*.

De igual forma, con ayuda de la función *isWeekend* marcaremos con un 1 aquellas fechas que representen días no laborables (sábados y domingos).

3) Fichero de datos meteorológicos

(a) Carga de ficheros

Los datos meteorológicos constan de un solo fichero, del cual nos quedaremos con las columnas de valores numéricos. Es decir, desecharemos los valores representados con texto ya que no los podemos computar en los futuros modelos predictivos.

(b) Datos nulos y outliers

Se comprueba que no existan datos nulos tras la carga de los documento. Nos encontramos con 13 datos de presión atmosférica sin valor. Al ser tan pocos casos, los rellenaremos con el mismo valor anterior de la serie temporal. No podemos calcular medias ya que 4 de los nulos son contiguos. No se cree que vayan a afectar a los resultados.

(c) Conversión de tipos de datos

De igual forma que con las cargas anteriores, los datos se han cargado en formato texto y deberemos modificar los tipos que son numéricos pasándolos a tipo decimal (*double*).

(d) Frecuencias de las lecturas

Al querer hacer *merge* del *dataframe* de datos meteorológicos en el *dataframe* de consumos y festivos, nos encontramos con un problema de frecuencias en las fechas.

Los datos de medidas de consumo se han tomado con una frecuencia de 30 minutos, es decir, existen 48 lecturas por día. Mientras que los datos meteorológicos han sido tomados con una frecuencia horaria, es decir, 24 lecturas por día.

Si optamos por seguir con el *merge*, nos encontramos que perderemos el 50% de los datos que ya teníamos en el *dataframe* de consumos y festivos, más de 19k registros. Nos quedaríamos solo con los datos de consumo de cada hora.

Para no perder tanta información, la solución por la que se ha optado es la siguiente: duplicaremos los registros de datos meteorológicos (x2) sumándole a la columna de fecha 30 minutos y aplicaremos en cada columna numérica de los nuevos registros la media entre sus valores vecinos (valor anterior y posterior).

Es decir, a modo de ejemplo, si tenemos que un día a las 8:00h había una temperatura de 25ºC y el mismo día a las 9:00h habían 27ºC, insertaremos un nuevo registro en el *dataframe* con la hora 8:30h con 26ºC.

Ahora sí, una vez creados estos nuevos registros, procederemos a hacer el *merge* con los datos de consumo y festivos, consiguiendo como resultado un *dataset* con casi 40k registros de consumo eléctrico, datos meteorológicos y datos de calendario laboral.

3.3 Análisis del dataset resultante

Una vez hemos limpiado y preprocesado el *dataset* original, podemos realizar un pequeño análisis del *dataset* obtenido. En cuanto a tamaño de los datos podemos decir que:

- Hay 39.471 registros.
- Hay 11 variables: *fecha*, *consumo*, *festivo*, *visibility*, *windBearing*, *temperature*, *dewPoint*, *pressure*, *apparentTemperature*, *windSpeed*, *humidity*.

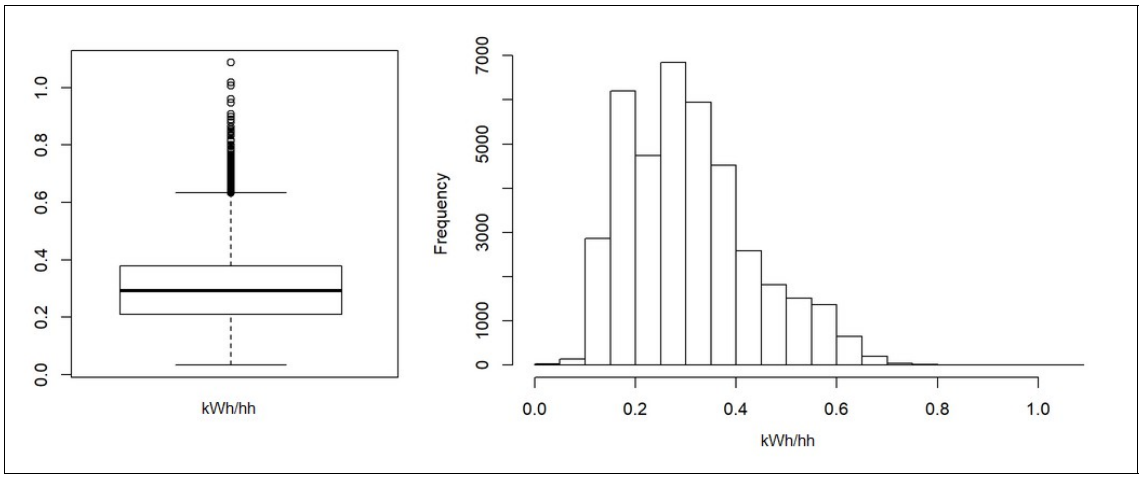
```
   fecha              consumo          festivo         visibility
Length:39471       Min.   :0.0335   Min.   :0.0000   Min.   : 0.27
Class :character   1st Qu.:0.2093   1st Qu.:0.0000   1st Qu.:10.37
Mode  :character   Median :0.2919   Median :0.0000   Median :12.32
                   Mean   :0.3086   Mean   :0.3051   Mean   :11.29
                   3rd Qu.:0.3788   3rd Qu.:1.0000   3rd Qu.:13.10
                   Max.   :1.0870   Max.   :1.0000   Max.   :16.09

    windBearing     temperature        dewPoint         pressure
   Min.   :  0.0   Min.   :-5.640   Min.   :-9.980   Min.   : 975.7
   1st Qu.:134.0   1st Qu.: 6.425   1st Qu.: 2.760   1st Qu.:1007.3
   Median :218.0   Median : 9.975   Median : 6.615   Median :1014.5
   Mean   :197.5   Mean   :10.528   Mean   : 6.551   Mean   :1014.0
   3rd Qu.:256.0   3rd Qu.:14.540   3rd Qu.:10.420   3rd Qu.:1021.7
   Max.   :359.0   Max.   :32.400   Max.   :19.880   Max.   :1043.3

   apparentTemperature   windSpeed         humidity
   Min.   :-8.880      Min.   : 0.040   Min.   :0.23
   1st Qu.: 3.810      1st Qu.: 2.440   1st Qu.:0.70
   Median : 9.125      Median : 3.710   Median :0.81
   Mean   : 9.268      Mean   : 3.932   Mean   :0.78
   3rd Qu.:14.540      3rd Qu.: 5.110   3rd Qu.:0.89
   Max.   :32.420      Max.   :14.800   Max.   :1.00
```

Figura 3.6. *Resumen del dataset resultante*

Podemos observar en las siguientes figuras (3.7 y 3.8) los consumos eléctricos de nuestro *dataset* en forma de diagrama de caja y de histograma.

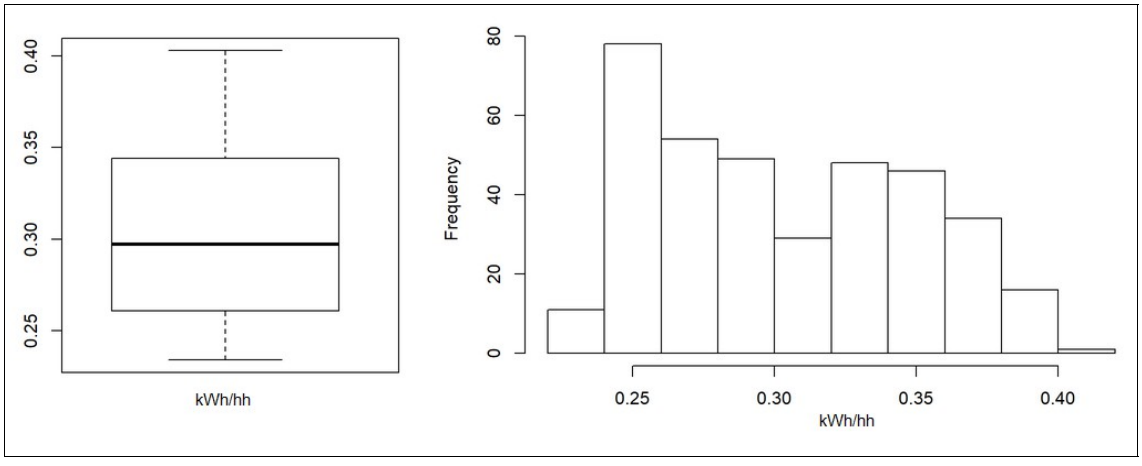
Observamos como los consumos tienen una mediana de 0.3 kWh/hh con un máximo de hasta 0.6 kWh/hh. En el histograma también observamos una moda sobre los 0.3 kWh/hh.

También podemos ver los *outliers* que ya hemos mencionado en el apartado anterior en el diagrama de caja.



Figuras 3.7 y 3.8. *Diagrama de caja e histograma de consumos (kWh/hh)*

Si agregamos estos datos por fecha (diariamente), tenemos que el pico máximo de consumo cae hasta los 0.4 kWh/hh y la moda se posiciona en los 0.25 kWh/hh. La mediana se mantiene en el mismo valor (0.3 kWh/hh) (figuras 3.9 y 3.10).



Figuras 3.9 y 3.10. *Diagrama de caja e histograma de consumos medios por día (kWh/hh)*

A continuación analizaremos las correlaciones entre las distintas variables del *dataset*. Para ello nos ayudaremos de la siguiente imagen (figura 3.11) donde se muestra un mapa de calor entre las variables o *features* del *dataset*.

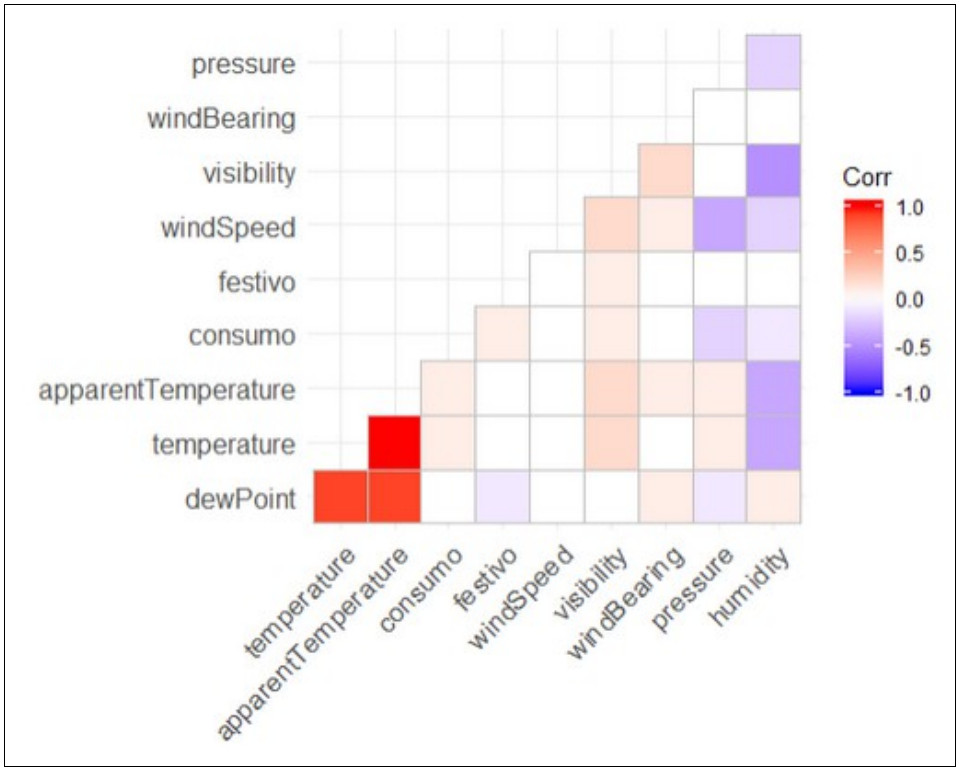


Figura 3.11. *Correlación entre las variables o características del dataset*

Se observan como las variables *temperature*, *dewPoint* (temperatura de rocío) y *apparentTemperature* (temperatura ambiente) tienen una alta correlación. Se podía prever ya que estas dos últimas variables dependen mucho de la temperatura. Las podemos obviar en el momento de crear nuestros modelos predictivos.

