Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Estadística

Creación de un modelo predictivo de consumo energético de un edificio inteligente

Autor: Javier Martínez Sánchez

Tutor: J. Belarmino Pulido Junquera

Creación de un modelo predictivo de consumo energético de un edificio inteligente

Resumen

El 40% del consumo energético de la UE y del 36% de las emisiones de gases de efecto invernadero son consecuencia de edificios. Esto manifiesta la necesidad de mejora en la eficiencia energética de las distintas edificaciones para proveer un desarrollo sostenible.

En este contexto surgen los denominados Smart Buildings, edificios optimizados para reducir su consumo energético hasta valores cercanos a cero. Estos edificios están equipados con sistemas que permiten medir su rendimiento en distintas secciones.

Las diversas técnicas de análisis de datos con las que contamos en la actualidad nos permiten desarrollar distintos modelos predictivos para variables que nos resulten de interés. El consumo energético es también susceptible de ser predicho mediante distintos modelos de regresión, "machine learning" y series temporales.

En este trabajo se elaborará un conjunto de modelos de distinta índole para predecir el consumo energético del edificio LUCIA utilizando distintas variables que aporten información a la hora de explicar la variabilidad en dicho consumo.

Abstract

Buildings account for 40% of the EU's energy consumption and 36% of its greenhouse gas emissions. This highlights the necessity to improve the energy efficiency of buildings to ensure sustainable development.

This has led to the emergence of smart buildings, buildings optimized to reduce their energy consumption to near-zero levels. These buildings are equipped with systems that make it possible to measure their performance in different areas.

The diverse data analysis techniques that we have nowadays allow us to develop several predictive models for the most interesting variables. Energy consumption is also susceptible of being predicted by distinct regression, machine learning, and time series models.

In this project, different kind of models will be used to predict energy use of LUCIA building using several variables that provide information when explaining variability in this consumption.

Agradecimientos

A mi tutor, Belarmino, por su guía y ayuda en el desarrollo de este trabajo.

A mi familia, por estar a mi lado siempre que lo he necesitado.

Contenido

Tabla de ilustraciones

1 Introducción

1.1 Motivación

El análisis de los datos energéticos permite la descomposición del consumo y sus costes asociados por departamentos dentro de una organización. En las auditorías energéticas se realiza una inspección y análisis de los patrones de consumo eléctrico de un edificio documentando los flujos energéticos de entrada y salida. Las auditorías energéticas pueden mejorar la eficiencia general de un edificio, al reducir el coste de la electricidad y las emisiones de gases de efecto invernadero. No obstante, las inspecciones físicas requieren personas y equipos especializados, lo que aumenta el coste y genera dudas sobre la rentabilidad de esta. Según el departamento estadounidense de energía las estimaciones indican que la eficiencia se puede haber aumentado un 30% en 2030 implementando las tecnologías existentes. La herramienta óptima para identificar las oportunidades de mejora de la eficiencia es, sin embargo, el análisis de datos (1).

1.2 Relación con asignaturas del grado

El estudio que se realizará a continuación está fuertemente ligado a las técnicas de análisis de datos estudiadas en el grado de estadística, principalmente en las asignaturas de *análisis de series temporales, técnicas de aprendizaje automático y análisis de datos.* Para este proyecto también se requerirá de métodos de manejo de software estadístico aprendidos en *Computación estadística.* También se emplearán otros algoritmos similares no estudiados en dichas asignaturas.

1.3 Edificios inteligentes

Los edificios inteligentes son aquellos que buscan proporcionar sistemas de iluminación, climatización y ventilación de forma que las emisiones de CO2 sean cercanas a 0 (NZEB-Nearly Zero Energy Building) (2). Un ejemplo de edificio inteligente es el edificio LUCIA situado en el campus Miguel Delibes de la Universidad de Valladolid.

1.3.1. LUCIA

El edificio LUCIA (Lanzadera Universitaria de Centros de Investigación Aplicada) ha obtenido certificaciones que le avalan como una de las edificaciones más sostenibles del mundo. El edificio optimiza el uso de luz natural y utiliza lucernarios de vidrio fotovoltaico, cuenta con grandes niveles de aislamiento térmico y un sistema de ventilación eficiente a través de chimeneas solares acompañado de un sistema de energía geotérmica que acondiciona el aire exterior obtenido para la ventilación (3).

Ilustración 1.1 Edificio LUCIA tomado de: (3)

1.4 Objetivos y alcance

Este trabajo consiste en el análisis de los distintos datos disponibles para un edificio inteligente y su relación con los consumos eléctricos o energéticos del edificio. En concreto, en este trabajo se elaborará un modelo predictivo para la sección de refrigeración del edificio. Para que esta última tarea resulte útil, se debe tener en cuenta que los parámetros de los sistemas del edificio son ajustados cada semana, lo que implica realizar predicciones a una semana.

El principal objetivo de este TFG es comprobar la viabilidad de un modelo predictivo para el consumo de la sección de refrigeración del edificio LUCIA y al mismo tiempo comprobar la influencia de los modos de operación del edificio asociado a distintas condiciones meteorológicas. Las tareas asociadas a este objetivo son el análisis y limpieza de los datos, así como el estudio y evaluación de distintos tipos de modelos predictivos: estadísticos o basados en aprendizaje automático, tanto para todos los datos como para los datos asociados a distintos clusters encontrados.

El edificio LUCIA dispone de dos sistemas de producción de frío, el primero es una enfriadora eléctrica "convencional", y el segundo un sistema de refrigeración a través de una máquina de absorción. El primero, el de la enfriadora eléctrica, consume energía eléctrica para enfriar el agua utilizada en el sistema de distribución de frío del edificio. En cambio, el segundo (la absorción), emplea agua caliente y electricidad para su ciclo frigorífico (electricidad en la propia enfriadora por absorción y en una torre de refrigeración necesaria para disipar el calor producido en el ciclo).

1.5 Estado del arte

A diferencia de los modelos físicos, los modelos basados en datos no requieren información muy detallada sobre el edificio, ya que se basan en históricos de predicción. El creciente interés por estos modelos ha supuesto la publicación de varios artículos con distintos

enfoques. Sin embargo, pocos estudios analizan este problema desde una perspectiva multivariante que incluyan más características a analizar (4).

La ASHRAE (Sociedad Estadounidense de Calefacción, Refrigeración y Aire Acondicionado) ha elaborado una guía (5) sobre los procedimientos a seguir y las métricas adecuadas en la medición de las demandas energéticas de un edificio. Esta asociación recomienda el uso del coeficiente de variación de la raíz del error cuadrático medio (CV-RMSE), el cual no debe ser mayor que 0.3 en medidas del consumo total del edificio.

Existen algunos trabajos realizados sobre el consumo energético de un edificio inteligente como el trabajo de fin de grado de Alejandro Barón (6), o el artículo (7) en los que se analiza el consumo del edificio Alice Perry en Irlanda. Por otro lado, también se ha realizado un análisis del comportamiento del consumo de refrigeración en el edificio LUCIA en el trabajo de fin de grado de Cristina García (8).

1.6 Sistema de gestión de datos

El sistema utilizado para la recogida de datos relativos a los distintos consumos del edificio LUCIA y otros edificios de la Universidad de Valladolid se llama DESIGO y ha sido diseñado por SIEMENS©. Los distintos analizadores situados en cada una de las plantas del edificio registran los valores obtenidos en este sistema con distintas periodicidades, lo que implica tener que definir períodos concretos y agrupar las variables si se desea compararlas. Cada registro incluye tanto el valor como la fecha y hora de cada medición. Es posible acceder a los datos de este sistema mediante una máquina virtual donde se extraen los datos de cada analizador desde archivos por meses.

1.7 Clasificación del problema y métodos de predicción

La naturaleza del problema de predicción de la carga energética es fundamentalmente la predicción de series temporales. Podemos agrupar las técnicas para este tipo de análisis en dos grandes grupos:

- **Análisis estadístico**: Estos métodos requieren que los residuos sean normales e independientes.
    - **ARIMA:** En este tipo de modelos las predicciones se realizan mediante una combinación lineal de algunas variables anteriores y errores aleatorios independientes. Estos modelos son útiles si las observaciones son correladas. Los modelos SARIMA añaden estacionalidad a los modelos ARIMA.
- **Técnicas de Machine Learning:** Estas técnicas no hacen ninguna suposición sobre la distribución subyacente a los datos, esto mejora la capacidad predictiva de estos modelos a costa de la facilidad de interpretación de los modelos estadísticos. En particular utilizaremos modelos de Deep Learning entre los que destacan:

- **Perceptrones:** Estas redes neuronales contienen una o varias capas de neuronas.
- **Redes Recurrentes (RNNs):** En estas redes existen conexiones de una neurona consigo misma de forma directa o indirecta, lo cual, nos permite modelar de forma explícita relaciones temporales, estableciendo relaciones entre valores en el instante actual y valores en instantes pasados.

1.8 Planificación

Para realizar una breve planificación del proyecto se tendrán en cuenta las distintas tareas que se deben llevar a cabo, así como su coste. Se valorarán los riesgos que pueden afectar al correcto desarrollo del proyecto y se analizarán los recursos necesarios para el mismo.

1.8.1. Tareas

Tarea	Subtarea	Estimación	Duración Total
Objetivos y alcance	Reuniones	3h	15h
	Estudio de métodos	7h	
	Planificación	5h	
Entendimiento de los datos	Lectura de la documentación de los datos	5h	20h
	Exploración de los datos	15h	
Tratamiento de datos	Selección de variables	10h	20h
	Limpieza del conjunto de datos	5h	
	Transformaciones del conjunto de datos	5h	
Modelado	Métodos estadísticos	30h	60h
	Métodos de aprendizaje automático	30h	
Evaluación	Evaluación final de modelos	5h	5h
Redacción de la memoria	Redacción de la memoria	30h	30h

1.8.2. Riesgos

Identificación de riesgos que pueden influir en el desarrollo del proyecto:

Riesgo	Probabilidad	Impacto
Conceptos mal aprendidos	Muy baja	Alto
Objetivos no comprendidos	Media	Alto
Datos inadecuados para el problema	Baja	Crítico
Datos poco documentados	Baja	Bajo
Datos demasiado complejos	Alta	Medio
Inexistencia de modelos adecuados	Muy baja	Crítico
Problemas en la elaboración del modelo	Media	Bajo
Evaluación incorrecta	Baja	Alto

1.8.3. Recursos

- Ordenador personal
- Cisco Webex Meetings ©
- R
    - Librería Tidyverse(dplyr, stringr, ggplot2)
    - UnivOutl
    - Tensorflow, Keras
    - RandomForest
- SAS ©
- Microsoft Teams ©
- Microsoft Office ©

2 Fundamento teórico

2.1 Modelos estadísticos

2.1.1. Modelos ARIMA

Este tipo de modelos son útiles para representar la dependencia de los valores de una serie temporal de su pasado (9). Los tipos de modelos son:

Modelos AR

Los modelos AR o autorregresivos generalizan la idea de regresión representando los valores anteriores de la serie como regresores. Estos son los modelos más simples y los primeros en estudiarse. Se define el orden de este tipo de modelos como el número de retardos anteriores al valor a predecir utilizados en el modelo (10). Ejemplos de este tipo de modelos son:

AR(1)

Este tipo de modelos toman la observación anterior para cada predicción. Se considera a una serie de este modelo si es generada por:

$$z_t = c + \Phi z_{t-1} + a_t \qquad (1)$$

Donde c y Φ son constantes no estimadas mientras que $\mathbf{a_t}$ es un proceso de ruido blanco con varianza σ^2 denominado innovación (10). c es una constante para toda la serie mientras que Φ es el coeficiente del valor anterior. Si $-1 < \Phi < 1$ el proceso es estacionario.

AR(p)

Es la generalización de los modelos autorregresivos de cualquier orden. Su fórmula es la siguiente:

$$z_t = c + \sum_{i=1}^{p} (\Phi_i z_{t-i}) + a_t \qquad (2)$$

En este caso usaremos p retardos para modelizar la serie. Podemos ver que las constantes son análogas a las de los procesos AR(1).

Estos modelos no tienen en cuenta el error aleatorio de las observaciones anteriores. Para modelizarlos necesitaremos los modelos MA (moving average).

Modelos MA

Los modelos MA o de media móvil son aquellos que utilizan el error aleatorio de observaciones anteriores para predecir los siguientes valores. Estos modelos son siempre estacionarios. Estos modelos nos permiten representar procesos de memoria corta mejor que los AR (10). De forma análoga podemos definir los siguientes ejemplos:

MA(1)

Al igual que en los AR(1), solo se utiliza un retardo para representar la serie, pero en este caso utilizaremos la innovación de la observación anterior. Su fórmula es:

$$z_t = c + \theta a_{t-1} + a_t \tag{3}$$

Al igual que en los modelos vistos anteriormente c es una constante para toda la serie y a_t es un proceso de ruido blanco con varianza σ^2. La diferencia es que se utilizará como regresor el valor anterior de a_t en lugar de la observación. En estos modelos los coeficientes se denotan como θ.

MA(q)

De esta forma se generalizan los modelos MA para añadir más retardos.

$$z_t = c + \sum_{i=1}^{q} (\theta_i a_{t-i}) + a_t \tag{4}$$

Modelos ARMA

Estos modelos son una combinación de los dos anteriores para intentar combinar las propiedades de ambos y ser representados con pocos parámetros (10). Para que una serie pueda ser modelizada mediante un modelo ARMA sigue siendo necesario que sea estacionaria. Estos modelos intentan predecir el valor siguiente a partir de las observaciones e innovaciones anteriores. Algunos ejemplos de este modelo son:

ARMA(1,1)

Estos modelos son una composición de un modelo AR(1) y un MA(1). Utiliza como regresores la observación anterior y su innovación Su fórmula es:

$$z_t = c + \Phi z_{(t-1)} + \theta a_{(t-1)} + a_t \tag{5}$$

ARMA(2,1)

Estos modelos son una composición de un modelo AR(2) y un MA(1). Utiliza como regresores las dos observaciones anteriores y la última innovación Su fórmula es:

$$z_t = c + \Phi_1 z_{t-1} + \Phi_2 z_{t-2} + \theta a_{t-1} + a_t \tag{6}$$

ARMA(1,2)

Estos modelos son una composición de un modelo AR(1) y un MA(2). Su fórmula es:

$$z_t = c + \theta_1 a_{t-1} + \theta_2 a_{t-2} + a_t \tag{7}$$

ARMA(p,q)

La generalización de los modelos anteriores nos permite seleccionar tantas observaciones denotadas por p e innovaciones denotadas por q como deseemos mediante la siguiente expresión:

$$z_t = c + \sum_{i=1}^{p} (\Phi_i z_{t-i}) + \sum_{i=1}^{q} (\theta_i a_{t-i}) + a_t \tag{8}$$

Estos modelos son estacionarios siempre que la parte autorregresiva lo sea también.