Sobre el consumo eléctrico, variable por la cual realizaremos las predicciones, vemos que no tiene ninguna correlación muy fuerte con los datos meteorológicos ni el de calendario. Pero se pueden observar correlaciones débiles con las siguientes variables:

1. temperatura
2. festivo (días no laborable)
3. visibilidad

Sobre la temperatura, no es ninguna sorpresa, ya hemos comentado en la introducción de esta memoria que las series temporales de consumos eléctricos acostumbran a ser estacionarias según la estación del año. En cuanto a la visibilidad, también entendemos que en horas nocturnas hay más consumo que en horas diurnas. Finalmente, sobre los festivos, entendemos que los fines de semana hay menor consumo ya que las fábricas no abren, pero hay que recordar que este *dataset* se basa solo en consumos de hogares. Así que debe ser por otra causa distinta como por ejemplo aspectos socioculturales de la población como puede ser el ocio.

En la siguiente imagen (figura 3.12) podemos ver como los datos de consumo (en kWh/hh) representan una serie temporal estacionaria, ya que dependiendo del mes en curso el consumo crece y decrece haciendo que la media y la varianza sean constantes en el tiempo. También se trata de una serie estacional ya que los patrones se repiten de forma periódica provocando una tendencia constante en cada periodo.

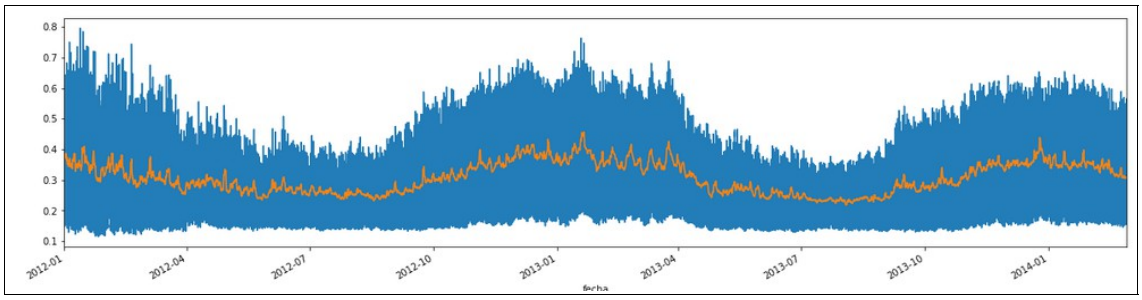


Figura 3.12. *Serie de consumo eléctrico (azul) y su tendencia (naranja)*

A continuación analizaremos con más detalle la correlación entre consumo y temperatura del año 2013. Se muestra en la siguiente figura (figura 3.13) la serie temporal de consumo medio diario (kWh/hh) y la de temperatura (ºC) con los datos normalizados.

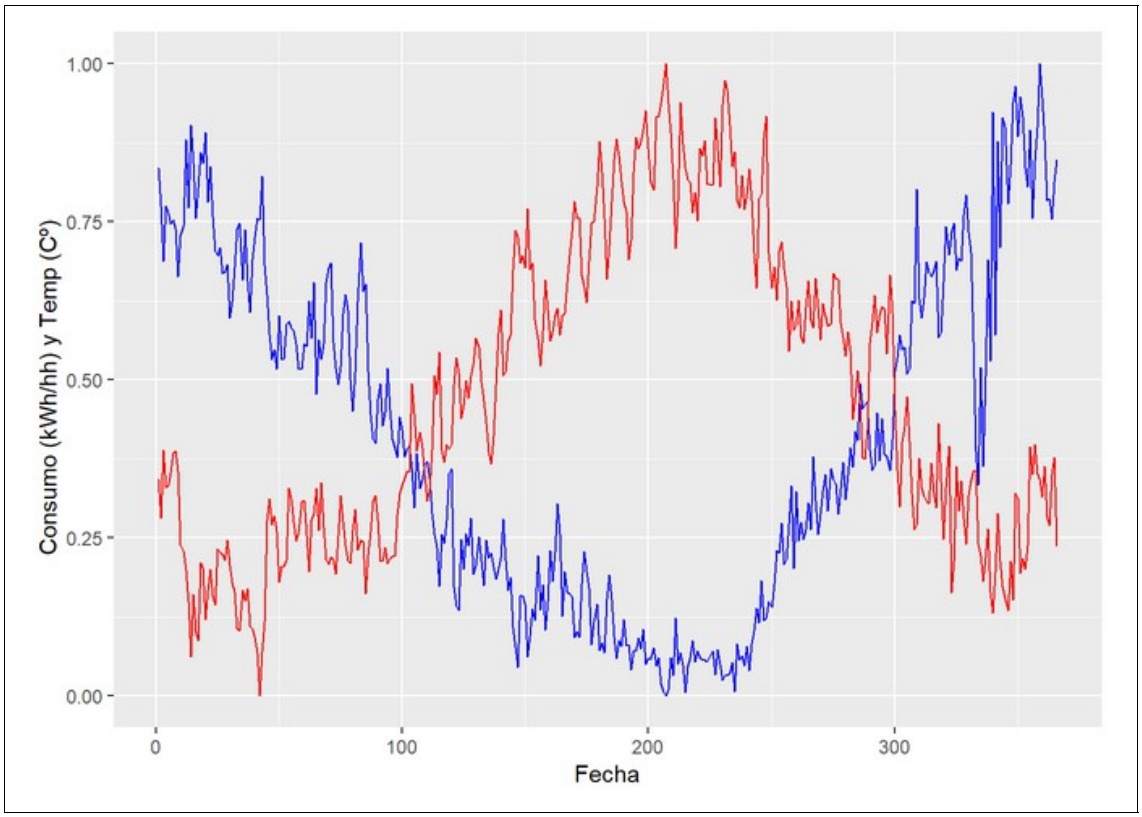


Figura 3.13. *Serie de consumo eléctrico (azul) y temperatura (rojo) normalizadas [0,1]*

Se puede apreciar como los meses de verano, con mayor temperatura, el consumo eléctrico en los hogares desciende. Este hecho debe ser, lo más probable, que en verano no se usan sistemas de calefacción eléctrica ni calentadores de agua.

Hay un dato peculiar en la serie temporal de consumo (línea azul) sobre el mes de noviembre 2013 (índice ~350 en el gráfica). Se puede apreciar un descenso importante durante unos días del consumo eléctrico en los hogares. Si investigamos sobre acontecimientos ocurridos en ese otoño 2013 en el Reino Unido, encontramos que justamente durante esas fechas se produjo una tormenta que afectó al sur de Inglaterra, Holanda y norte de Francia con importantes destrozos y varias pérdidas humanas [22][23]. La llamada tormenta de 'San Judas' causó distintas inundaciones que dejaron sin luz a miles de hogares del sur de Inglaterra y canceló durante algunos días los vuelos de aeropuertos londinenses.

Suponemos que la causa de ese descenso de consumo viene provocado por dicha tormenta. Ya comentamos también en la introducción de esta memoria que una de las posibles causas en variaciones del consumo eléctrico pueden ser acontecimientos puntuales como daños meteorológicos o crisis económicas y/o sanitarias como la que estamos viviendo hoy en día.

Finalmente, sobre las festividades, hemos visto que existía una correlación entre días laborables y consumo eléctrico. En la siguiente figura (figura 3.14) vemos que los días no laborables tienen un mayor consumo eléctrico que los días laborables.

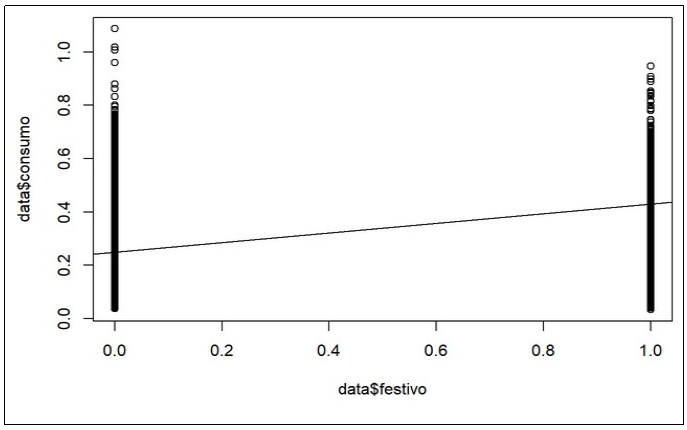


Figura 3.14. *Consumos eléctricos (kWh/hh) en días laborables (festivo=0) y días festivos (festivo=1)*

Normalmente se piensa que los días festivos el consumo es menor ya que lugares de trabajo: oficinas, fábricas, etc. se encuentran cerrados. Pero como ya hemos dicho, este *dataset* se basa en datos de consumo de hogares londinenses, así que supongo que si el consumo es mayor cuando la gente no trabaja es porque pasan más tiempo en el hogar o bien consumen más ocio (centros comerciales, discotecas, parques de atracciones, etc). Así que al no disponer de consumos industriales o de oficinas, hay que dar pie a que se trata de un aspecto socio-económico en particular de los habitantes de Londres.

El código en R para el preprocesamiento y análisis del *dataset* se puede encontrar en:
https://github.com/cfiguerap/uoc-data-science-tfm/tree/master/src/0-dataset-preprocessing

4. Creación y entrenamiento de los modelos predictivos

4.1 Modelos usados en el estudio

De la possible colección de modelos predictivos que podemos probar con nuestro *dataset,* se han escogido los modelos más usados para el tipo de problema que queremos resolver: la predicción de series temporales.

De entre ellos, podemos diferenciar dos grandes grupos de modelos: los modelos tradiciones y los modelos de aprendizaje automático (ML). Los modelos usados son los siguientes:

Modelos tradicionales	ARIMA (*autoregressive integrated moving average)*
Modelos deep learning	LSTM (*Long short-term memory*)
	GRU (*Gated recurrent unit*)
	CNN (*Convolutional neural network*)
	Modelo híbrido: CNN-LSTM

La gran diferencia se encuentra entre el modelo ARIMA y el resto. Se trata de un modelo estadístico formalizado por *Box-Jenkins* en los años setenta para la predicción de serie univariables, mientras que el resto de modelos se basan en *deep learning* y fueron formalizados muchos años después (entre 20 y 30 años dependiendo del modelo) para series temporales univariables y multivariables.

Todos los modelos se han implementado en *Python*, un lenguaje de programación interpretado y multiplataforma. Usando principalmente el paquete *statsmodels* para los modelos estadísticos y *Keras* para los de deep learning.

El paquete *statsmodels* es un módulo estadístico para *Python* que permite la exploración de datos y creación de modelos estadísticos. Mientras que *Keras* es una librería de código abierto para la creación de redes neuronales en *Python* que ofrece una API (*application programming interface*) que facilita el uso de *TensorFlow*, otra biblioteca muy popular para redes neuronales desarrollada por *Google*.

4.2 Modelos Tradicionales: ARIMA

El módulo estadístico de *Python* ofrece un método para modelar ARIMA en el siguiente paquete: `statsmodels.tsa.arima_model.ARIMA`. Pero al tratar una serie temporal estacional, se ha usado otro método más acorde a la naturaleza de nuestros datos: `statsmodels.tsa.statespace.sarimax.SARIMAX`

Aunque uno de los métodos estadísticos más usados para la predicción de series temporales univariadas es el método ARIMA, usaremos su variante extendida SARIMA por la siguiente razón: aunque ARIMA trabaja y da buenos resultados con series temporales con tendencia, no soporta tan bien series con componentes estacionales como en nuestro caso de investigación.

Aún así, el proceso es muy similar. Primero de todo cargaremos los datos de nuestro *dataset* y prepararemos los datos para el entrenamiento de nuestro modelo. Para este tipo de modelos, en comparación con los modelos de redes neuronales, he sufrido muchos problemas de memoria en la computadora donde se han ejecutado las distintas predicciones. Por eso, para este modelo se ha tenido que trabajar con un conjunto de entrenamiento mucho más pequeño y seguramente esto afecte a la futura comparativa entre modelos.

Se ha escogido un conjunto de entrenamiento de 10.000 lecturas de consumo y un conjunto de test de 1.440 lecturas, que representan los datos de 30 días de consumo. Recordamos que nuestro *dataset* dispone de 48 lecturas de consumo diarias. No se ha podido usar el total del *dataset* a causa de que este modelo tiene un coste computacional y tiempo de entrenamiento muy elevado (de la magnitud de minutos).

Como ya se explicó con anterioridad (apartado 2.5), el modelo ARIMA (y SARIMA a su vez) trabajan observando diferencias entre los valores de la serie temporal y necesitan de 3 parámetros iniciales:

- ***P***: Número de observaciones pasadas (llamado orden de retraso).
- ***D***: Número de veces que los valores son diferenciadas (grado de diferencia).
- **Q**: Tamaño de la ventana de promedio móvil (orden de promedio móvil).

Para el cálculo de los parámetros más óptimos se puede usar la función `auto_arima` del paquete `pmdarima`. Esta función, prueba de forma empírica hasta encontrar los tres parámetros que mejor resuelven la predicción de datos futuros de nuestra serie temporal. Hay que advertir que al probar distintos parámetros el tiempo de ejecución es muy elevado.

Una vez se encuentran los datos iniciales más óptimos, en nuestro caso han sido: P(2), D(1) y Q(2), podemos crear nuestro modelo predictivo obteniendo el siguiente resumen del modelo obtenido (figura 4.1).

En dicho resumen aparecen los valores de AIC, BIC y HQIC de nuestro modelo. Estos valores representan los criterios de información *Akaike* (AIC), *Bayesiano* (BIC) y *Hannan-Quinn* (HQIC). Estos criterios muestran una medida de calidad relativa de un modelo estadístico. Según el método `auto_arima` que hemos usado con anterioridad, este modelo resultante ha obtenido los mejores criterios.

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	10000
Model:	SARIMAX(2, 1, 2)x(2, 1, 2, 48)	**Log Likelihood**	22978.913
Date:	Mon, 27 Apr 2020	**AIC**	-45939.826
Time:	10:58:50	**BIC**	-45874.977
Sample:	0	**HQIC**	-45917.870
	- 10000		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.0607	0.185	-0.328	0.743	-0.423	0.302
ar.L2	0.0809	0.083	0.975	0.330	-0.082	0.244
ma.L1	-0.1318	0.184	-0.715	0.475	-0.493	0.230
ma.L2	-0.1955	0.116	-1.682	0.093	-0.423	0.032
ar.S.L48	-0.5911	0.188	-3.141	0.002	-0.960	-0.222
ar.S.L96	0.0098	0.015	0.652	0.514	-0.020	0.039
ma.S.L48	-0.2704	0.188	-1.437	0.151	-0.639	0.098
ma.S.L96	-0.5769	0.172	-3.351	0.001	-0.914	-0.239
sigma2	0.0006	4.87e-06	118.108	0.000	0.001	0.001

Ljung-Box (Q):	413.56	**Jarque-Bera (JB):**	6313.93
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.16	**Skew:**	0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	6.89

Figura 4.1. *Resumen del modelo SARIMA obtenido con los parámetros P(2), D(1) y Q(2)*

Una vez obtenido el modelo, ya podemos proceder a su entrenamiento y finalmente a su evaluación de predicción. También hay que mencionar que el entrenamiento de nuestro conjunto de datos tiene un tiempo de ejecución elevado, del orden de distintos minutos. Para la evaluación de la predicción obtenida se ha usado el valor medio de diferentes ejecuciones con distintos datos de la raíz del error cuadrático medio (RMSE) obtenido al diferenciar la serie temporal real con la serie pronosticada por nuestro modelo SARIMA.

En este caso, el valor medio de RMSE ha sido de: **0.02146**

A continuación mostramos ejemplos gráficos de la serie temporal real (en azul) y la pronosticada por nuestro modelo (en rojo) sobre los datos del conjunto de test sobre 1 semana y 1 día (figuras 4.2 y 4.3 respectivamente).

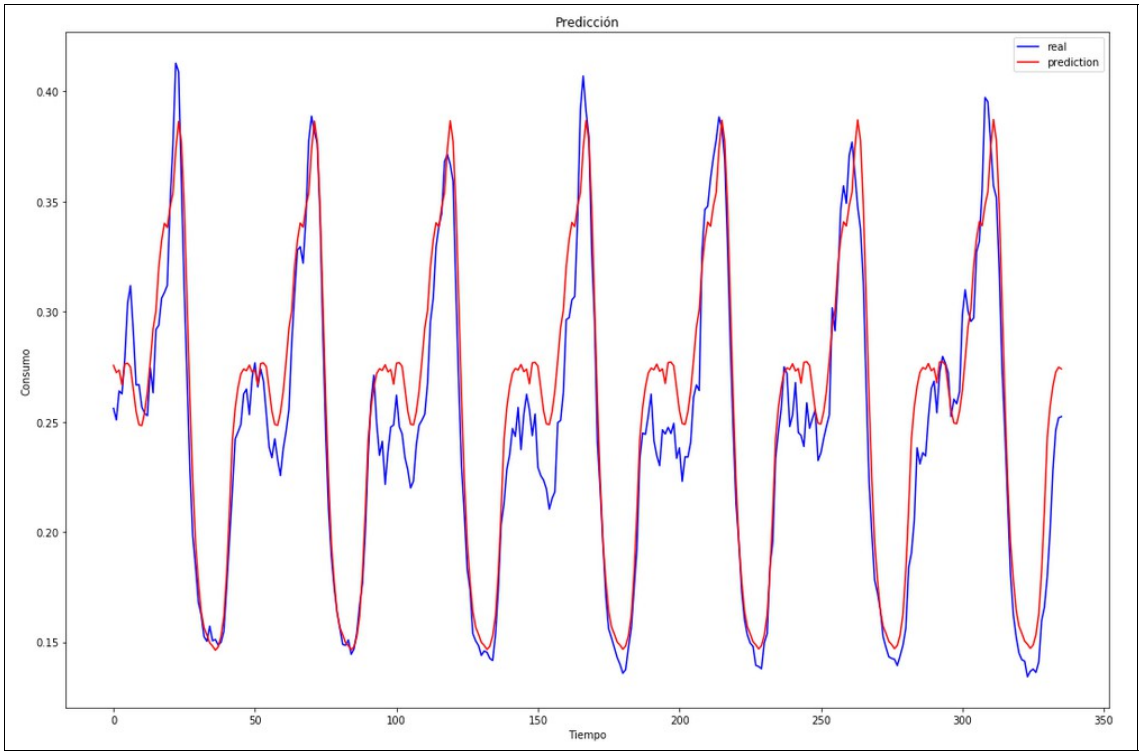


Figura 4.2. *Predicción de consumo en kWw/hh (en rojo) sobre la serie original (en azul) sobre 7 días del modelo SARIMA implementado*

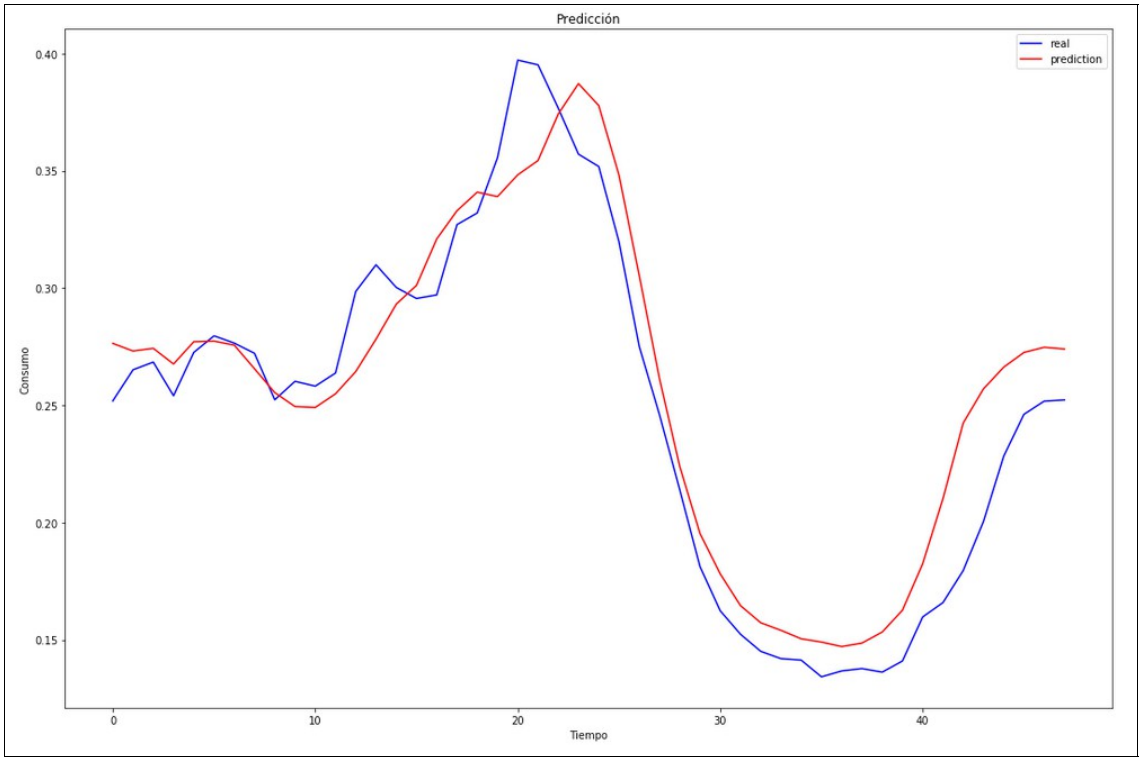


Figura 4.3. *Predicción de consumo en kWw/hh (en rojo) sobre la serie original (en azul) sobre 24 horas del modelo SARIMA implementado*

4.3 Modelos Deep Learning

A continuación se muestran los modelos del tipo red neuronal creados para la predicción del consumo eléctrico. A diferencia del modelo anterior, trabajaremos también con datos meteorológicos y de calendario (días festivos).

Otra diferencia con el modelo anterior es que usaremos *Keras*, con esta librería los modelos necesitan que se les indique las entradas conocidas y salidas conocidas de nuestro conjunto de entrenamiento por separado. Pero al tratarse de una serie temporal, lo que se va a realizar es una transformación de los datos de la siguiente manera:

- Los datos de salida serán el valor de la serie temporal en el instante [*t*].
- Los datos de entrada serán un array de los valores de la serie temporal en los instantes anteriores: [*t-n, ..., t-3, t-2, t-1*] donde *n* indica la longitud del *array*.
- Este parámetro *n* en el código fuente de este proyecto lo llamamos *lookback*.

En resumen, lo que indica el parámetro *lookback* es el número de valores que añadimos como parámetros de entrada a la red neuronal, que son anteriores y contiguos al instante que estamos añadiendo a la red neuronal como salida conocida. Es decir, a la red neuronal se le pasa como entrada una ventana de valores anteriores y como salida el valor posterior. El parámetro *lookback* determina el tamaño de esta ventana de datos de aprendizaje.

Ejemplo: transformación de la entrada de la serie temporal a ventana.