Modelos ARIMA

La mayoría de las series temporales no son estacionarias y los modelos estudiados hasta ahora son, en su mayoría, para procesos estacionarios. Muchas de estas series pueden convertirse en estacionarias al aplicar sobre ellas una diferenciación. De esta forma podremos modelar algunas series no estacionarias con los modelos vistos anteriormente. Algunos ejemplos de modelos ARIMA son:

Modelo ARIMA(p,1,q)

En este modelo se diferencia una sola vez la serie. Es útil para eliminar una tendencia lineal. Tras diferenciar se ajusta un modelo ARMA(p,q). Su expresión es la siguiente:

$$\Delta^1 z_t = z_t - z_{t-1} = c + \sum_{i=1}^{p} (\Phi_i (z_{t-i} - z_{t-1-i})) + \sum_{i=1}^{q} (\theta_i a_{t-i}) + a_t \tag{9}$$

En este modelo se elimina una observación que se utiliza para diferenciar la serie.

Modelo ARIMA(p,i,q)

Es la generalización de los anteriores su fórmula es:

$$\Delta^i z_t = c + \sum_{j=1}^{p} (\Phi_j (z_{t-j} - z_{t-i-j}) + \sum_{j=1}^{q} (\theta_j a_{t-j}) + a_t \quad (10)$$

Modelo SARIMA(p,i,q) $(P,I,Q)_s$

Para las series que presentan variaciones estacionales debidas a causas ajenas a la propia serie podemos utilizar un modelo SARIMA (Seasonal Autoregressive Integrated Moving Average). Estos modelos añaden, en una serie de período s, como regresores las observaciones anteriores de la misma estación. Con estos modelos se trata de recoger, además de la dependencia entre observaciones de la misma estación, relaciones de dependencia entre observaciones consecutivas (9). También añaden la posibilidad de diferenciar estacionalmente la serie, es decir restar observaciones de la misma estación anteriores a la serie ya diferenciada regularmente.

$$\begin{aligned} \Delta^I \Delta^i z_t = c + \sum_{j=1}^{p} (\Phi_i \Delta^I \Delta^i z_{t-i}) + \sum_{k=1}^{P} (\Phi_k \Delta^I \Delta^i z_{t-ks}) \\ + \sum_{k=1}^{P} \sum_{i=1}^{p} (\Phi_k \Phi_i \Delta^I \Delta^i z_{t-ks-i}) + \sum_{i=1}^{q} (\theta_i a_{t-i}) + \sum_{k=1}^{Q} (\Theta_k a_{t-ks}) \\ + \sum_{k=1}^{Q} \sum_{i=1}^{q} (\Theta_k \theta_i a_{t-ks-i}) + a_t \end{aligned} \quad (11)$$

Donde J es el orden de diferenciación estacional, P el orden de la parte autorregresiva estacional. Φ_k es el regresor de la parte autorregresiva estacional de orden k, Q el orden de la parte de media móvil estacional. Θ_k es el regresor de la parte de media móvil estacional y s indica la periodicidad de la serie.

2.2 Modelos de inteligencia artificial

2.2.1. Introducción

Análisis de series temporales con modelos de inteligencia artificial

La inteligencia artificial (IA) puede definirse como la rama de la informática que se ocupa de la automatización del comportamiento inteligente (11). La IA pretende construir sistemas y máquinas capaces de aprender y adaptarse a entornos cambiantes (12). La

inteligencia artificial es muy interdisciplinar abarcando ciencias como la neurociencia, la psicología, tecnologías de la información, la física, las matemáticas y la lógica (13). Podemos utilizar la inteligencia artificial para realizar predicciones a través de un conjunto de técnicas de aprendizaje automático.

El Aprendizaje Automático es una rama de la Inteligencia Artificial que abarca diferentes técnicas, las cuales permiten dotar a los computadores de la capacidad de "aprender" modelos tales que, de forma automática, pueden ser usados, por un lado, para resolver problemas nuevos o, por otro lado, para mejorar el rendimiento en problemas ya vistos (14). Existen distintos tipos de aprendizaje automático. Las tres principales ramas son:

- Aprendizaje Supervisado: Si depende de datos previamente etiquetados
- Aprendizaje no Supervisado: Si la máquina no cuenta con ninguna indicación previa
- Aprendizaje por Refuerzo: En este caso, la máquina aprende a través de experiencias en las que obtendrá información positiva o negativa sobre su propia respuesta.

Las redes neuronales artificiales son sistemas cuyo primer nivel jerárquico consiste en un aprendizaje sencillo que se transmitirá a la siguiente capa que tomará la información obtenida para extraer más información. Este proceso se repite en todas las capas de la red. Estas redes tratan de imitar redes neuronales biológicas mediante sistemas computacionales. Para su funcionamiento utilizan neuronas artificiales que son análogas a las del sistema nervioso animal. La transmisión de la información entre capas es la analogía computacional de la sinapsis en un cerebro biológico.

Una de las formas de aprendizaje automático más utilizadas hoy en día es el Aprendizaje Profundo. El Aprendizaje Profundo, conocido como Deep Learning en inglés, es un conjunto de algoritmos de aprendizaje automático con abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos. La mayoría de los enfoques de Deep Learning están asociados a las redes neuronales artificiales (15).

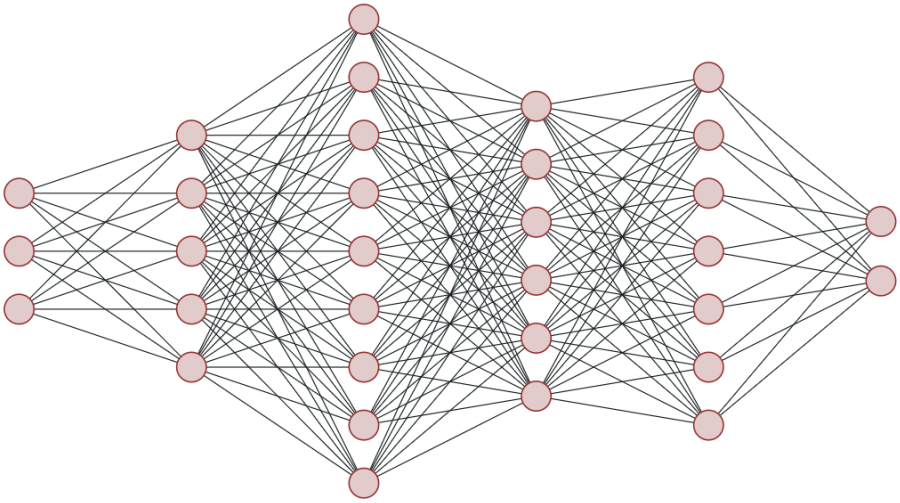


Ilustración 2.1 Ejemplo de red neuronal

El primer modelo de red neuronal fue realizado por Warren McCulloc y Walter Pitts en 1943 y fue utilizada en una máquina computacional por primera vez en 1954 por Wesley A.Clark (16).

Las redes neuronales se pueden utilizar para el estimar valores (regresión), para valores categóricos (clasificación) y se pueden extender para predecir series temporales.

Neuronas

Las neuronas son las unidades básicas de una red neuronal artificial. Son funciones matemáticas que toman unos datos de entrada, generalmente ponderados, para producir una activación mediante una función no lineal llamada función de activación.

El resultado de una neurona se puede expresar de la siguiente forma:

$$y = \varphi\left(\sum_{j=0}^{m} w_j x_j\right) \tag{12}$$

Donde:

- y: es el resultado que transmitirá la neurona
- φ: es la función de activación
- m: es el número de entradas
- w: es el peso asociado a cada entrada
- x: es el valor de la entrada

En una comparación con una neurona real los pesos cumplen la función de las dendritas, el sumatorio la del soma y la función de activación la del axón. Se trabajará con distintas funciones de activación ya que un cambio en éstas puede proporcionar resultados distintos.

2.2.2. Redes Neuronales recurrentes

Las redes neuronales recurrentes son aquellas en las que las neuronas tienen una conexión directa o indirecta con ellas mismas (17). De esta forma la información se propaga también hacia atrás creando ciclos.

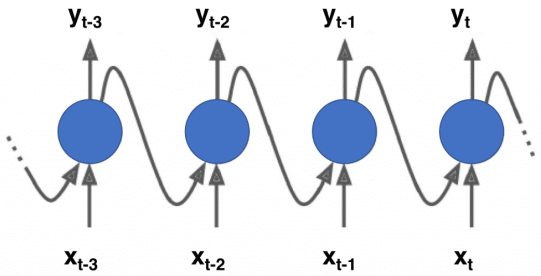


Ilustración 2.2 Neurona recurrente distintos pasos de tiempo. Tomado de (18)

Este tipo de redes neuronales son idóneas para la predicción de series temporales ya que, al utilizar datos de observaciones anteriores para la predicción actual, se consigue conservar la variable tiempo.

En este tipo de redes neuronales frecuentemente surge un problema de evanescencia del gradiente. Para solucionarlo existen algunas arquitecturas.

LSTM

Esta variante de red neuronal añade una unidad de memoria y un estado oculto a cada neurona de forma que se mantenga la información importante durante el proceso de entrenamiento y permitiendo que los datos se olviden si éstos no son relevantes (18).

La estructura de un LSTM es la siguiente:

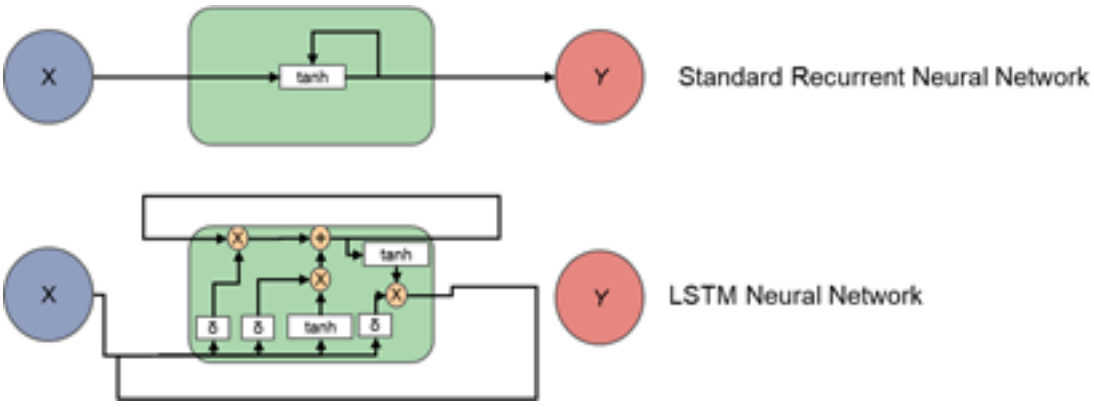


Ilustración 2.3 RNN vs LSTM. Tomado de (19)

Donde c_t son los resultados obtenidos por la unidad para la observación en el instante t, h_t es el estado oculto de la unidad en el tiempo t, x_t son los datos de entrada a la unidad LSTM para el tiempo t, X simboliza el producto término a término de sus inputs, + simboliza lo mismo pero esta vez una suma en lugar de un producto y σ representa la función sigmoide.

A la primera función sigmoide se le denomina "forget gate" . Esta puerta decide qué información es importante y cuál no. La función sigmoide actúa sobre aquella que esté dada por la entrada actual x_t y el estado oculto h_{t-1}. La función sigmoide proporciona un valor entre 0 y 1. La utilidad de esta puerta es la posibilidad de olvidar en parte el estado anterior de la unidad, multiplicándolo por valores próximos a 1 si es necesario mantener esa información o próximos a 0 en caso contrario.

El estado actual X_t y el estado oculto anterior h_{t-1} son procesados por la siguiente función sigmoide. En este caso los valores tomarán un valor cercano a 0 si son importantes y cercano a 1 en caso contrario. Esta misma información será transmitida a la función tangente hiperbólica y se realizará un producto punto por punto del resultado final. A esta puerta se la denomina "input gate".

Una vez se haya obtenido los valores resultantes de la "forget gate" y de la "input gate", estos se suman por términos y el resultado será el estado actual de la unidad c_t. De esta forma la unidad ya estará actualizada.

Finalmente, la "output gate" determinará el valor de estado oculto. Este estado contiene información de las entradas previas. Los valores del estado actual y el estado oculto anterior pasan por la tercera función sigmoide. Después, la función tangente hiperbólica transformará el nuevo estado de la unidad. Por último, el producto término a término de estos dos valores nos proporcionará el estado oculto actual de la unidad.

Las ecuaciones en términos de tiempo de una LSTM son (20):

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{13}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{14}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{15}$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{16}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{17}$$

$$h_t = o_t \odot \sigma_c(c_t) \tag{18}$$

Donde W y U contienen los pesos de la entrada y las conexiones recurrentes respectivamente. Los subíndices i se refiere a la "input gate", o a la "output gate", f a la "forget gate" y c a la celda de memoria. La variable h se refiere al estado oculto de la LSTM y $\tilde{c}_t$ el vector de activación de la celda en el instante t. σ_g es la función sigmoide y σ_c la tangente hiperbólica.

GRU

Otra solución al problema de las redes neuronales recurrentes simples son las llamadas GRU "Gated recurrent unit". Tanto la GRU como la LSTM utilizan mecanismos de puertas para controlar su proceso de memoización. No obstante, la GRU es significativamente más rápida computacionalmente ya que es más simple.

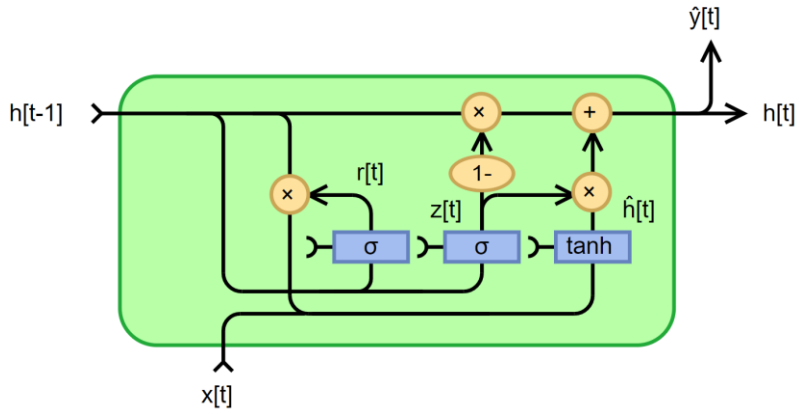


Ilustración 2.4 GRU

Donde h_t es el estado oculto de la unidad en el tiempo t, x_t son los datos de entrada a la unidad GRU para el tiempo t, x simboliza el producto término a término de sus inputs, + simboliza lo mismo pero esta vez una suma en lugar de un producto, σ representa la función sigmoide e y_t representa la salida de la unidad.