Si nuestra serie temporal es un *array*
de 12 elementos como el siguiente:

[0.1, 0.4, 0.6, 0.8, 0.9, 1.1, 1.2, 1.3, 1.0, 0.8, 0.6, 0.5]

Los parámetros de entrada y salida de nuestra red neuronal para un *lookback=4 serán los siguientes, tendremos 12-4=8 ventanas*:

Entrada 1: **[0.1, 0.4, 0.6, 0.8]**, Salida 1: **[0.9]**
Entrada 2: **[0.4, 0.6, 0.8, 0.9]**, Salida 2: **[1.1]**
Entrada 3: **[0.6, 0.8, 0.9, 1.1]**, Salida 3: **[1.2]**
Entrada 4: **[0.8, 0.9, 1.1, 1.2]**, Salida 4: **[1.3]**
Entrada 5: **[0.9, 1.1, 1.2, 1.3]**, Salida 5: **[1.0]**
Entrada 6: **[1.1, 1.2, 1.3, 1.0]**, Salida 6: **[0.8]**
Entrada 7: **[1.2, 1.3, 1.0, 0.8]**, Salida 7: **[0.6]**
Entrada 8: **[1.3, 1.0, 0.8, 0.6]**, Salida 8: **[0.5]**

De esta forma la red neuronal, al ser entrenada por ventanas, puede predecir un valor futuro a partir de los valores pasados. Pero hay que tener en cuenta el tamaño de la ventana o *lookback*. Si la ventana es demasiado grande la red neuronal puede tender a sobreentrenarse (*overfitting*). Si es demasiado pequeña, veremos que el modelo predice con una mala precisión (*underfitting*).

Hemos comprobado empíricamente que con un lookback de 48 elementos (datos de 1 día), el modelo tiene una buena precisión. En la siguiente figura (figura 4.4) podemos ver qué ocurre si usamos una ventana muy pequeña (*lookback*=4).

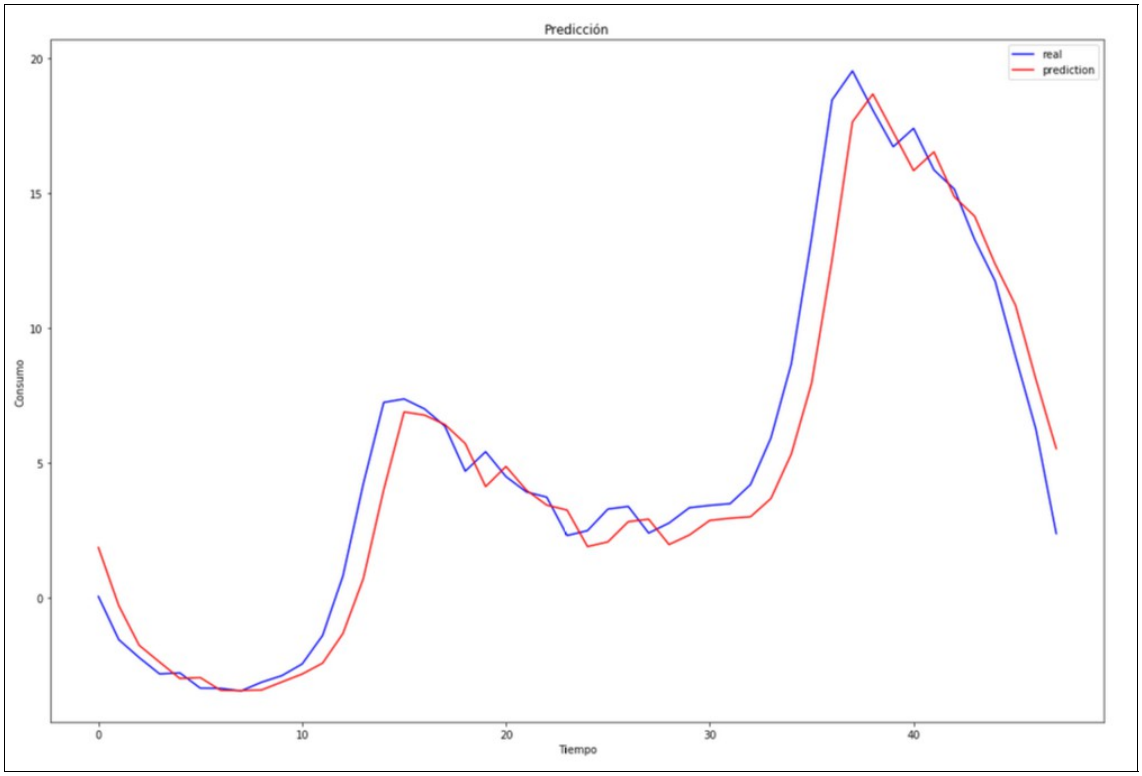


Figura 4.4. *Predicción de consumo en kWw/hh (en rojo) sobre la serie original (en azul) sobre 24 horas por una NN con lookback=4.*

Podemos apreciar como la serie temporal real y la pronosticada por la red neuronal produce como un efecto de *lag* hacia la derecha. Este efecto se produce porque, al tener pocos datos de entrada, nuestra red neuronal ha aprendido que el valor futuro en el instante [t] siempre es muy cercano al valor del instante anterior [t-1], así que siempre nos devuelve el último valor conocido o un valor muy semejante.

Pero además de datos de consumo, disponemos de datos meteorológicos y de calendario en nuestro *dataset*. Añadiremos a cada ventana de datos de consumo aquellos parámetros que han reportado mayor correlación en el análisis previo del capítulo anterior: temperatura, día laborables y visibilidad.

Por otro lado, en cuanto al tamaño de los conjuntos de entrenamiento, podemos usar conjuntos más grandes ya que el uso de redes neuronales requiere de unos requisitos de hardware más bajos que los que hemos necesitado con ARIMA.

En este apartado de modelos *deep learning* hemos partido el *dataset* en un conjunto de test de 3 meses (4.320 lecturas del año 2014) y el resto como conjunto de entrenamiento o aprendizaje, un total de 33.448 lecturas que representan casi 2 años de medidas (2012 y 2013). Así que, en resumen, tenemos que el conjunto de entrenamiento representa el 87% de los datos y el de validación un 13%.

4.3.1 Modelo LSTM

El modelo LSTM (*Long short-term memory*) es una extensión de las RNN (*recurrent neural network*), que básicamente tienen una memoria para aprender de experiencias importantes que han pasado a lo largo del tiempo.

El modelo escogido para nuestras pruebas tiene la siguiente configuración (figura 4.5). Se trata de un modelo formado por una capa de 50 neuronas, que recibe los datos de consumo eléctrico de un día (48 mediciones) y devuelve un valor que representará el siguiente valor futuro o lectura pronosticada por nuestra red neuronal.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 50)                19800
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 19,851
Trainable params: 19,851
Non-trainable params: 0
_________________________________________________________________
```

Figura 4.5. *Definición de nuestro modelo LSTM.*

En la siguiente figura (figura 4.6) mostramos la tasa de error en entrenamiento y validación de nuestro modelo. El error en entrenamiento (*loss*) nos indica la tasa de error de nuestro modelo durante su entrenamiento, mientras que el de validación (*val_loss*) refleja la misma tasa pero con datos del conjunto de test. Un buen modelo que generaliza su aprendizaje muestra estas dos tasas con tendencia a cero. En caso contrario, indicaría que el modelo tiende a sobreentrenar (*overfitting*) o dispone de pocos datos para generalizar su aprendizaje (*underfitting*).

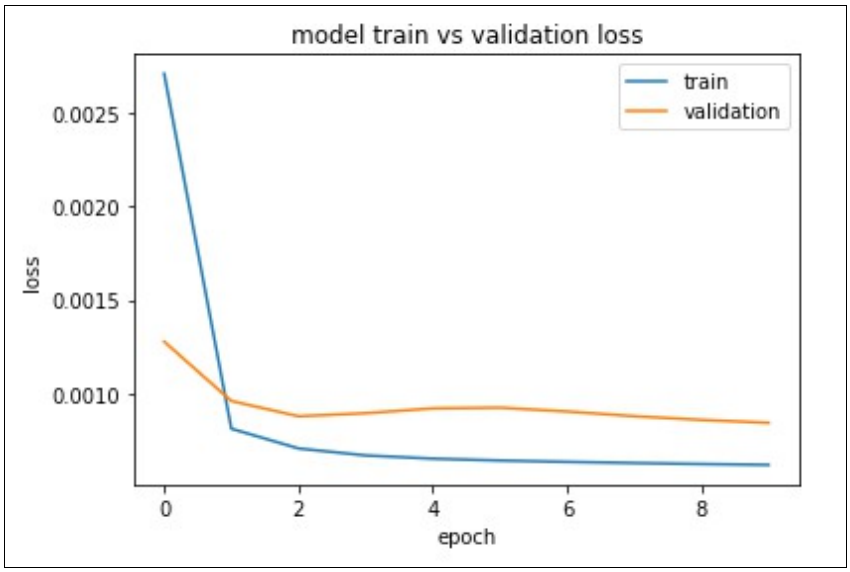


Figura 4.6. *Tasa de error de entrenamiento (azul) y validación (naranja) de nuestro modelo LSTM (solo datos consumo)*

Tras 10 épocas de entrenamiento, podemos observar como la tasa de error de entrenamiento (en azul) tiende a cero rápidamente, mientras que la de validación (en naranja) llega a subir en alguna época pero pronto vuelve a tener una tendencia a cero. Esto significa que nuestro modelo tendrá un buen comportamiento.

Para la evaluación de la predicción obtenida, de igual forma que con el resto de modelos, se usará el valor medio de diferentes ejecuciones de la raíz del error cuadrático medio (RMSE) obtenido al diferenciar la serie temporal real con la serie pronosticada por nuestro modelo.

En este caso, el valor medio de RMSE ha sido de: **0.01956**

A continuación mostramos un ejemplos gráfico de la serie temporal real (en azul) y la pronosticada por nuestro modelo (en rojo) sobre los datos del conjunto de test sobre 1 día (figura 4.7).

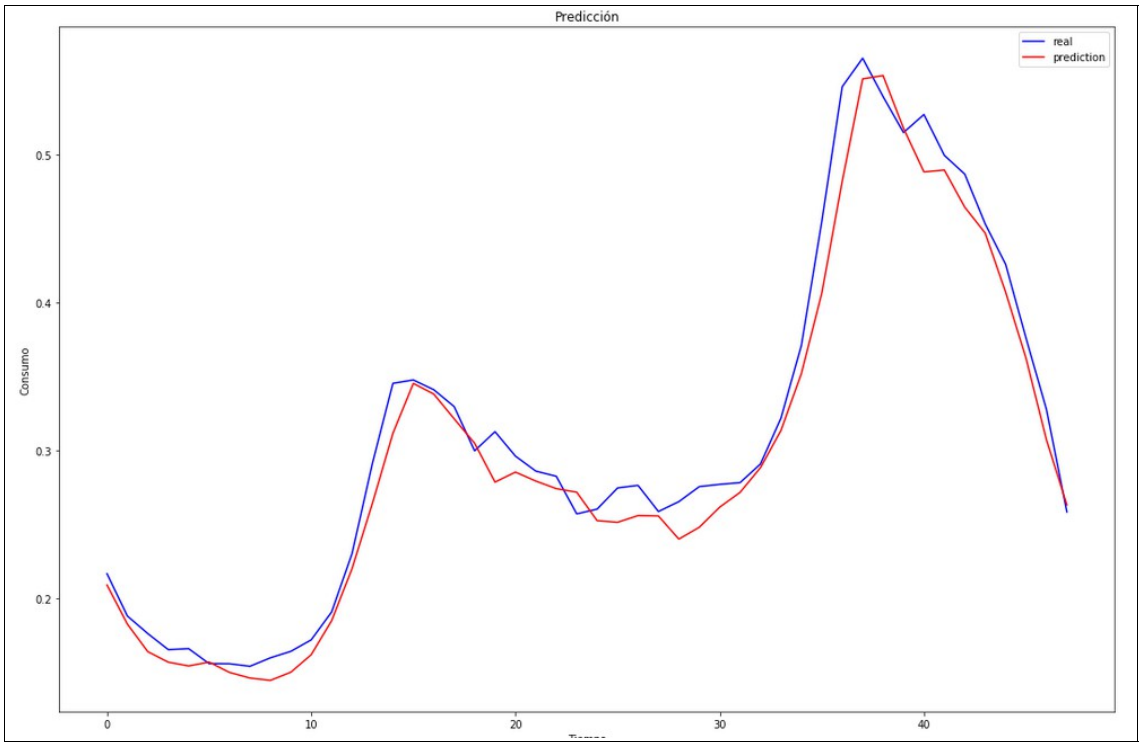


Figura 4.7. *Predicción de consumo en kWw/hh (en rojo) sobre la serie original (en azul) sobre 24 horas del modelo LSTM implementado (solo datos consumo)*

Si añadimos en el *dataset* de entrada los valores que disponemos de meteorología (temperatura y visibilidad) y de calendario (días laborables y festivos), obtenemos las siguientes tasas de error (figura 4.8) y un nuevo valor medio de RMSE que desciende hasta: **0.01412**.

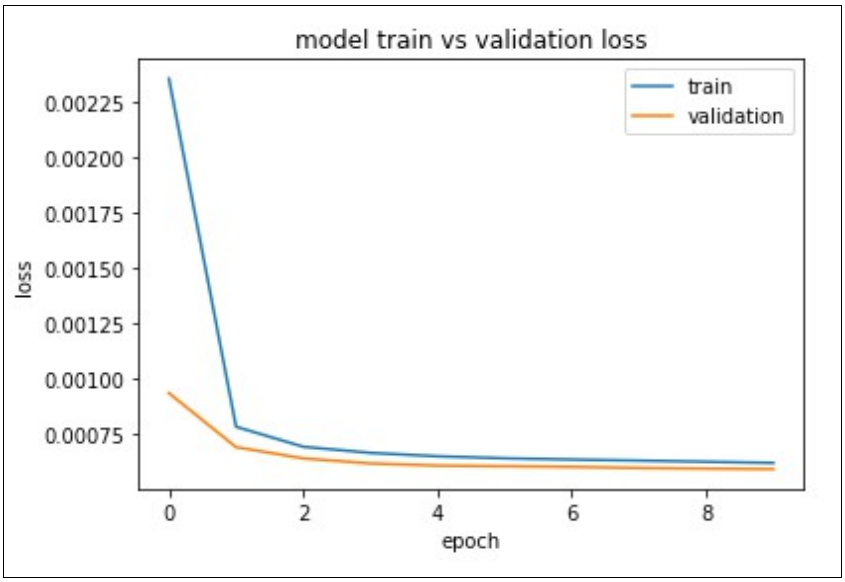


Figura 4.8. *Tasa de error de entrenamiento (azul) y validación (naranja) de nuestro modelo LSTM*

Podemos observar como la tasa de error de entrenamiento y validación descienden más rápidamente hacia 0, lo que indica que este nuevo modelo ha generalizado mejor su aprendizaje y tiene un error de predicción un 28% menor.

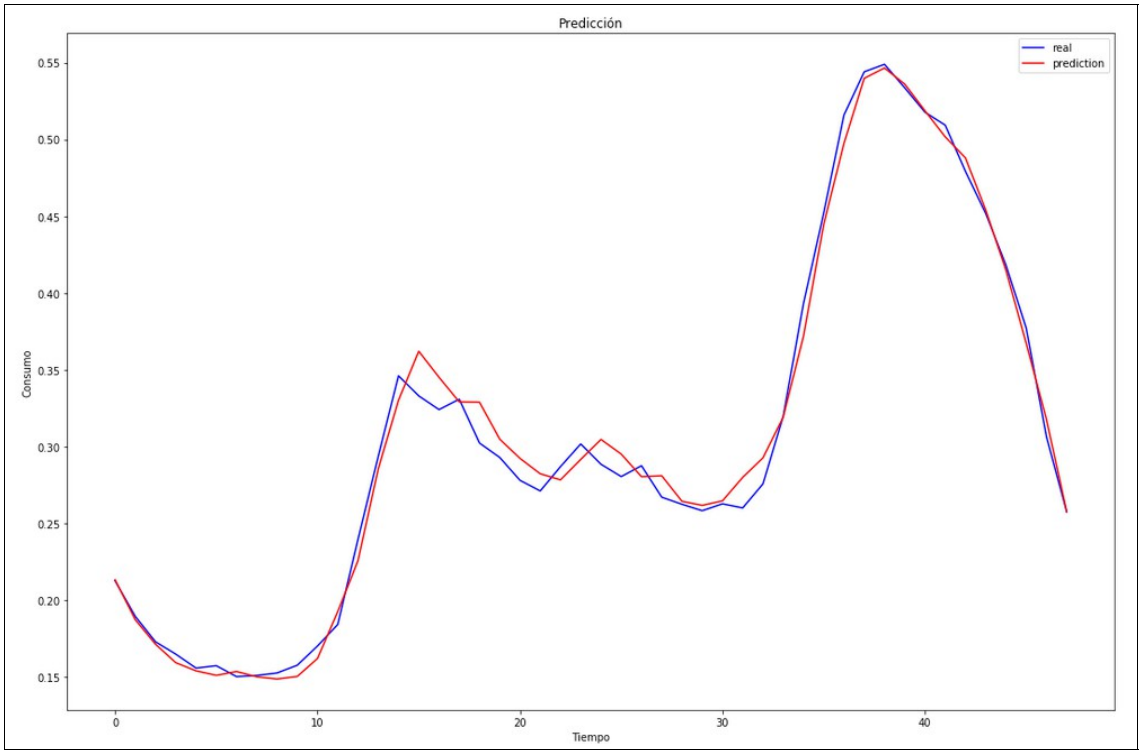


Figura 4.9. *Predicción de consumo en kWw/hh (en rojo) sobre la serie original (en azul) sobre 24 horas del modelo LSTM multivariado*

4.3.2 Modelo GRU

El modelo GRU (*Gated recurrent unit*) es un modelo que surgió en 2014 ofreciendo un resultado muy similar al LSTM con un coste computacional menor. Si entrenamos este modelo con la misma configuración que el modelo anterior, obtenemos las siguientes tasas de error (figuras 4.10 y 4.11).

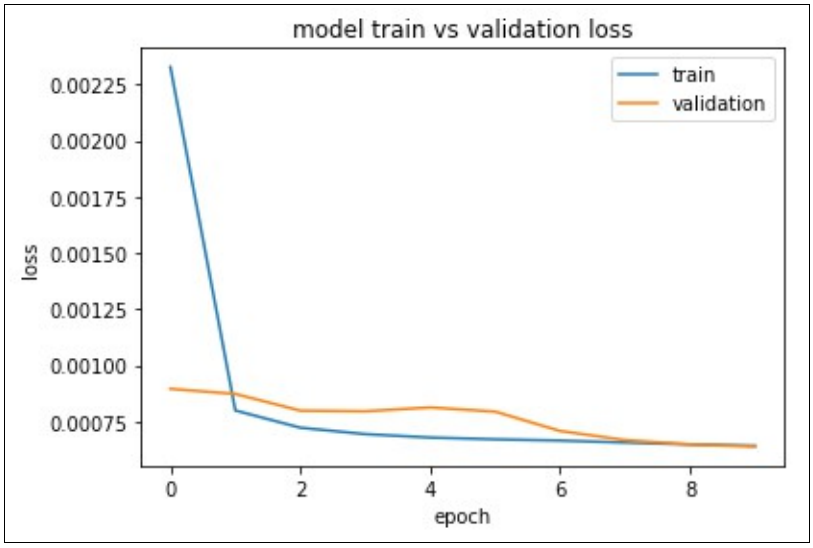


Figura 4.10. *Tasa de error de entrenamiento (azul) y validación (naranja) de nuestro modelo GRU (solo datos consumo)*

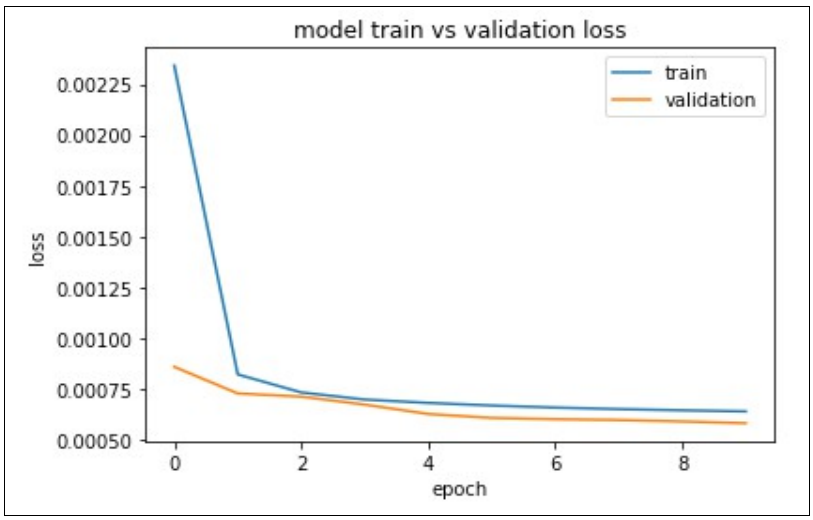


Figura 4.11. *Tasa de error de entrenamiento (azul) y validación (naranja) de nuestro modelo GRU*

Tenemos un comportamiento muy parecido con el modelo LSTM, donde en el entrenamiento con solo datos de consumo (univariable), vemos como la tasa de error de validación no desciende tan rápidamente, hecho que nos hace pensar si existe algún tipo de sobreentrenamiento si solo usamos datos de consumo.

Esta vez el valor medio de RMSE ha sido de **0.01608** en el caso del modelo univariable y de **0.01427** en el modelo multivariable, lo que representa un 12% de mejora al usar los datos meteorológicos y de calendario.

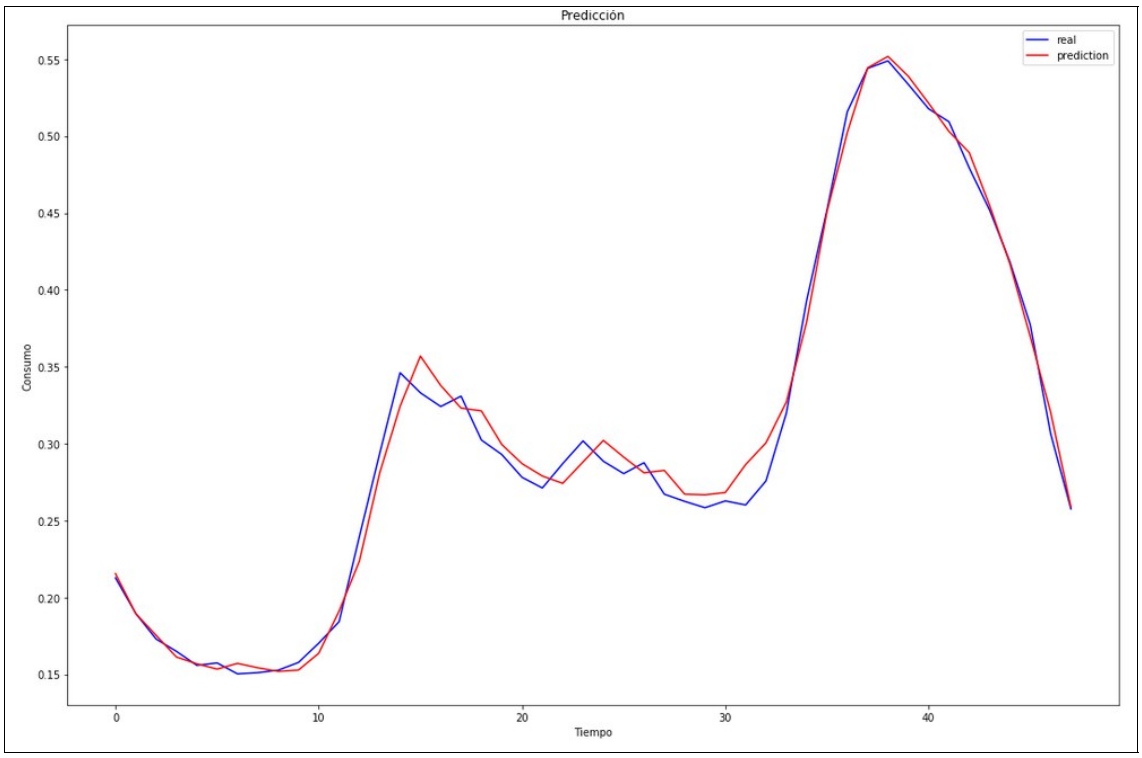


Figura 4.12. *Predicción de consumo en kWw/hh (en rojo) sobre la serie original (en azul) sobre 24 horas del modelo GRU multivariado*

4.3.3 Modelo CNN

El modelo CNN (*Convolutional Neural Network*) son una extensión de las FNN (*feedforward neural network*). Aunque son más conocidas por el buen pronóstico de clasificación de imágenes, también las podremos usar para el pronóstico de series temporales. En este tipo de redes cada nodo funciona como un filtro en forma de ventana que se desliza por los datos en busca de patrones.

El modelo escogido para nuestras pruebas tiene la siguiente configuración (figura 4.13). Se trata de un modelo formado por una capa con los siguientes elementos: convolución (*Conv1D*), *pooling* (*MaxPooling1D*) y reducción de la dimensión (*Flatten*), que recibe los datos de consumo eléctrico de un día (48 mediciones) y devuelve un valor que representará el siguiente valor futuro o lectura pronosticada por nuestra red neuronal.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 46, 128)           512
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 23, 128)           0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2944)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2945
=================================================================
Total params: 3,457
Trainable params: 3,457
Non-trainable params: 0
_________________________________________________________________
```