Al igual que las LSTMs, las GRUs añaden puertas y un estado oculto para controlar el flujo de la información. En este caso se utilizan dos puertas: la "update gate" y la "reset

gate". Estas puertas realizan combinaciones para decidir qué información del estado oculto debe ser actualizada o reestablecida. A diferencia de los LSTMs, la GRU solo cuenta con estas dos puertas y combina la "input gate" y la "forget gate" en la "update gate". Además, carece de puerta de salida.

La "update gate" es la responsable de determinar la cantidad de información previa que se necesita pasar al siguiente estado. El vector de entrada será sumado término a término y al aplicar la segunda función sigmoide obtendremos la salida de la puerta.

La "reset gate" tiene como función decidir qué cantidad de datos de la información pasada se necesita eliminar. Es igual a la "update gate" en cuanto a sus cálculos. Sin embargo, esta utiliza la primera función sigmoide que tendrá unos pesos distintos. La "reset gate" contiene la información relevante de la observación anterior y es multiplicada término a término por el estado oculto anterior y posteriormente se suma el vector de entrada. A lo anterior se le aplica la función tangente hiperbólica. De esta forma se decide qué información es relevante y cual no. Este resultado se denomina contenido actual en memoria.

Al contenido actual en memoria se le aplica la multiplicación termino a término por uno menos el valor de la "reset gate". Para obtener el resultado final se realiza la suma término a término de este resultado y el producto término a término del estado oculto anterior y la "reset gate".

Las ecuaciones en términos de tiempo de una GRU son (21):

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{19}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{20}$$

$$\hat{h}_t = \sigma_c(W_c x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{21}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{22}$$

Donde W y U contienen los pesos de la entrada y las conexiones recurrentes respectivamente. Los subíndices z se refiere a la "update gate" y r a la "reset gate". La variable h se refiere al estado oculto de la GRU.

2.2.3. Árboles de decisión

Los árboles de decisión consisten en un esquema de decisiones binarias a partir de unas determinadas variables de forma que se estratifica el espacio de respuesta en un número de regiones. Aun siendo de muy fácil interpretabilidad, un árbol de decisión generalmente no es el algoritmo más competente para la resolución de problemas de aprendizaje supervisado.

Aunque es posible utilizar este algoritmo en problemas de regresión, los árboles de decisión suelen ser usados para clasificación.

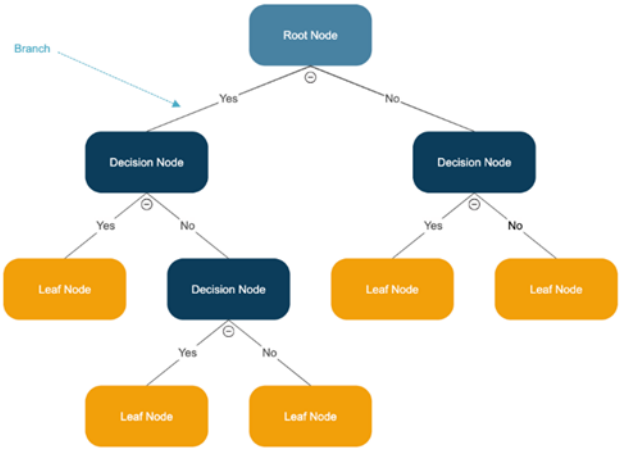


Ilustración 2.5 Árbol de decisión. Tomado de (22)

Regresión

Para construir un árbol de decisión es necesario dividir el espacio en un conjunto disjunto de posibles valores. A cada región del espacio se le asigna como valor la media de los valores del conjunto de entrenamiento pertenecientes a dicha región. Para cada observación del conjunto a clasificar se predice el valor asignado a su región. Para determinar la frontera de decisión de cada nodo se utiliza el valor que mejor divida los dos subconjuntos, es decir el que minimice el RSS definido a continuación.

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_1})^2 \tag{23}$$

Donde $\hat{y}_{R_1}$es la media de la primera región y $\hat{y}_{R_1}$de la segunda. Tras esto, se repite el proceso hasta llegar al número de grupos deseado (23).

Clasificación

En los árboles de clasificación el espacio de respuesta está naturalmente dividido por el dominio del conjunto de respuesta, ya que este es discreto. Por tanto, bastará con asignar la clase más común dentro de cada región.

Para ello se puede utilizar el error de clasificación, es decir la tasa observaciones de una región cuya clase es distinta a la asignada a ésta. Por otro lado, el índice de Gini mide la pureza de cada nodo, es decir, la similitud entre las clases de las observaciones en cada región (23).

$$G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk})$$

Capacidades

Los árboles de clasificación mejoran los modelos de regresión para aquellos datos no lineales. La similitud a un razonamiento humano y su fácil visualización facilitan en gran medida su comprensión. No obstante, para que conseguir una calidad predictiva aceptable, frecuentemente será necesario agregarlos mediante técnicas de bagging y boosting.

Bagging

Para paliar la alta varianza de los árboles de decisión, es posible agregar varios clasificadores y obtener el promedio de estos para cada observación. Para entrenar cada clasificador, se utilizan varios conjuntos de entrenamiento o varias muestras de uno. Para la clasificación se utiliza un sistema de votación en el que la clase más repetida será la predicha.

Medidas de importancia

Para la mejor interpretación del bagging podemos obtener una medida de la importancia de cada variable mediante el índice de Gini. A mayor valor en los nodos de dicha variable, más importancia tendrá el regresor.

Random Forest

Un problema del algoritmo anterior es la alta correlación que tienen generalmente los distintos árboles de la muestra, ya que todos ellos tenderán a utilizar la variable más explicativa para el nodo raíz. Para evitar esto, el algoritmo random forest selecciona un subconjunto de variables(generalmente la raíz cuadrada del total) a seleccionar para cada decisión. Esto es particularmente útil si las variables a usar tienen una alta correlación. Para ajustar un random forest a un problema de regresión, se promedia el resultado de los valores obtenidos por cada árbol.

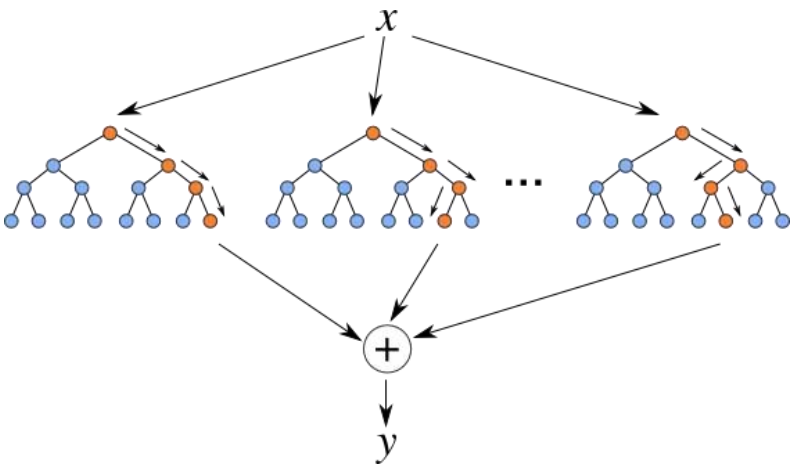


Ilustración 2.6 Random Forest. Tomado de (24)

3 Análisis de datos

3.1 Exploración de los conjuntos de datos

Para la elaboración de este proyecto se han manejado tres particiones de datos. En ellas se encuentran variables relacionadas con el consumo energético del edificio, de su refrigeración y de las condiciones externas de humedad y temperatura.

Las unidades de medida son:

- **Humedad**: Humedad relativa en porcentaje
- **Temperatura**: Grados centígrados
- **Consumo**: MWh (analizador 29), KWh (resto)

3.1.1. Partición inicial

Este fichero ya fue extraído con anterioridad al inicio de este trabajo. Contiene las medidas de varios analizadores y sensores del edificio, entre ellos los necesarios para calcular el consumo total del edificio, algunos para la sección de refrigeración y las condiciones externas de humedad y temperatura con sus respectivas fechas de medición. Los datos comprenden entre enero de 2019 y mayo de 2021.

Condiciones externas

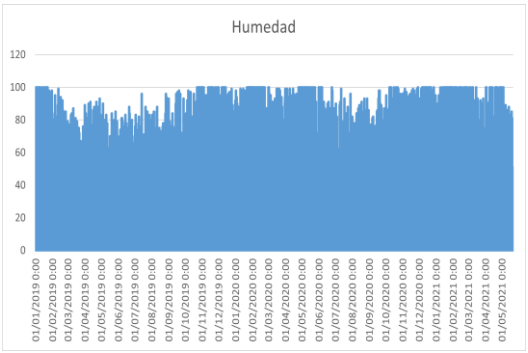


Ilustración 3.1 Humedad en el período de entrenamiento

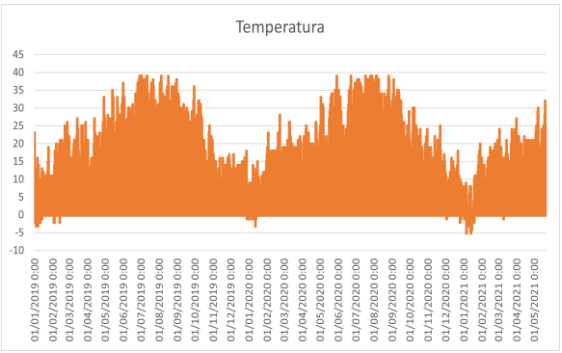


Ilustración 3.2 Temperatura en el período de entrenamiento

Consumo total del edificio

Las variables relacionadas con el consumo energético son incrementales ya que miden la demanda de forma acumulativa. Esto implica que será necesario diferenciar para obtener el consumo en cada instante.

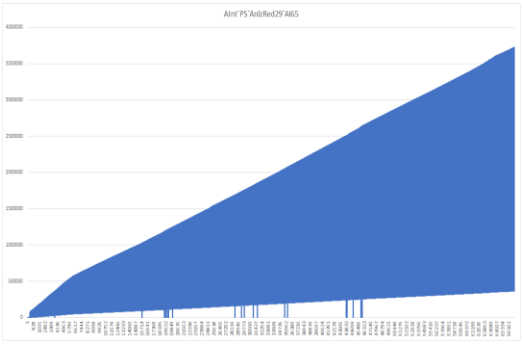


Ilustración 3.3 Analizador 29

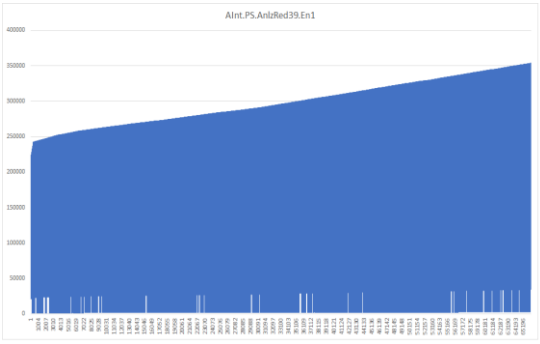


Ilustración 3.4 Analizador 39

En los datos de los analizadores hay observaciones iguales a 0 así como valores vacíos que serán desestimados.

Consumo de la sección de refrigeración

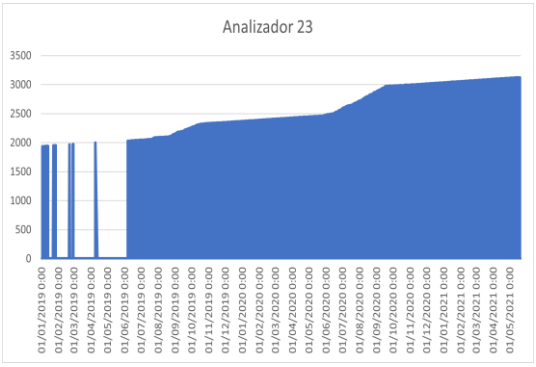


Ilustración 3.5 Analizador 23

Ilustración 3.6 Analizador 24

A la vista de estos datos, es necesario descartar los 5 primeros meses del conjunto en lo referente al consumo de esta sección, ya que estos muestran un comportamiento erróneo.

3.1.2. Fichero complementario de entrenamiento

Este fichero ha sido extraído mediante la interfaz del sistema DESIGO. Se ha obtenido debido a la ausencia de algunas de las variables necesarias para el análisis del consumo de la sección a analizar en el conjunto anterior. Contiene la información de los analizadores relacionados con la sección de refrigeración desde enero de 2019 hasta mayo de 2021.

Consumo de la sección de refrigeración

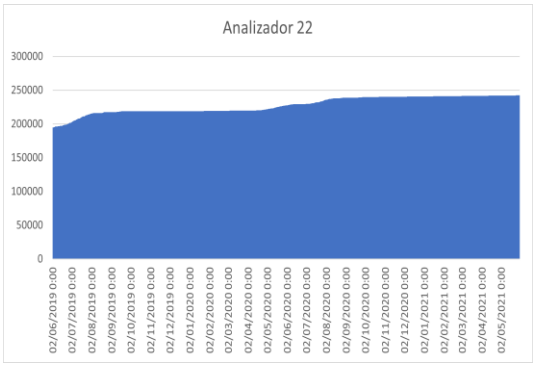


Ilustración 3.7 Analizador 22

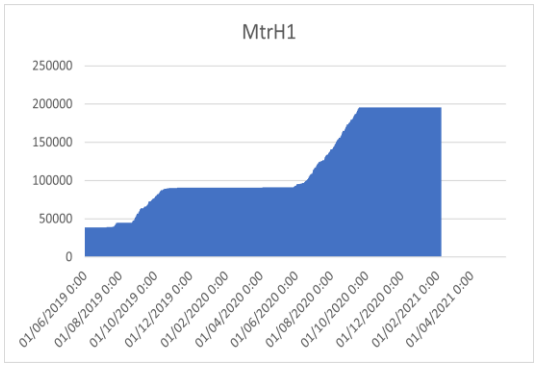


Ilustración 3.8 Variable MtrH1

Estos gráficos junto a las ilustraciones muestran la importancia de los sistemas de enfriamiento por absorción frente a los convencionales (analizador 22).