Figura 4.13. *Definición de nuestro modelo CNN.*

En la siguiente figura (figura 4.14) mostramos la tasa de error en entrenamiento y validación de nuestro modelo. Podemos observar como la tasa de error de entrenamiento disminuye rápidamente hacia cero pero la de validación queda de forma estacionaria. El hecho de que no tienda hacia cero nos puede indicar que al modelo le pueden faltar más épocas de entrenamiento (se han realizado 20 en este caso). Pero la tasa de error alcanzada ya es bastante baja.

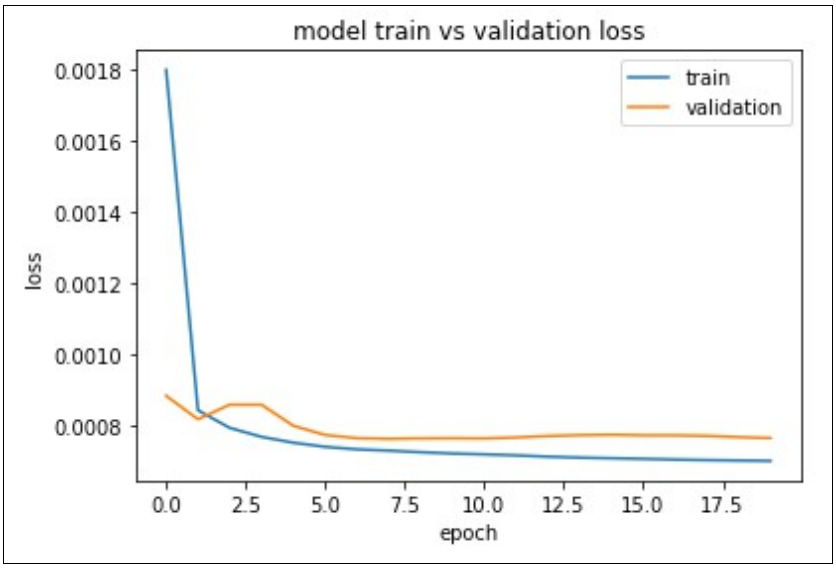


Figura 4.14. *Tasa de error de entrenamiento (azul) y validación (naranja) de nuestro modelo CNN (solo datos consumo)*

Si usamos los datos de meteorología y calendario de las que disponemos, la tasa de error de validación parece ser más baja y con mejor tendencia a cero (figura 4.15), así que seguramente tendremos un mejor pronóstico.

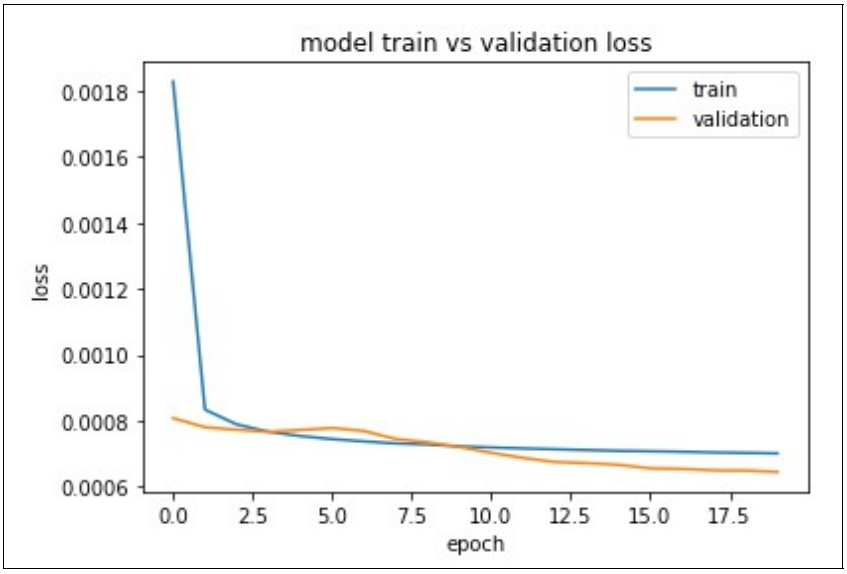


Figura 4.15. *Tasa de error de entrenamiento (azul) y validación (naranja) de nuestro modelo CNN*

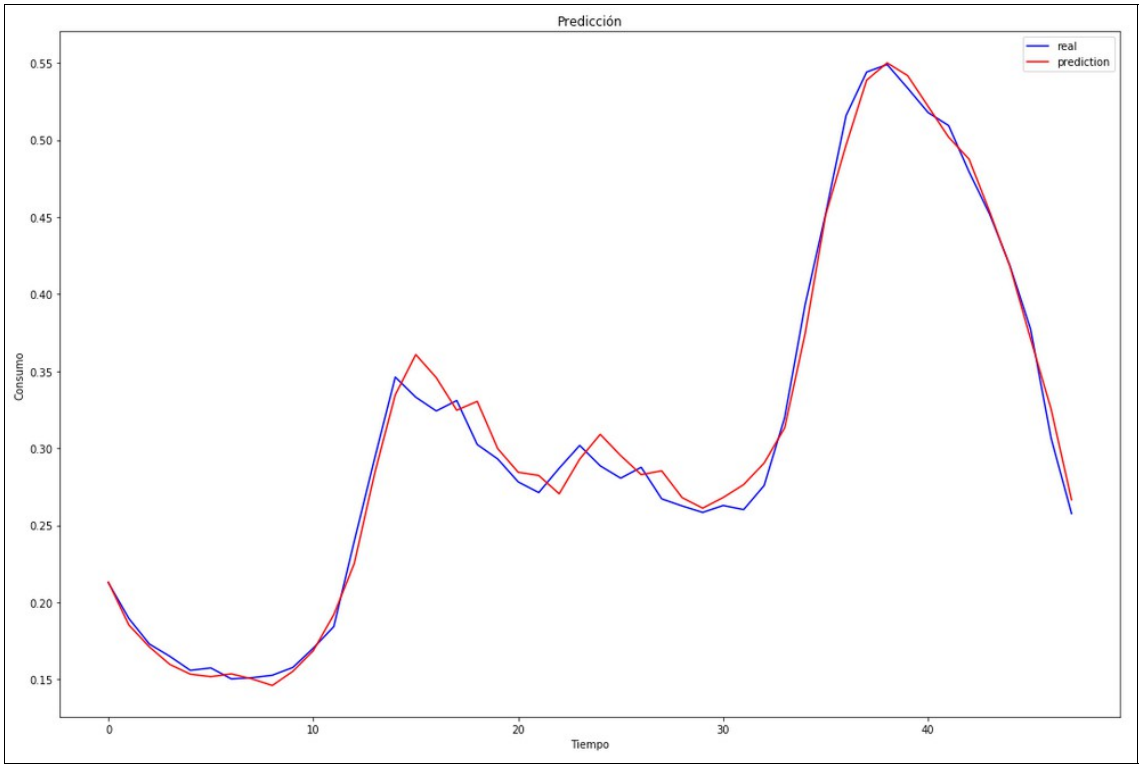


Figura 4.16. *Predicción de consumo en kWw/hh (en rojo) sobre la serie original (en azul) sobre 24 horas del modelo CNN multivariado*

Con el modelo CNN univariable el valor medio de RMSE ha sido de **0.01668** y de **0.01519** en el modelo multivariable, lo que representa un 9% de mejora al usar los datos meteorológicos y de calendario.

En la figura anterior (figura 4.16) se muestra un ejemplo gráfico de la serie temporal real (en azul) y la pronosticada por nuestro mejor modelo (en rojo) sobre los datos del conjunto de test sobre 1 día.

4.3.4 Modelo CNN-LSTM

Hasta ahora hemos visto como trabajan distintos modelos de predicción como el CNN o el LSTM, donde cada uno de ellos aporta a nuestra predicción una serie de capacidades distintas para un mismo fin. Es por eso que en ocasiones se desarrollan modelos híbridos, donde el modelo tiene la ventaja de tener más de una de estas capacidades y así obtener una mejor predicción.

En este caso desarrollaremos un modelo híbrido con una red neuronal del tipo CNN y LSTM. Aplicaremos en primer lugar una capa convolucional y de reducción (*pooling*). Seguidamente el resultado alimentará a una red neuronal LSTM a través de una capa de redimensionado (*flatten*). La definición se puede ver en la siguiente imagen (figura 4.17).

```
model = Sequential()
model.add(TimeDistributed(Conv1D(filters=128, kernel_size=3),
model.add(TimeDistributed(MaxPooling1D(pool_size=2)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed_1 (TimeDist (None, None, 49, 128)     512
_________________________________________________________________
time_distributed_2 (TimeDist (None, None, 24, 128)     0
_________________________________________________________________
time_distributed_3 (TimeDist (None, None, 3072)        0
_________________________________________________________________
lstm_1 (LSTM)                (None, 50)                624600
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 625,163
Trainable params: 625,163
Non-trainable params: 0
_________________________________________________________________
```

Figura 4.17. *Definición de nuestro modelo CNN-LSTM.*

Si entrenamos este modelo con la misma configuración que usamos para los modelos CNN y LSTM por separado, obtenemos las siguientes gráficas de tasas de error (figuras 4.18 y 4.19).