3.1.3. Fichero de prueba

Para evitar la reducción de la cantidad de datos disponibles para el entrenamiento de los modelos, se obtienen más datos para probar los distintos modelos. Este conjunto contiene tanto las mediciones de humedad y temperatura externas como las relativas al consumo energético de la sección de refrigeración.

Condiciones externas

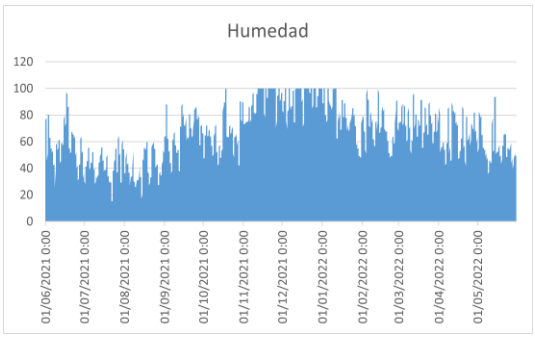


Ilustración 3.9 Humedad en el periodo de prueba

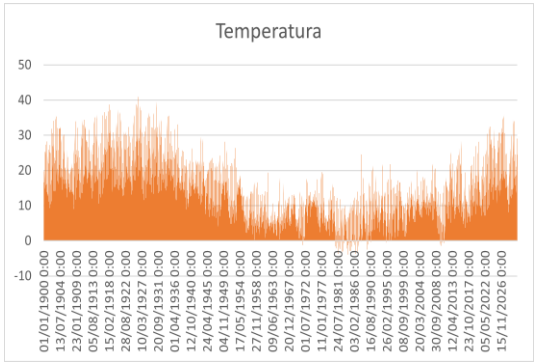


Ilustración 3.10 Temperatura en el periodo de prueba

Se pueden observar periodos de tiempo con alta temperatura y baja humedad y otros con baja temperatura y alta humedad. Resulta evidente la implicación que la estacionalidad tendrá sobre la demanda de energía para la refrigeración.

Consumo de la sección de refrigeración

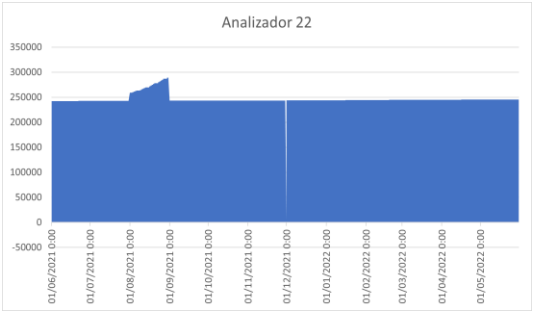


Ilustración 3.11 Analizador 22 en el periodo de prueba

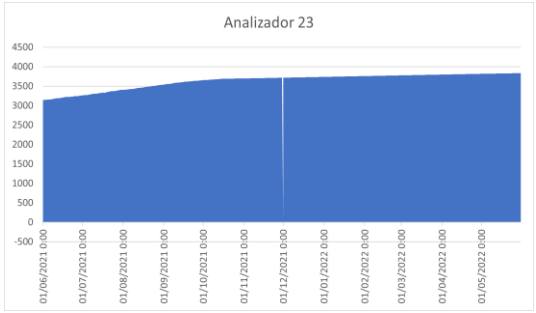


Ilustración 3.12 Analizador 23 en el periodo de prueba

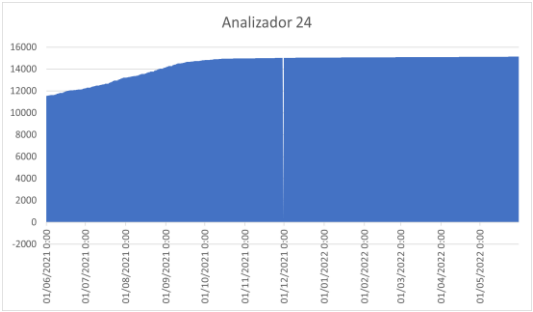


Ilustración 3.13 Analizador 24 en el periodo de prueba

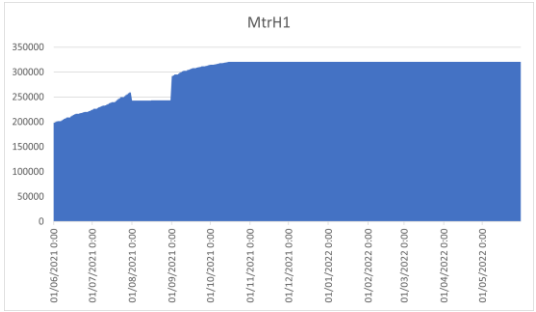


Ilustración 3.14 Variable MtrH1 en el periodo de prueba

Será necesario tratar los errores por los que el consumo energético total disminuye respecto a valores anteriores. En primer lugar, el mes de agosto del año 2021, en el que hay un descenso en la función de consumo total de la variable MtrH1 y un aumento en la demanda de la energía enfriadora convencional (analizador 22). Será necesario conservar los valores incluso siendo negativos para que el error se corrija al calcular el consumo total de la sección. También hay un error en noviembre de ese mismo año por el que la función toma el valor 0. En este último caso, será suficiente con no tener en cuenta el valor de dicho registro.

3.2 Análisis del consumo global

Primero se realizará un análisis del consumo energético total del edificio LUCIA. Para ello disponemos del conjunto de datos de las variables obtenidas por los analizadores del edificio. El conjunto de datos a analizar aquí comprende entre enero de 2019 y mayo de 2021. Véase la documentación de los sistemas del edificio.

Por tanto, se utilizará la fórmula indicada (*Consumo Edificio LUCIA(t) = AnlzRed39(t) * 1000 - AnlzRed29(t)*) para calcular la energía total consumida. En el conjunto de datos existen varias variables para cada uno de los dos analizadores.

Una variable del analizador 29 es "AInt.PS.AnlzRed29.AI65". Su nombre proviene de área de integraciones, planta sótano y analizador de red 29. Para el analizador 39 tenemos la variable "AInt.PS.AnlzRed39.En1".

Se realizan previamente varias transformaciones. En ambos analizadores se eliminan los valores vacíos (NA) y nulos (0) junto a sus correspondientes fechas. Tras esto, se realiza una agregación por horas de los valores tomando su máximo en cada una de ellas. y se eliminan los primeros seis meses ya que sus datos son erróneos. Posteriormente, se diferencian las series para obtener la energía consumida en cada hora. Finalmente aplicamos la fórmula indicada anteriormente. En aquellas horas en las que no se dispone de información se deja el valor a cero.

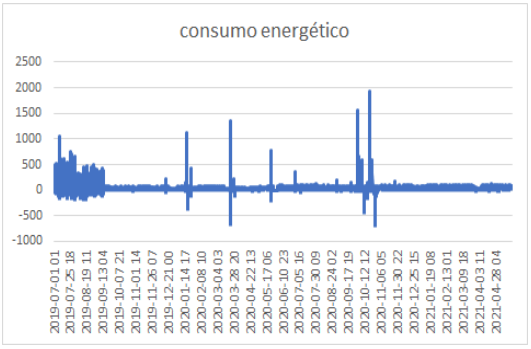


Ilustración 3.15 Consumo total (con outliers)

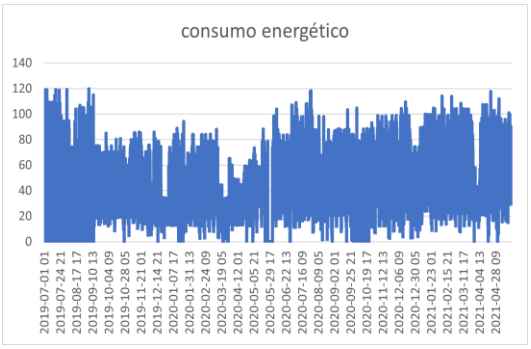


Ilustración 3.16 Consumo total (sin outliers)

A la vista del resultado, hay un número significativo "outliers" y valores negativos que impiden interpretar el consumo energético correctamente. Por consiguiente, se sustituyen los valores de los outliers por la media de las observaciones anterior y posterior y se fijan los negativos a 0. Aún tras esta corrección, se observa cierto comportamiento atípico en los meses de verano de 2019.

3.3 Resoluciones temporales

En el desarrollo de este trabajo se han agrupado los datos utilizado dos resoluciones temporales.

3.3.1. Datos por horas

Al disponer de mediciones relativamente frecuentes de las distintas variables se prueba a agrupar los datos por horas.

Condiciones externas

Ya que generalmente existen varios registros dentro de una misma hora, agruparemos las distintas observaciones utilizando la media como valor para cada una en ambas variables:

Datos de entrenamiento

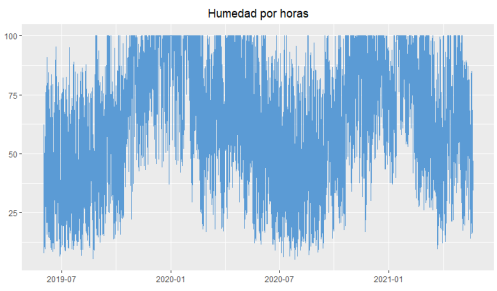


Ilustración 3.17 Humedad en el periodo de entrenamiento

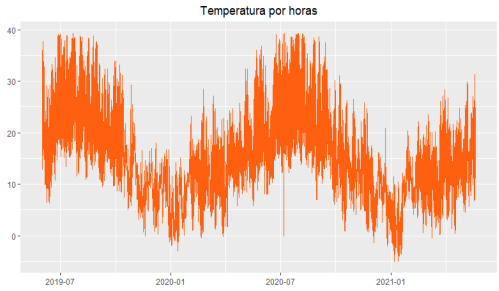


Ilustración 3.18 Temperatura en el periodo de entrenamiento

Datos de prueba

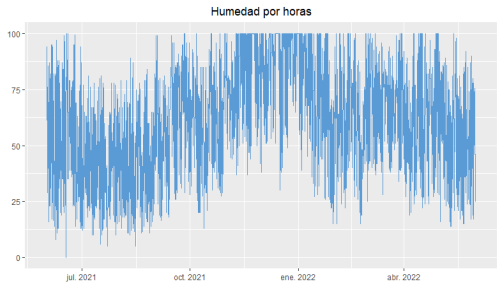


Ilustración 3.19 Humedad en el conjunto de prueba

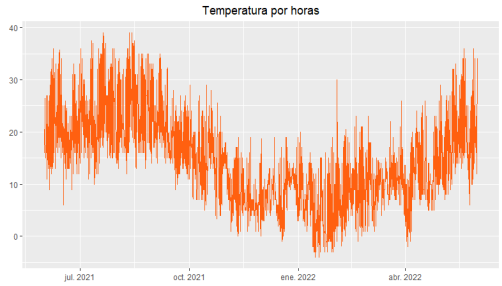


Ilustración 3.20 Temperatura en el periodo de prueba

Consumo en refrigeración

Para obtener el consumo energético horario se han tomado los máximos por hora de las series (3.5, 3.6, 3.7 y 3.8 en el caso del conjunto de entrenamiento y 3.11, 3.12, 3.13 y 3.14 en el conjunto de prueba), se ha diferenciado con respecto a la hora anterior y se han sumado las 4 variables para obtener el consumo total en cada hora. Tras esto se eliminan posibles valores vacíos, erróneos o outliers.

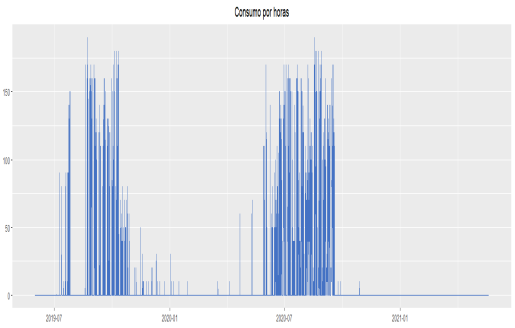


Ilustración 3.21 Consumo de la sección de refrigeración en el periodo de entrenamiento

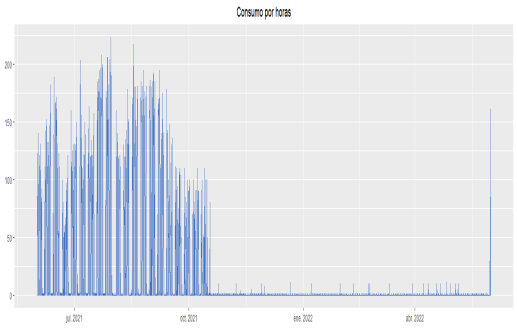


Ilustración 3.22 Consumo de la sección de refrigeración en el periodo de prueba

Se comprueba que el consumo de la sección de refrigeración es alto en los meses más cálidos, muy bajo en los de temperaturas medias y prácticamente nulo en los meses fríos. En el caso de agosto de 2019, el consumo nulo podría deberse al cierre del edificio, o a un error en los analizadores correspondientes.

3.3.2. Datos por días

Para reducir una posible inestabilidad en las distintas observaciones, se agrupan los datos por días para obtener un conjunto menos granular que reduzca el error *ver:* (5).

Condiciones externas

Para las condiciones externas, también es conveniente agrupar los datos en función de la media, en este caso diaria.

Datos de entrenamiento

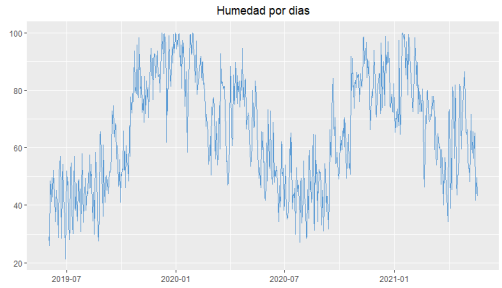


Ilustración 3.23 Humedad en el periodo de entrenamiento

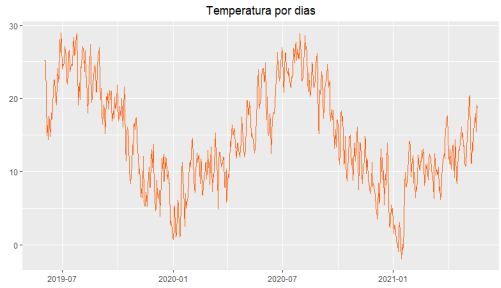


Ilustración 3.24 Temperatura en el periodo de entrenamiento

Datos de prueba

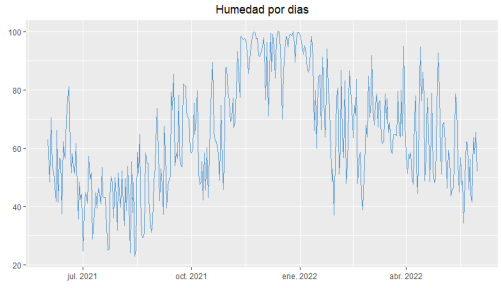


Ilustración 3.25 Humedad en el periodo de prueba

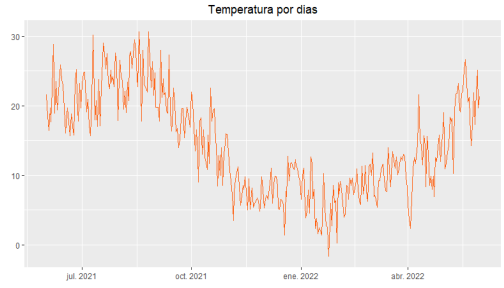


Ilustración 3.26 Temperatura en el periodo de prueba

Consumo en refrigeración

Datos de entrenamiento

Para obtener el consumo energético diario se han tomado los máximos por día de las series (3.5, 3.6, 3.7 y 3.8 en el caso del conjunto de entrenamiento y 3.11, 3.12, 3.13 y 3.14 en el conjunto de prueba), se ha diferenciado con respecto al día anterior y se han realizado transformaciones similares a las realizadas en el análisis por horas.