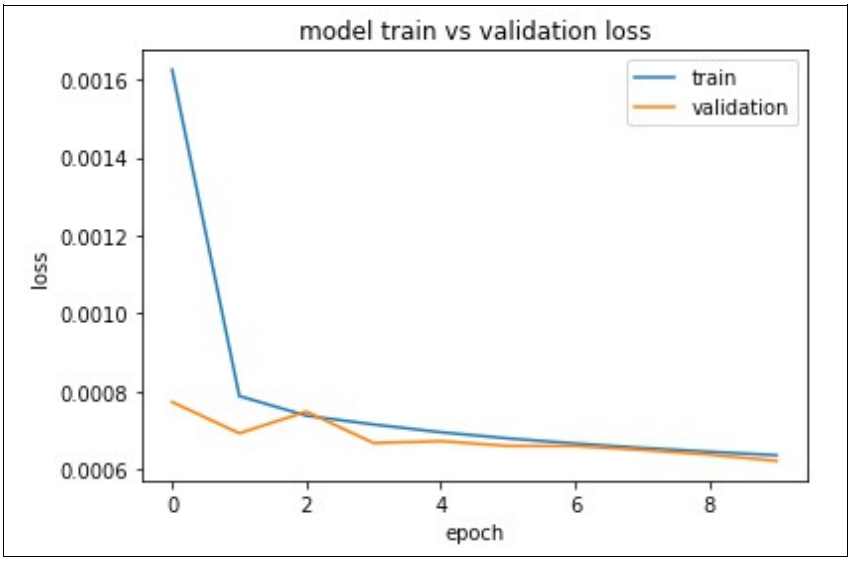


Figura 4.18. *Tasa de error de entrenamiento (azul) y validación (naranja) de nuestro modelo CNN-LSTM (solo datos consumo)*

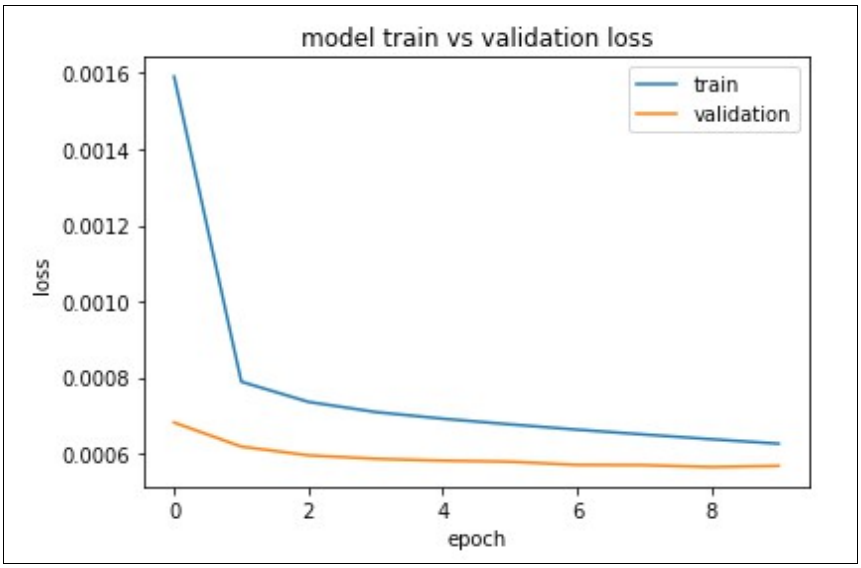


Figura 4.19. *Tasa de error de entrenamiento (azul) y validación (naranja) de nuestro modelo CNN-LSTM*

Podemos observar como la tasa de error de entrenamientos, tanto univariable como multivariable, tiende rápidamente a cero con pocas épocas. Pero en el caso del error de validación, cuando sólo usamos datos de consumo (univariable) vemos una pequeña fluctuación y estacionalidad en las primera épocas, pero aún así la tasa es muy baja y prevemos una buena predicción.

Con el modelo CNN-LSTM univariable el valor medio de RMSE ha sido de **0.01438** y de **0.01414** en el modelo multivariable, lo que representa un 2% de mejora al usar los datos meteorológicos y de calendario.

En la siguiente figura (figura 4.20) se muestra un ejemplo gráfico de la serie temporal real (en azul) y la pronosticada por nuestro modelo híbrido (en rojo) sobre los datos del conjunto de test sobre 1 día.

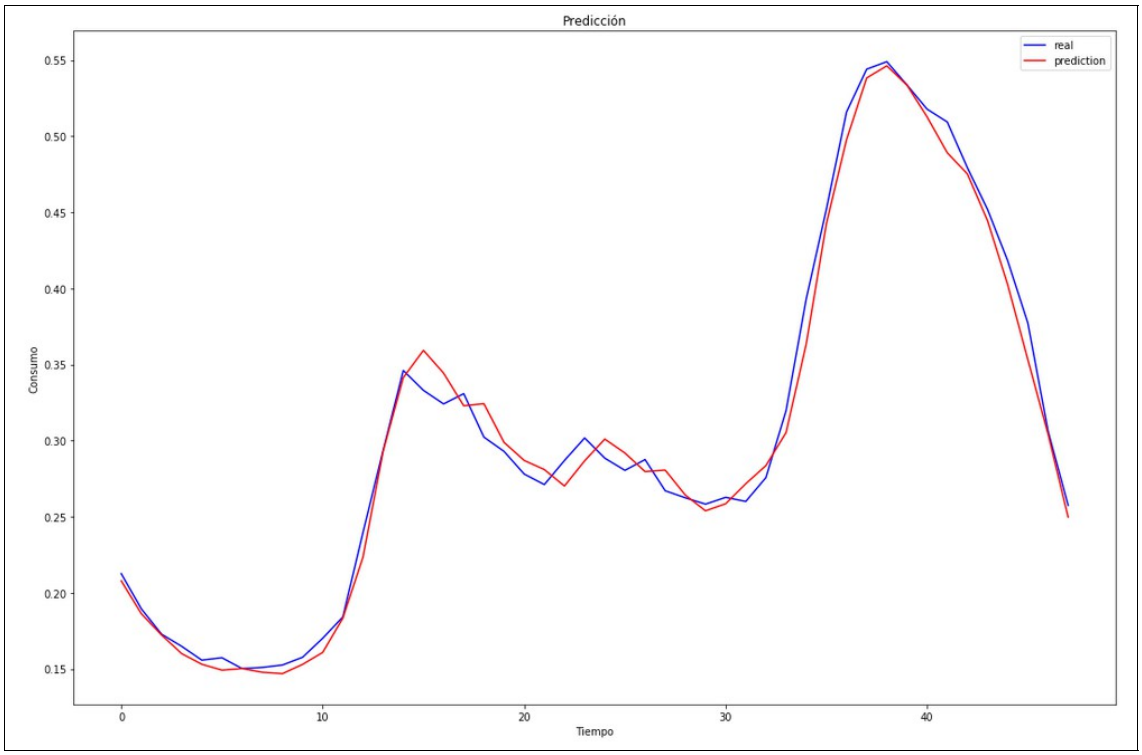


Figura 4.20. *Predicción de consumo en kWw/hh (en rojo) sobre la serie original (en azul) sobre 24 horas del modelo CNN-LSTM multivariables*

El código en *Python* para *el entrenamiento de los distintos modelos* se puede encontrar en:
https://github.com/cfiguerap/uoc-data-science-tfm/tree/master/src/1-arima-model
https://github.com/cfiguerap/uoc-data-science-tfm/tree/master/src/2-lstm-model
https://github.com/cfiguerap/uoc-data-science-tfm/tree/master/src/3-gru-model
https://github.com/cfiguerap/uoc-data-science-tfm/tree/master/src/4-cnn-model
https://github.com/cfiguerap/uoc-data-science-tfm/tree/master/src/5-cnn-lstm-model

5. Conclusiones del estudio

5.1 Comparación de los resultados

A continuación compararemos los diferentes resultados de los modelos entrenados en este proyecto. Para la comparación de los modelos se usará la raíz del error cuadrático medio (RMSE) obtenido al diferenciar la serie temporal real con la serie pronosticada por cada modelo.

Otro dato a comparar podría ser el tiempo de entrenamiento, pero en el caso de los modelos de *deep learning* se ha detectado que estos tiempos son muy parecidos (valores de entre 2-3 segundos por época) y no nos aportan mucho valor al estudio. En el caso del método ARIMA, como ya hemos comentado anteriormente, el tiempo total de entrenamiento sí que es muy superior, de la magnitud de minutos (~10 minutos).

Veamos primero (figuras 4.21 y 4.22) la comparación de modelos por su RMSE. Podemos ver que los modelos que mejor han pronosticado nuestro problema de predicción de consumos eléctricos han sido los LSTM, GRU y el híbrido CNN-LSTM. Pero si nos fijamos, si solo usamos datos de consumo, el modelo híbrido es el que obtiene la tasa de error más pequeña.

Así que podríamos decir que el modelo que mejor ha pronosticado el consumo eléctrico de nuestro problema, ha sido el híbrido CNN-LSTM. Pero hay que tener en cuenta que la diferencia de error obtenida ha sido similar entre los distintos modelos.

En el caso de ARIMA no tenemos datos sobre el modelo multivarable ya que no se ha implementado.

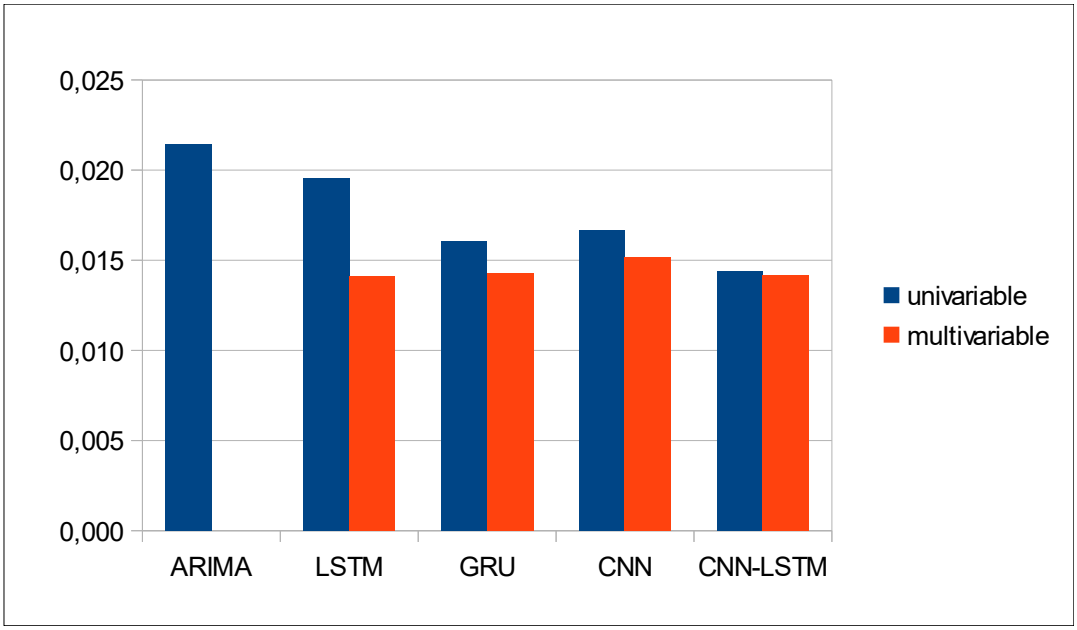


Figura 4.21. *Raíz del error cuadrático medio (RMSE) de cada modelo implementado. Comparación sobre datos univariables y multivariables*

	univariable	multivariable
ARIMA	0,02146	
LSTM	0,01956	0,01412
GRU	0,01608	0,01427
CNN	0,01668	0,01519
CNN-LSTM	0,01438	0,01414

Figura 4.22. *Raíz del error cuadrático medio (RMSE) de cada modelo implementado. Comparación sobre datos univariables y multivariables*

Si nos fijamos podremos ver como los modelos han realizado mejores predicciones cuando se han usado los datos de meteorología y de fías festivos. La mayor mejora ha sido en el caso del modelo LSTM. En la figura 4.23 podemos comparar como los casos de entrenamiento multivariable generalmente han funcionado mejor.