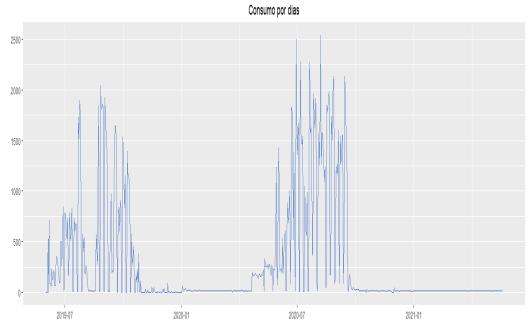


Ilustración 3.27 Consumo de la sección de refrigeración en el periodo de entrenamiento

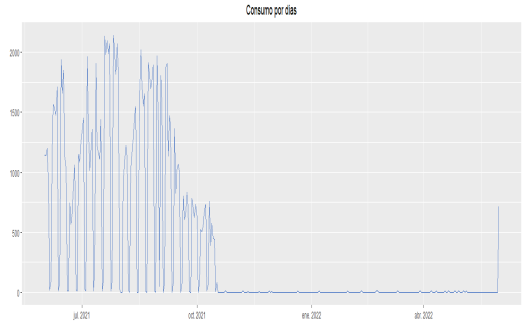


Ilustración 3.28 Consumo de la sección de refrigeración en el periodo de prueba

Al agrupar el consumo por días es posible observar una mayor continuidad en los valores altos ya que los consumos diurno y nocturno están agrupados. Esto reduce drásticamente el ruido de la serie. También se puede apreciar con mayor claridad que este consumo es prácticamente nulo los fines de semana.

3.4 Segmentación temporal

El comportamiento de la demanda energética de la sección de refrigeración tiene patrones de comportamiento distintos a lo largo del año, lo que puede provocar errores en los modelos. Debido a la restricción temporal de los datos no es posible desestacionalizar anualmente las series, por lo que se segmentarán los datos en clusters cuyo comportamiento sea similar. Esta división en tres clusters por semanas fue realizada en el Trabajo de Fin de Grado de Cristina García Sánchez (8) mediante un algoritmo de clasificación no supervisado para los años 2019 y 2020.

- **Cluster 0**: Está formado por observaciones dentro de un periodo de tiempo caracterizado por humedad alta y temperatura baja.
- **Cluster 1**: Está formado por observaciones dentro de un periodo de tiempo caracterizado por humedad y temperatura media.
- **Cluster 2**: Está formado por observaciones dentro de un periodo de tiempo caracterizado por humedad baja y temperatura alta.

3.4.1. 1NN

Para asignar a un cluster a las semanas de los años 2021 y 2022 se utilizará un clasificador 1NN (véase: KNN). Para entrenar este clasificador se obtendrán las medias de temperatura y humedad junto a las etiquetas ya asignadas en (8) de las semanas de 2019 y 2020.

Los clusters para el año 2021 son:

Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Cluster	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Semana	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Cluster	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	2	2	2
Semana	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52		
Cluster	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0		

Los clusters para el año 2022 son:

Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Cluster	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1
Semana	19	20	21	22														
Cluster	1	1	1	2														

3.4.2. Condiciones externas en cada cluster

Humedad por clusters

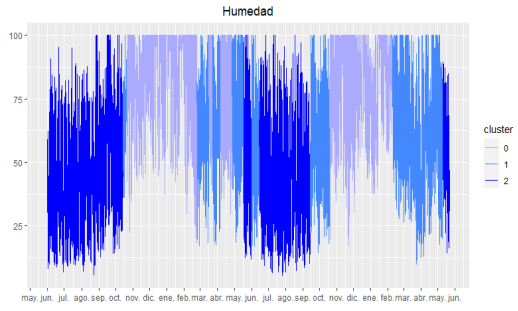


Ilustración 3.29 Humedad en el periodo de entrenamiento por clusters

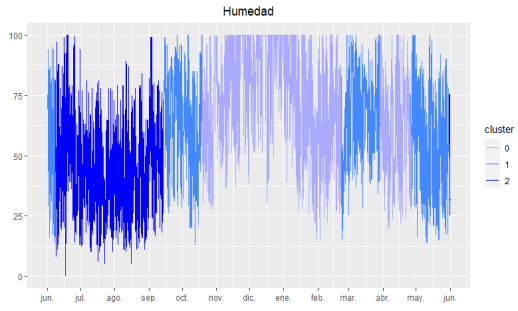


Ilustración 3.30 Humedad en el periodo de prueba por clusters

Temperatura por clusters

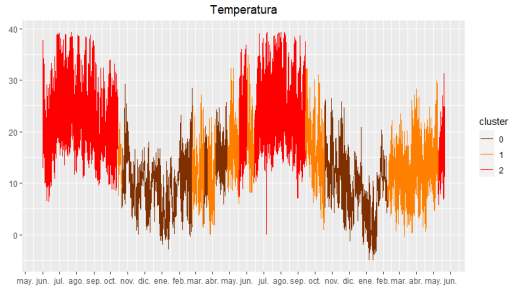


Ilustración 3.31 Temperatura en el periodo de entrenamiento por cluster

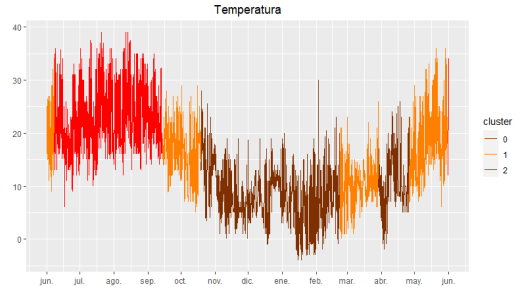


Ilustración 3.32 Temperatura en el periodo de prueba por clusters

El cluster de las temperaturas más elevadas (en rojo) es el de humedades más bajas (en azul oscuro) y el de las temperaturas más bajas (en marrón) el de humedades más altas (en gris).

3.4.3. Consumo energético en cada cluster

Consumo en cada cluster por horas

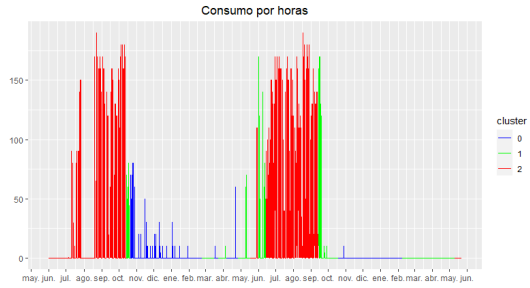


Ilustración 3.33 Consumo en el periodo de entrenamiento por clusters

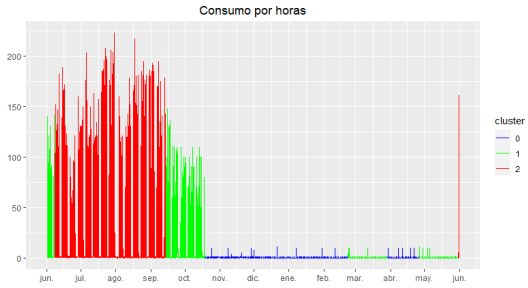


Ilustración 3.34 Consumo en el periodo de prueba por clusters

Consumo en cada cluster por días

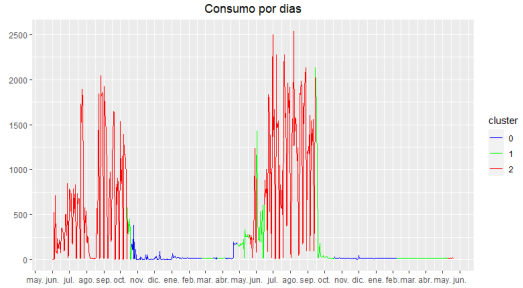


Ilustración 3.35 Consumo en el periodo de entrenamiento por clusters

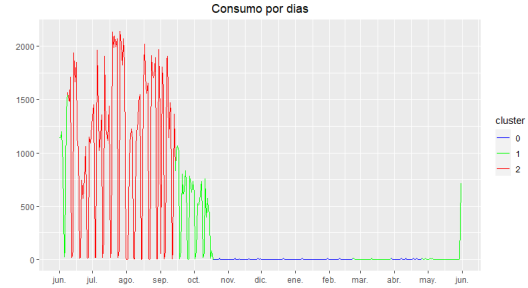


Ilustración 3.36 Consumo en el periodo de prueba por clusters

Características de los datos de cada cluster

Cluster 0

Como cabe esperar, el consumo energético de este cluster es mínimo, ya que en épocas invernales en principio no es necesaria la refrigeración del edificio. Sin embargo, puede existir cierta demanda debido a otros factores como demandas necesarias para el correcto funcionamiento de los sistemas del edificio. Estos valores no nulos no parecen seguir un patrón concreto. Las observaciones de este cluster no contribuyen en gran medida a la consumo total de las series.

Cluster 1

En períodos intermedios se observa un comportamiento muy heterogéneo de las series. En algunos casos, sobre todo en las observaciones cercanas al cluster 0, el consumo es prácticamente nulo, mientras que, en otros, sobre todo en las observaciones cercanas al cluster 2, hay un consumo significativo e incluso similar a periodos estivales. Esto puede suponer un problema al modelar los datos del cluster.

Cluster 2

En este cluster se encuentra la mayor parte del consumo de esta sección y sus observaciones determinan en gran medida la varianza del total de la serie. Por tanto, es fundamental reflejar el comportamiento de estos datos en los modelos para obtener predicciones precisas de las demandas de energía para refrigeración.

4 Técnicas de predicción utilizadas

4.1 Modelos estadísticos: SARIMA

Estos modelos permiten predecir los valores de una serie en función de los valores y errores de predicción anteriores dentro de la misma serie, lo que puede resultar de utilidad al no disponer de una gran cantidad de variables explicativas. Para su construcción se utilizará la metodología de Box Jenkins.

Las múltiples estacionalidades en las agrupaciones por horas (anual, semanal y diaria) dificultan el proceso de construcción de los modelos SARIMA horarios. Al tratarse de observaciones a una semana, es necesario utilizar valores predichos de las horas anteriores en lugar de valores reales. Las observaciones que distan un número de períodos mayor que 1 no aportan tanta información a la predicción. Además, la gran cantidad de ruido impide que las predicciones sean acertadas. Por tanto, estos modelos se ajustarán utilizando únicamente la resolución diaria.

4.1.1. Datos completos

Dado que la serie tiene una clara inestabilidad de la varianza, se aplica una transformación y = log(x+1). La serie resultante tiene estacionalidad, por lo que se aplica una diferenciación estacional.

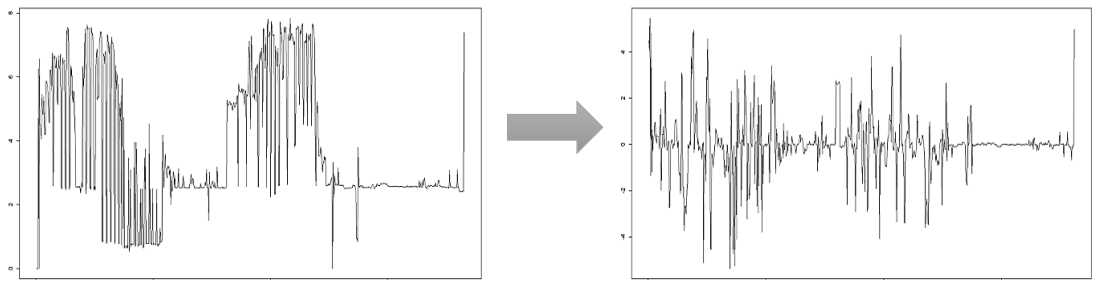


Ilustración 4.1 Diferenciación de la serie transformada con datos completos

Tras la diferenciación la serie mantiene cierta irregularidad en algunas zonas. Observando la serie se puede afirmar que no se cumple la condición de estacionariedad, por lo que no es posible ajustar un modelo.

4.1.2. Cluster 0

El comportamiento de la serie dentro de este cluster es difícil de modelar ya que la gran mayoría de valores son cercanos a 0 y los que tienen un valor algo mayor parecen no seguir un patrón determinado. Aun así, se prueba a realizar una transformación log(1+x) y diferenciar para comprobar si es posible explicar los picos de consumo.

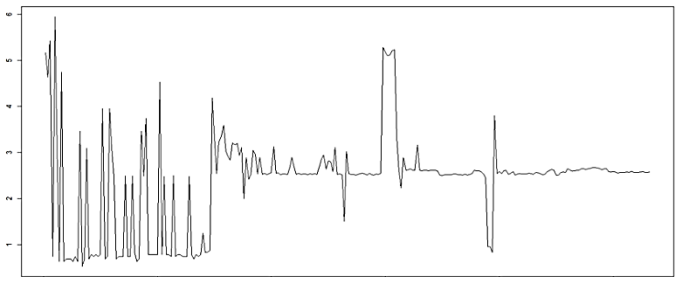


Ilustración 4.2 Datos del cluster 0 transformados

Aun tras la diferenciación es evidente que la media no es constante lo cual implica que el cluster tiene un comportamiento no estacionario. Por tanto, no será posible ajustar un modelo ARIMA.

4.1.3. Cluster 1

Este cluster al igual que el anterior será difícil de predecir. La mayoría de este cluster está formada por valores bajos, pero también contiene otros similares a los del cluster 2. Al ser heterogéneo, es probable que el rendimiento de un modelo SARIMA no sea el correcto.

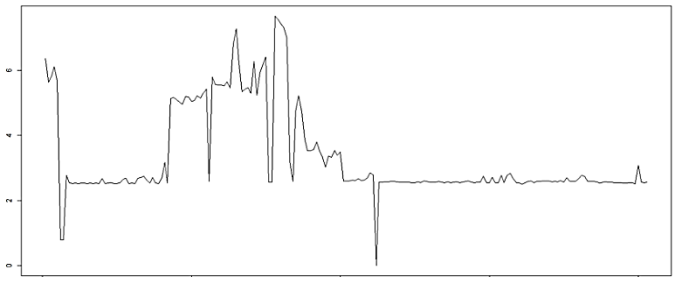


Ilustración 4.3 Datos del cluster 1 transformados

Se comprueba que, pese a transformar los datos, la media es distinta en cada tramo.

4.1.4. Cluster 2

En este cluster se recoge la mayor parte de la varianza del total de la serie por lo que es necesario analizarle de forma precisa.

Identificación

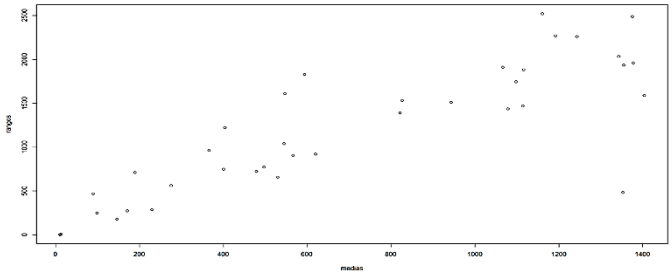


Ilustración 4.4 Diferenciación de la serie con datos completos ya transformada

Pese a ser un cluster bastante homogéneo, se puede observar una alta correlación positiva entre media y varianza. Se aplica la misma transformación que en modelo anteriores.

El periodograma de la serie indica estacionalidad por lo que se aplica una diferenciación.

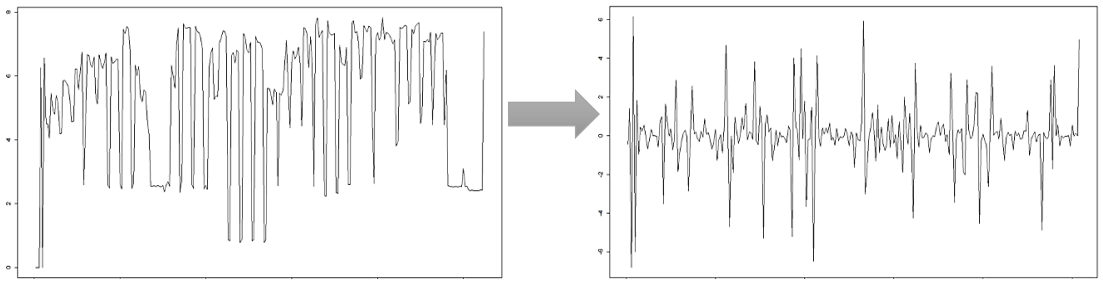


Ilustración 4.5 Diferenciación de la serie con los datos del cluster 2 ya transformada

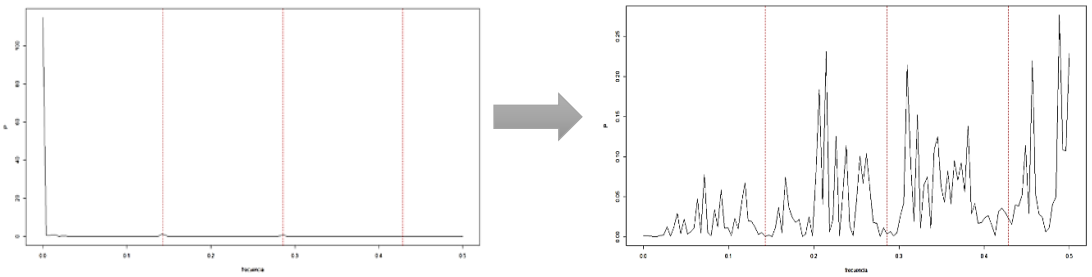


Ilustración 4.6 Periodogramas de las series transformada y diferenciada

Tras la diferenciación se desestacionaliza la serie, la cual presenta un comportamiento estacionario.

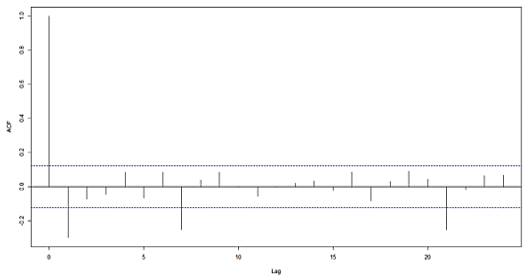


Ilustración 4.7 ACF de la serie resultante

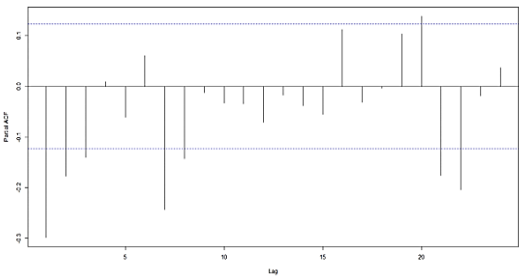


Ilustración 4.8 PACF de la serie resultante

La función de autocorrelación de la serie se anula tras el primer retardo, mientras que la función de autocorrelación parcial decrece exponencialmente a partir de éste. Tanto la ACF como la PACF se anulan tras el primer retardo múltiplo de 7. Por tanto, se ajusta un modelo SARIMA(0,0,1)(1,1,1) mediante los estimadores de máxima verosimilitud.

Estimación de parámetros y validación

Estimación de probabilidad máxima					
Parámetro	Estimación	Error estándar	t valor	Approx Pr > |t|	Retardo
MA1,1	0.16411	0.04294	3.82	0.0001	1
MA1,2	0.76886	0.06739	11.41	<.0001	7
AR1,1	0.25101	0.09752	2.57	0.0101	7

Correlaciones de las estimaciones de parámetro			
Parámetro	MA1,1	MA1,2	AR1,1
MA1,1	1.000	-0.145	-0.157
MA1,2	-0.145	1.000	0.739
AR1,1	-0.157	0.739	1.000

Ilustración 4.9 Estimación y correlación de los parámetros ajustados en el modelo SARIMA(0,0,1)(1,1,1)

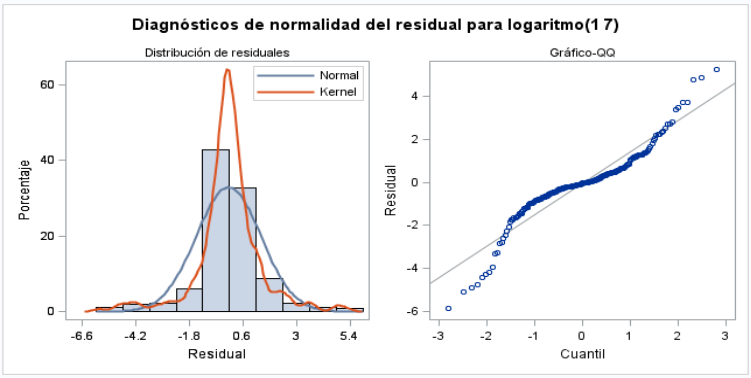


Ilustración 4.10 Gráficos de normalidad de residuos en el modelo SARIMA(0,0,1)(1,1,1)

No es posible validar este modelo debido a la alta correlación presentada entre las variables estacionales (0.739). Será necesario eliminar alguna de las dos variables. La estimación de los parámetros del modelo SARIMA(0,0,0)(1,1,1) no converge, lo que nos indica inestabilidad en el ajuste del modelo. Por tanto, se prueba un modelo SARIMA(0,0,1)(1,1,0) con los siguientes resultados:

Estimación de probabilidad máxima					
Parámetro	Estimación	Error estándar	t valor	Approx Pr > |t|	Retardo
MA1,1	0.49748	0.05631	8.84	<.0001	1
AR1,1	-0.32085	0.06092	-5.27	<.0001	7

Correlaciones de las estimaciones de parámetro		
Parámetro	MA1,1	AR1,1
MA1,1	1.000	-0.068
AR1,1	-0.068	1.000

Ilustración 4.11 Estimación y correlación de los parámetros ajustados en el modelo SARIMA(0,0,1)(1,1,0)

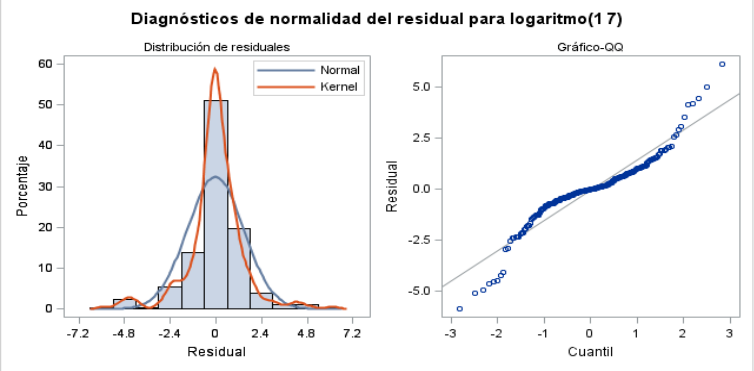


Ilustración 4.12 Gráficos de normalidad de residuos en el modelo SARIMA(0,0,1)(1,1,0)

En este modelo no se cumple la hipótesis de normalidad de residuos, por lo que no podemos validar el modelo.

4.2 Modelos de aprendizaje automático

4.2.1. Random forest

Variables

Las variables utilizadas para el entrenamiento de estos modelos son:

- **Humedad**: Humedad relativa medida en la hora o día.
- **Temperatura**: Grados centígrados en la hora o día.
- **Mes:** El mes de la observación. Aporta información al margen de las condiciones externas, incluso en los datos segmentados.
- **Día de la semana:** Es importante sobre todo para tener en cuenta si el edificio se encuentra abierto o no, como los fines de semana.
- **Hora:** Hora de la observación. Es útil a la hora de explicar la ocupación del edificio. Sólo se utiliza en modelos horarios.
- **Valor de la semana anterior**: Se utiliza el valor de la observación del mismo día y misma hora(en modelos horarios) en la semana anterior para la predicción.

Estructura y parámetros

Los parámetros utilizados para los Random Forest son:

- Número de árboles: Según (25), no es recomendable utilizar más de 200, ya que es un número lo suficientemente grande para la mayoría de datasets.
- Número de variables seleccionadas en cada división: Para este parámetro se utiliza el estándar en árboles de regresión, p/3 con redondeo hacia 0, donde p es el número total de variables. Es decir, un regresor para modelos diarios (5 variables) y 2 para modelos horarios (6 variables).

4.2.2. Redes neuronales recurrentes

Las Redes Neuronales Recurrentes (RNN) son los principales modelos de Aprendizaje Profundo para el modelado de series temporales debido a su capacidad de retroalimentación mediante predicciones anteriores.

LSTM

Como mejora de las Redes Neuronales Recurrentes simples, las LSTM han sido utilizadas en múltiples estudios sobre las demandas energéticas de los edificios (26) ya que se ven menos afectadas por el problema de la evanescencia del gradiente.

Variables

- **Valores anteriores de la serie**: Tanto en los modelos horarios como diarios se utilizarán las observaciones de la semana anterior como base para el modelo. Para que las redes neuronales funcionen correctamente será necesario estandarizar estos datos.
- **Condiciones externas**: Para mejorar las predicciones se utilizarán las medidas de temperatura y humedad estandarizadas de la semana a predecir.
- **Fecha y hora**: Dado que la temporalidad es el factor más problemático de la serie debido al limitado intervalo del conjunto de datos, es probable que al añadir esta información se mejore la precisión en el modelo (6). Para ello se utiliza una codificación One-Hot. Las variables añadidas son el mes, el día de la semana y, en el caso de los modelos horarios, la hora.

Estructura y parámetros

Se define una estructura basada en la del Trabajo de Fin de Grado de Alejandro Barón (6). La red estará formada por dos capas LSTM compuestas a su vez por 64 y 32 neuronas respectivamente. En cada una se utilizará una tasa de Dropout del 50%. Finalmente se

añade una neurona final en la que se fijan los pesos para todos los instantes de tiempo y será la responsable de la salida. Para obtener el resultado final será necesario revertir el proceso de estandarización.

En el entrenamiento se truncará la propagación del gradiente para tener un mayor contexto. La función de pérdida utilizada será el Error Medio Absoluto (MAE) y una ratio de aprendizaje de 0.001. Para los modelos diarios se utilizarán 65 épocas. Debido al elevado coste computacional, al entrenar los modelos horarios se utilizarán únicamente 16.

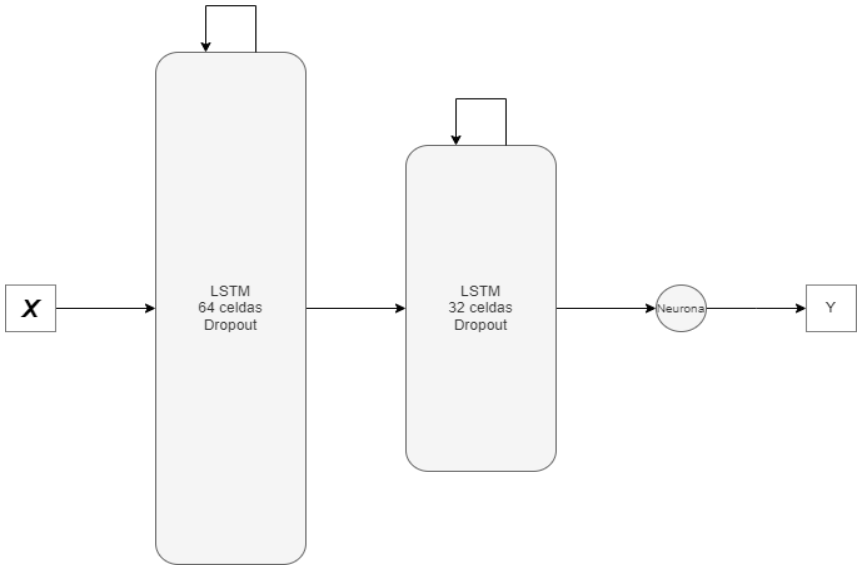


Ilustración 4.13 Diagrama de la estructura de la LSTM

GRU

Variables

Las variables utilizadas en los estos modelos son las mismas que las utilizadas en los LSTM.