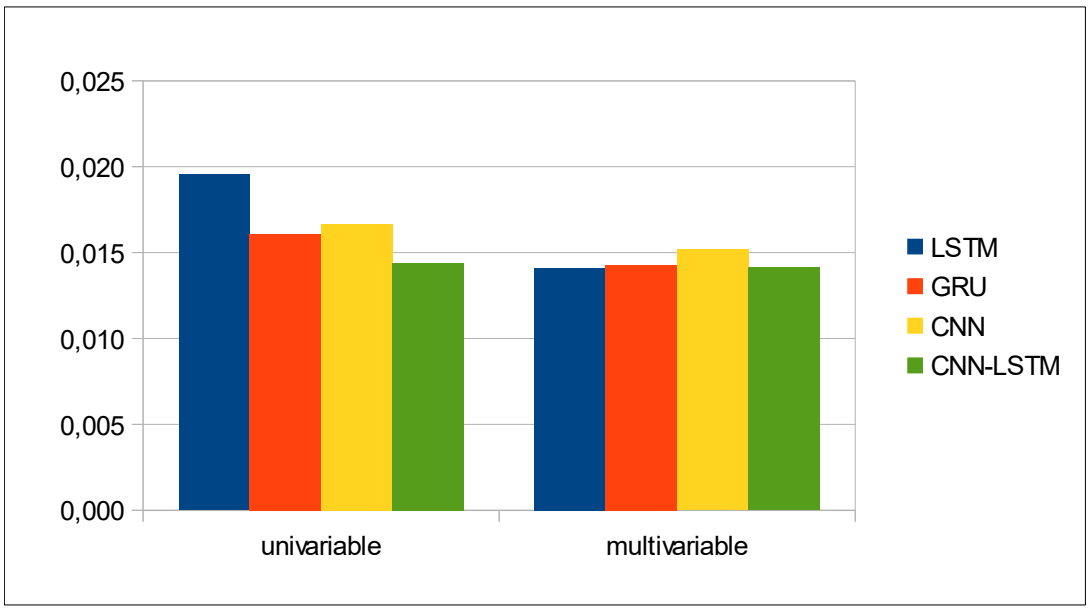


Figura 4.23. *Raíz del error cuadrático medio (RMSE) de modelos deep learning. Comparación sobre datos univariables y multivariables*

5.2 Técnicas tradicionales y deep learning

En el apartado 2.4 ya se describen las principales diferencias entre estos dos tipos de modelos, pero se van a comentar características encontradas después de implementar los distintos modelos.

Durante el entrenamiento de los modelos se ha podido observar que una de las grandes diferencias entre los modelos estadísticos o tradicionales como ARIMA y los modelos de *deep learning* ha sido que el modelo estadístico necesita de un trabajo

previo donde se trata de encontrar los parámetros P, D y Q que mejor se ajustan a un problema de predicción.

En cambio, los modelos de redes neuronales no necesitan de esta parametrización previa. Es cierto que también necesitan de una configuración de entrenamiento (número de neuronas, número de épocas, tamaño de *batch*, etc) pero se trata de una configuración mínima.

Este hecho hace indicar que el modelo ARIMA obtenido para una serie temporal, puede no ser de utilidad para otra serie temporal distinta, a menos que esta sea similar.

Es decir, si implementamos un modelo estadístico dada una serie temporal, dicho modelo seguramente pronosticará valores futuros erróneos para una serie temporal de distintas características (tendencia, estacionalidad, ruido,...). Este hecho puede promover la idea de que los modelos tradicionales no están tan bien preparados para series temporales con mucho ruido

Pero los modelos estadístico también tienen sus beneficios, al requerir de esta parametrización previa no necesitan de tantos datos para el entrenamiento. En nuestro problema concreto de consumo eléctrico no ha sido así, quizás porque disponíamos de un amplio *dataset* que ha favorecido en nuestra comparación a los modelos de *deep learning*, pero con otros tipos de problemas pueden tener mejor rendimiento.

También es cierto que los modelos de red neuronal son más complejos y tediosos de trabajar, ya que se requiere de un conocimiento previo de su funcionamiento y distintas arquitecturas: multicapas, *overfitting*, retropropagación,... Mientas que los modelos estadísticos son más fáciles de implementar para profesionales sin conocimientos en *machine learning*.

5.3 Análisis de la planificación

A modo resumen del trabajo, se han completado casi todos los objetivos definidos al inicio del proyecto. Ha quedado sin poder implementar el estudio de mejoras para modelos de *deep learning*, como puede ser *dropout*, *early stopping*, etc. Este objetivo quedará para futuras líneas de investigación. Pero a cambio, se ha podido implementar un modelo de predicción híbrido, tarea que no aparecía en la planificación de trabajo inicial.

En cuanto a fechas de entrega, la planificación inicial se ha podido seguir con éxito, con alguna pequeña desviación que se ha podido rectificar a tiempo. Se ha seguido una metodología en cascada para los entregables del proyecto pero también se ha usado una metodología más *agile* para la creación de los modelos predictivos. Es decir, los modelos han sido implementados de una forma iterativa a partir de un *dataset* más pequeño para agilizar así su implementación pudiendo reducir costes, por ejemplo, en tiempos de entrenamiento. Una vez se han encontrado las configuraciones idóneas de cada modelo, se ha procedido a crear los modelos con el *dataset* del proyecto.

5.4 Futuras líneas de investigación

Existen distintas líneas de investigación a seguir tras el trabajo realizado en este proyecto. Las más destacadas se comentan a continuación:

- ***Comparativa según tamaños de dataset***

  Se puede añadir una comparación entre los modelos estudiados en el proyecto partiendo del tamaño del conjunto de datos de entrenamiento. Así podríamos concluir cuál de los distintos modelos tiene una mejor predicción según el tamaño de datos de conocimiento con el que se entrena.

- ***Técnicas para evitar overfitting***

  Se puede contribuir a la comparación de modelos de *deep learning* con técnicas que pueden ayudar a la reducción del *overfitting* como pueden ser el *Dropout* o *Early Stopping* [24]. La primera consiste en desactivar neuronas de las capas ocultas de forma aleatorio ayudando al modelo a generalizar los datos de entrenamiento. El segundo método consiste en parar el entrenamiento basándonos en ciertas reglas para así evitar que nuestro modelo sobreentrene los datos de entrenamiento.

- ***Estudio de otros modelos híbridos***

  Se pueden implementar sobre nuestro *dataset* otros tipos de modelos híbridos, que además, no hace falta que sean sólo del tipo *deep learning. S*e puede implementar un modelo híbrido con técnicas estadísticas y de *deep learning,* como por ejemplo un modelo ARIMA-LSTM [25].

- ***Entrenamiento distribuido***

  A modo personal, me hubiera gustado poder entrenar los distintos modelos usando herramientas para entrenamiento distribuido. Por ejemplo, el uso de la *GPU* en la fase de entrenamiento gracias a la tecnología *CUDA* de *NVIDIA* [26], la cuál nos permite paralelizar las tareas de cómputo en distintos hilos que se ejecutan en la unidad gráfica, permitiendo así una reducción importante en el coste de entrenamiento en términos de tiempo.

6. Bibliografía

[1] Notícia: "*Almacenamiento de energía eléctrica a gran escala*" (27-feb-2020)
http://www.agenex.net/es/documentacion/381-almacenamiento-de-energia-electrica-a-gran-escala.html

[2] Notícia: "*Empresas españolas logran almacenar a gran escala electricidad de fuentes renovables*" (27-feb-2020)
https://www.energias-renovables.com/panorama/empresas-espanolas-logran-almacenar-a-gran-escala-20140605

[3] Web: "*Demanda de energía eléctrica en tiempo real*" (27-feb-2020)
https://demanda.ree.es/visiona/peninsula/demanda/total

[4] Web: "*Previsión meteorológica y clima Madrid*" (27-feb-2020)
https://www.weather-es.com/es/espana/madrid-el-tiempo-en-enero

[5] Notícia: "*Smartcity Málaga, 10 años después*" (29-feb-2020)
https://www.lavanguardia.com/natural/si-existe/20200128/473098862713/malaga-smart-city-endesa-laboratorio-living-lab-diez-anos-brl.html

[6] Libro: A. Palit, D. Popovic. "*Computational Intelligence in Time Series Forecasting*" (2005). (págs: 17-24)

[7] Web: "*Introduction to stationarity*"
https://www.analyticsvidhya.com/blog/2018/09/non-stationary-time-series-python

[8] Web: "*Decomposition of time series*" (wikipedia)
https://en.wikipedia.org/wiki/Decomposition_of_time_series

[9] Libro: R.J Hyndman, G. Athanasopoulos. "*Forecasting: Principles and Practice*" (2008). Capítulo 6: "Time series decomposition"

[10] Libro: Chrios Chatfield. "*Time-Series Forecasting*" (2000).
Capítulo 6: "A Comparative Assessment of Forecasting Methods"

[11] Web: "*Understanding deep learning models for time series forecasting and their advantages over traditional models like ARIMA*" (20-mar-2020)
https://towardsdatascience.com/deep-learning-for-time-series-and-why-deep-learning-a6120b147d60

[12] Web: "*How to Create an ARIMA Model for Time Series Forecasting*" (20-mar-2020)
https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

[13] Libro: J. Gironés, J. Casas, J. Minguillón, R. Caihuelas. "Minería de datos: modelos y algoritmos" (UOC, 2017). Capítulo 12: "*Redes neuronales*"

[14] Web: "*Redes neuronales ¿qué son?*" (21-mar-2020)
https://medium.com/@williamkhepri/redes-neuronales-que-son-a64d022298e0

[15] Web: "*Convolutional Neural Network Tutorial: From Basic to Advanced*" (21-mar-2020)
https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/

[16] Vídeo: J. Ordóñez. "*Deep Learning para el análisis de series temporales*" (21-mar-2020)
https://www.youtube.com/watch?v=7gsIkXpZx9E

[17] Web: "*Gated recurrent unit*"
https://en.wikipedia.org/wiki/Gated_recurrent_unit

[18] Notícia: "*Endesa alcanza los 10 millones de contadores inteligentes*" (10-may-2020)
https://www.endesa.com/es/prensa/sala-de-prensa/noticias/transicion-energetica/redes-inteligentes/Endesa-alcanza-los-10-millones-de-contadores-inteligentes-instalados

[19] Notícia: "*Enel: Italy reaping first-mover benefits of smart meters*" (10-may-2020)
https://www.euractiv.com/section/climate-environment/interview/enel-italy-reaping-first-mover-benefits-of-smart-meters

[20] Web: "*Energy in the UK — Analysis of the energy performance certificates*" (11-may-2020)
https://towardsdatascience.com/energy-in-the-uk-analysis-of-the-energy-performance-certificates-585665721b9c

[21] Web: "*R (lenguaje de programación)*" (13-may-2020)
https://es.wikipedia.org/wiki/R

[22] Notícia: "*Tormenta huracanada deja 13 muertos en Europa*" (16-may-2020)
https://www.bbc.com/mundo/ultimas_noticias/2013/10/131028_ultnot_acerca_tormenta_reino_unido

[23] Notícia: "*Una fuerte tormenta que golpea el norte europeo deja 11 muertos*" (16-may-2020)
https://www.lavozdegalicia.es/noticia/internacional/2013/10/29/fuerte-tormenta-golpea-norte-europeo-deja-11-muertos/0003_201310G29P28991.htm

[24] Web: "*Técnicas de Regularización Básicas para Redes Neuronales*" (12-junio-2020)
https://medium.com/metadatos/t%C3%A9cnicas-de-regularizaci%C3%B3n-b%C3%A1sicas-para-redes-neuronales-b48f396924d4

[25] Artículo: Hyeong Kyu Choi. "*Stock Price Correlation CoefficientPrediction with ARIMA-LSTMHybrid Model* "
https://arxiv.org/pdf/1808.01560.pdf

[26] Web: "*CUDA: Compute Unified Device Architecture*" (12-junio-2020)
https://es.wikipedia.org/wiki/CUDA