Estructura y parámetros

En el estudio realizado en (27) el mejor modelo era una GRU con dos capas. Para facilitar la comparativa, se utilizará el mismo número en cada capa que en la LSTM. También se añadirá un Dropout del 50% y una neurona de salida, pero en este caso, las celdas serán del tipo GRU.

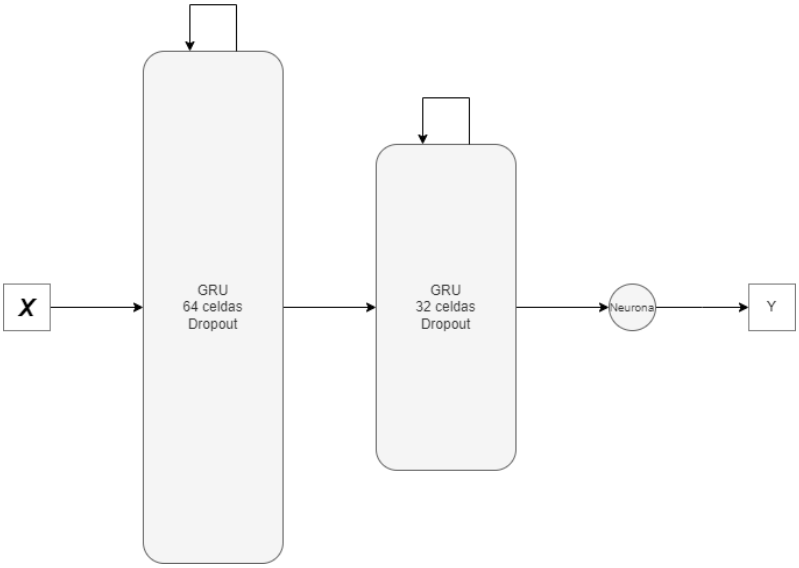


Ilustración 4.14 Diagrama de la estructura de la GRU

5 Resultados de los modelos para el consumo horario

5.1 Técnicas utilizadas para los datos completos

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	8.633317	22.7091	1.585594	0.6691577
LSTM	14.28235	41.64333	2.916347	< 0
GRU	21.01224	39.36784	2.756991	0.0003620476

Tabla 5.1 Resultados de cada modelo sobre el conjunto de validación con los datos horarios completos

Mientras que el Random Forest sí consigue explicar la mayor parte de la varianza, ambas redes neuronales no aportan ninguna información al problema.

5.1.1. Random Forest para regresión

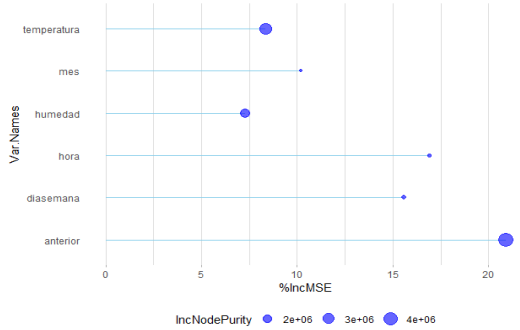


Ilustración 5.1 Importancia de las variables en el modelo horario Random Forest para los datos completos

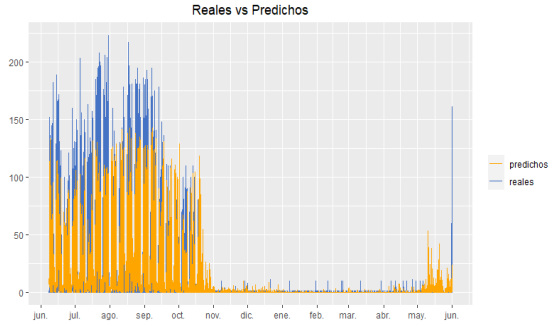


Ilustración 5.2 Gráfico de valores reales y valores predichos en el modelo horario Random Forest para los datos completos

Tras analizar el incremento del error en las observaciones *"out of bag"* se observa que todas las variables aportan información en este modelo. Destaca la importancia de la observación del periodo anterior. Las variables de tiempo (*mes*, *hora* y *diasemana*) no parecen ser útiles en aquellos árboles que las usan para tomar decisiones. Sin embargo, sí parecen reducir el error de las predicciones "out of bag". Por tanto, se puede interpretar que estas variables sí son valiosas.

Aunque las predicciones si recogen dos terceras partes de la varianza, siguen teniendo errores notables en las observaciones con un alto consumo. También se observan problemas a la hora de predecir cuándo va a comenzar la temporada de consumo alto, ya que empieza a crecer antes de tiempo. El modelo no parece ser lo suficientemente efectivo para los datos anuales y el CV-RMSE es muy superior al 0.3 objetivo.

5.1.2. Redes Neuronales Recurrentes

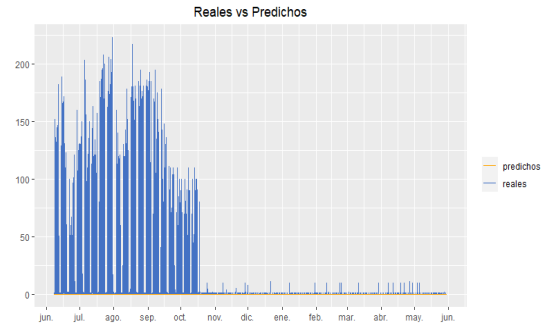


Ilustración 5.3 Gráfico de valores reales y valores predichos en el modelo horario GRU para los datos completos

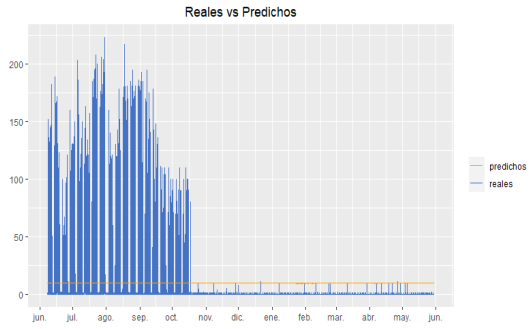


Ilustración 5.4 Gráfico de valores reales y valores predichos en el modelo horario LSTM para los datos completos

La gran cantidad de valores iguales o cercanos a 0 provoca que las redes neuronales aprendan a predecir valores muy bajos para no aumentar el error. Estos modelos quedan descartados.

5.2 Técnicas utilizadas para el cluster 0

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	0.620656	2.440453	25.26929	<0
LSTM	0.09474978	0.6393068	6.753875	≈0
GRU	0.09493996	0.6392749	6.753537	<0

Tabla 5.2 Resultados de cada modelo sobre el conjunto de validación con los datos horarios del cluster 0

Ningún modelo es capaz de descubrir ningún patrón en los datos, aunque aparentemente las redes neuronales tengan un error bajo.

5.2.1. Random Forest para regresión

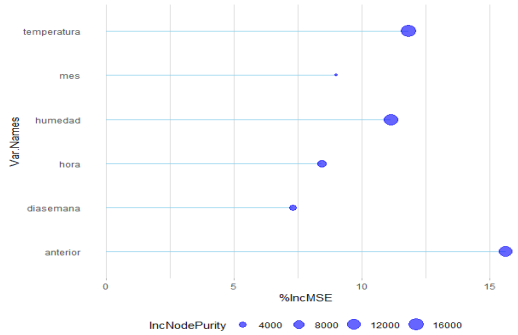


Ilustración 5.5 Importancia de las variables en el modelo horario Random Forest para los datos del cluster 0

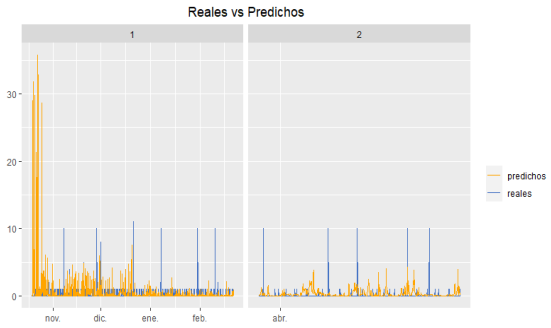


Ilustración 5.6 Gráfico de valores reales y valores predichos en el modelo horario Random Forest para los datos del cluster

Al igual que en el modelo anterior, el valor de la semana anterior es la variable más explicativa. En este caso, son más influyentes las condiciones externas que las variables de tiempo, con un aumento notable de la humedad.

No se consigue el objetivo planteado de predecir los picos de consumo en este cluster.

5.2.2. Redes Neuronales Recurrentes

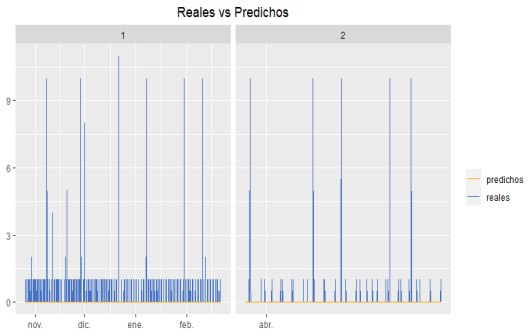


Ilustración 5.7 Gráfico de valores reales y valores predichos en el modelo horario GRU para los datos del cluster 0

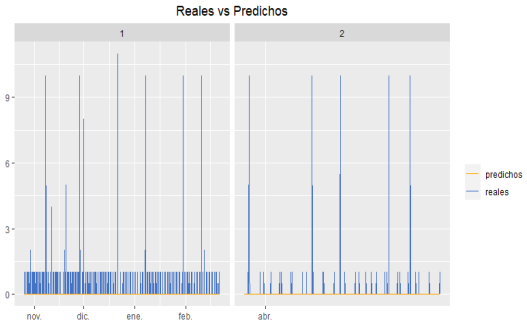


Ilustración 5.8 Gráfico de valores reales y valores predichos en el modelo horario LSTM para los datos del cluster 0

Al igual que en el modelo anterior, las redes neuronales recurrentes predicen valores practicamente nulos debido a la gran cantidad de ceros en el conjunto de entrenamiento.

5.3 Técnicas utilizadas para el cluster 1

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	4.627509	13.81387	2.033992	0.5632323
LSTM	7.118865	21.86815	3.219925	≈0
GRU	18.64781	22.4091	3.299576	0.003414526

Tabla 5.3 Resultados de cada modelo sobre el conjunto de validación con los datos horarios del cluster 1

El Random Forest proporciona mejores resultados que las redes neuronales para este cluster, aunque dista de ser suficiente para validar el modelo. Teniendo en cuenta la baja media del cluster, es difícil que la métrica CV-RMSE sea baja.

5.3.1. Random Forest para regresión

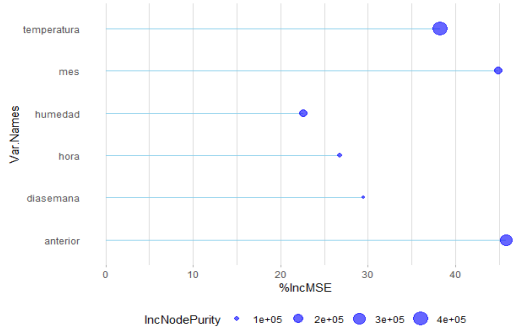


Ilustración 5.9 Importancia de las variables en el modelo horario Random Forest para los datos del cluster 1

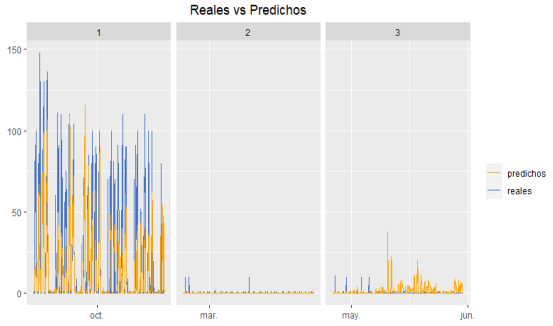


Ilustración 5.10 Gráfico de valores reales y valores predichos en el modelo horario Random Forest para los datos del cluster 1

En este cluster hay un gran incremento en la influencia de las variables con respecto a los demás. Esto es consecuencia de la heterogeneidad del cluster, por la que es necesario separar las observaciones con consumo elevado de las que prácticamente no tienen.

El modelo funciona relativamente bien para las condiciones del cluster, consiguiendo detectar el comportamiento de la serie. No obstante, las predicciones distan de ser precisas por lo que queda gran parte de la varianza sin explicar.

5.3.2. Redes Neuronales Recurrentes

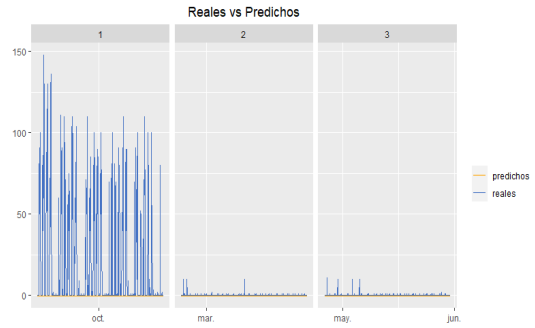


Ilustración 5.11 Gráfico de valores reales y valores predichos en el modelo horario GRU para los datos del cluster 1

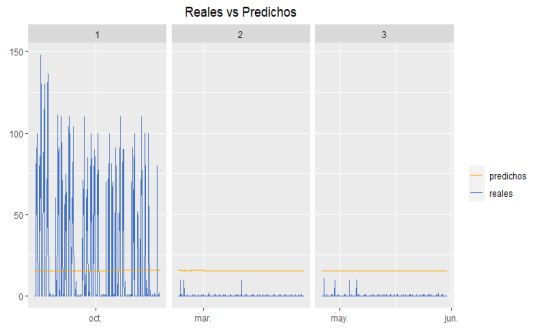


Ilustración 5.12 Gráfico de valores reales y valores predichos en el modelo horario LSTM para los datos del cluster 1

Las Redes Neuronales Recurrentes tampoco funcionan en este caso.

5.4 Técnicas utilizadas para el cluster 2

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	23.09768	37.87494	0.8495742	0.6525812
LSTM	44.25941	76.04067	1.720215	<0
GRU	47.76517	63.99274	1.447663	0.03174211

Tabla 5.4 Resultados de cada modelo sobre el conjunto de validación con los datos horarios del cluster 2

Aun teniendo un MAE y un RMSE superior a los de clusters anteriores, el modelo Random Forest es capaz de explicar más varianza y tiene un CV-RMSE menor. Esto se debe a los altos valores del cluster.

5.4.1. Random Forest para regresión

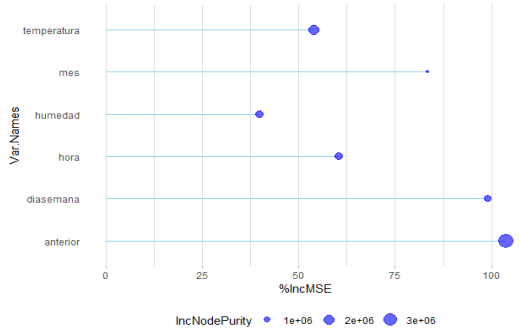


Ilustración 5.13 Importancia de las variables en el modelo horario Random Forest para los datos del cluster 2

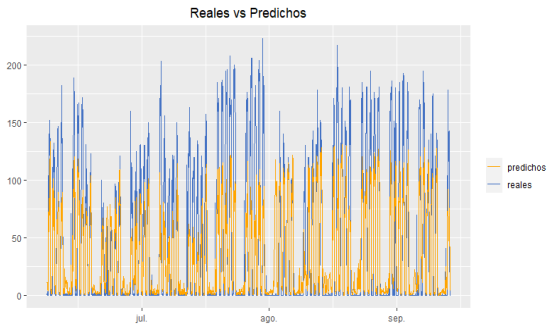


Ilustración 5.14 Gráfico de valores reales y valores predichos en el modelo horario Random Forest para los datos del cluster 2

En este caso, todas las variables afectan en mayor o menor medida al error de predicción, siendo las más explicativas el consumo de la semana anterior a la misma hora y el día de la semana, mostrando la importancia de la estacionalidad semanal. Por otro lado, la menos explicativa es la humedad.

Al igual que en el cluster anterior, el modelo logra entender el comportamiento de la serie, pero sigue sin acertar en la magnitud del consumo.

5.4.2. Redes Neuronales Recurrentes

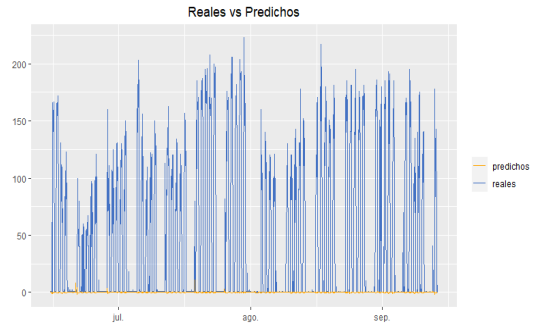


Ilustración 5.15 Gráfico de valores reales y valores predichos en el modelo horario GRU para los datos del cluster 2

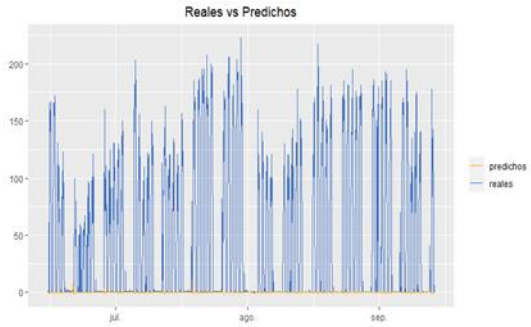


Ilustración 5.16 Gráfico de valores reales y valores predichos en el modelo horario LSTM para los datos del cluster 2

Aun siendo el cluster con valores de consumo más altos, la mediana es igual a 0, lo que implica que más de la mitad de las observaciones es igual a 0. Las redes siguen sin ser útiles en este caso.

6 Resultados de los modelos para el consumo diario

6.1 Técnicas utilizadas para los datos completos

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	146.4574	268.8648	0.7933652	0.8116589
LSTM	111.5707	209.4952	0.7086921	0.8541537
GRU	178.1814	378.3798	1.119961	0.6267606

Tabla 6.1 Resultados de cada modelo sobre el conjunto de validación con los datos diarios completos

Los modelos basados en redes neuronales mejoran mucho en su predicción con respecto a los datos horarios, especialmente la LSTM. También se logra una mejora sustancial del Random forest Aun así, no consiguen llegar a su objetivo.

6.1.1. Random Forest para regresión

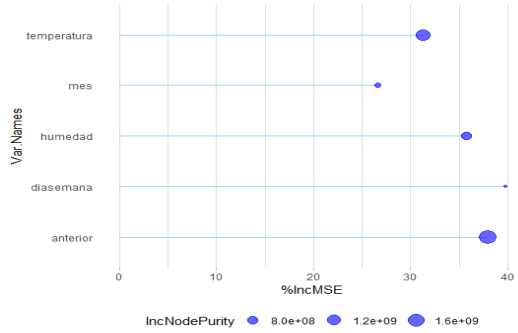


Ilustración 6.1 Importancia de las variables en el modelo diario Random Forest para los datos del cluster completos

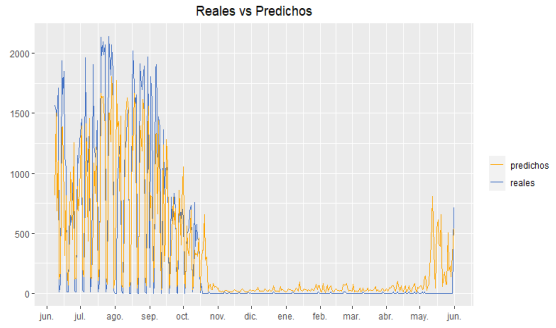


Ilustración 6.2 Gráfico de valores reales y valores predichos en el modelo diario Random Forest para los datos completos

Se observa una importancia similar en todas las variables.

El modelo es capaz de aproximar el valor real de la serie. Al igual que en el modelo horario, no es capaz de predecir el momento en el que los valores comienzan a crecer. El error absoluto decrece con respecto al modelo con datos horarios si se tiene en cuenta que cada observación del modelo diario comprende 24 del anterior.

6.1.2. Redes Neuronales recurrentes

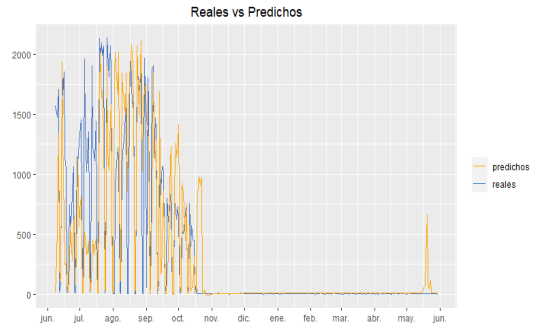


Ilustración 6.3 Gráfico de valores reales y valores predichos en el modelo diario LSTM para los datos completos

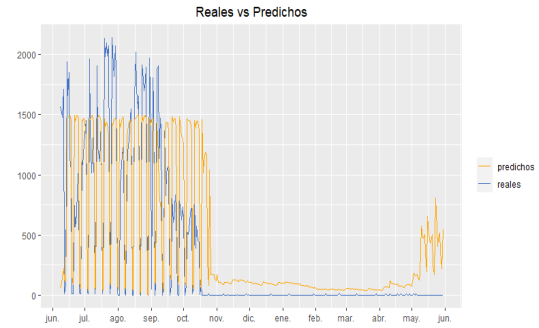


Ilustración 6.4 Gráfico de valores reales y valores predichos en el modelo diario GRU para los datos completos

En este caso, existen diferencias significativas entre la LSTM y la GRU. La LSTM ajusta relativamente bien la serie, aunque no es capaz de reducir el coeficiente de variación del RMSE por debajo del 0.3. La GRU solo es capaz de detectar cuándo el consumo es alto o bajo, pero no consigue aproximarse al valor exacto.

6.2 Técnicas utilizadas para el cluster 0

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	24.44943	37.04223	16.03113	<0
LSTM	208.2257	208.6927	92.1638	<0
GRU	372.6139	389.0628	171.8197	<0

Tabla 6.2 Resultados de cada modelo sobre el conjunto de validación con los datos diarios del cluster 0

Ningún modelo aporta información en este cluster.

6.2.1. Random Forest para regresión

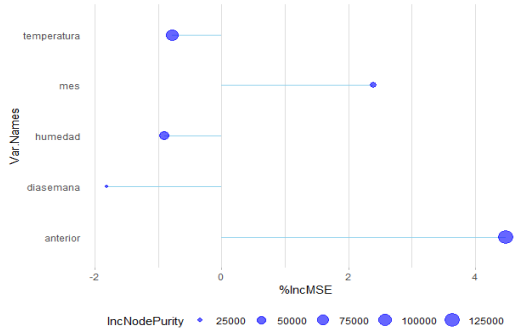


Ilustración 6.5 Importancia de las variables en el modelo diario Random Forest para los datos del cluster 0

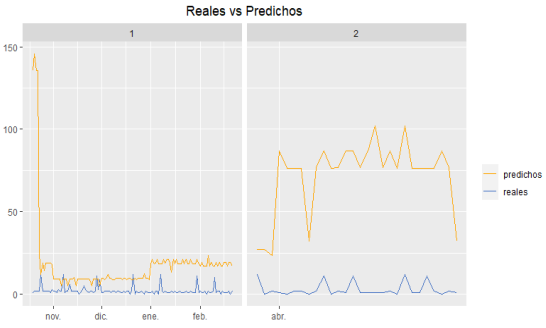


Ilustración 6.6 Gráfico de valores reales y valores predichos en el modelo diario Random Forest para los datos del cluster 0

En este modelo se observa que ninguna de las variables aporta información. Se descarta el modelo.

6.2.2. Redes Neuronales Recurrentes

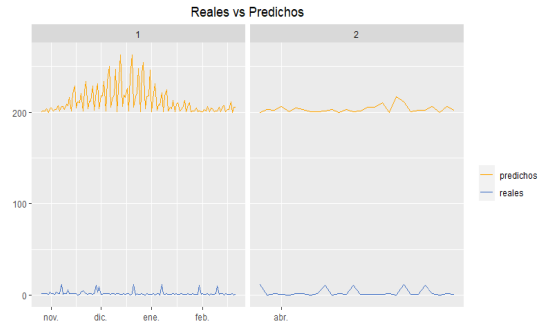


Ilustración 6.7 Gráfico de valores reales y valores predichos en el modelo diario GRU para los datos del cluster 0

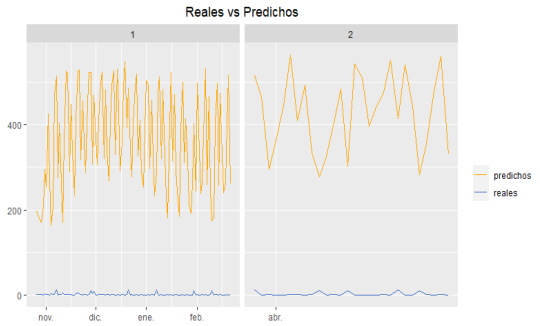


Ilustración 6.8 Gráfico de valores reales y valores predichos en el modelo diario LSTM para los datos del cluster 0

Ninguna de las redes es capaz de ajustarse a los datos.

6.3 Técnicas utilizadas para el cluster 1

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	168.9841	264.0375	1.627346	0.2355448
LSTM	1149.556	1498.824	9.237721	<0
GRU	419.0052	484.7888	2.987905	<0

Tabla 6.3 Resultados de cada modelo sobre el conjunto de validación con los datos diarios del cluster 1

Tanto la LSTM como la GRU vuelven a no aportar ningún resultado útil.

6.3.1. Random Forest para regresión

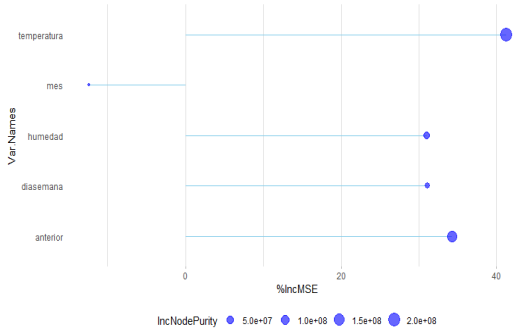


Ilustración 6.9 Importancia de las variables en el modelo diario Random Forest para los datos del cluster 1

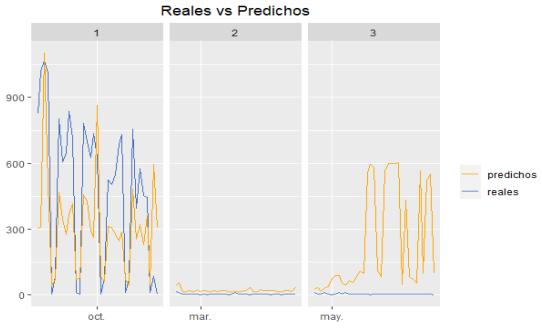


Ilustración 6.10 Gráfico de valores reales y valores predichos en el modelo diario Random Forest para los datos del cluster 1

La variable mes es la única que no toma ninguna importancia. Aunque esto pueda parecer contraintuitivo, sobre todo en un cluster donde el consumo depende de la cercanía al periodo de verano, puede deberse a que este fenómeno está explicado por la temperatura.

La bondad del modelo no es constante en los datos de entrenamiento. Los resultados distan de ser útiles.

6.3.2. Redes Neuronales Recurrentes

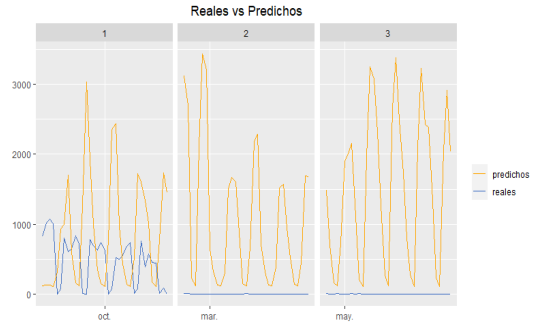


Ilustración 6.11 Gráfico de valores reales y valores predichos en el modelo diario LSTM para los datos del cluster 1

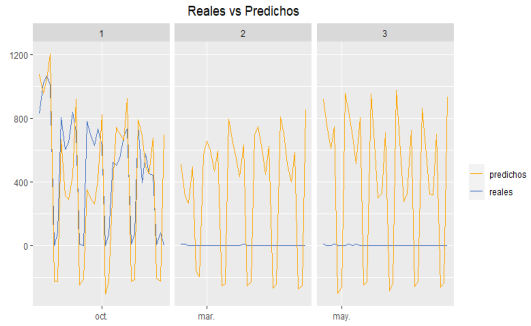


Ilustración 6.12 Gráfico de valores reales y valores predichos en el modelo diario GRU para los datos del cluster 1

Las Redes Neuronales tampoco tienen capacidad predictiva en este cluster.

6.4 Técnicas utilizadas para el cluster 2

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	368.7814	458.68	0.4387741	0.6707366
LSTM	177.9383	231.7649	0.2232505	0.9453402
GRU	288.087	367.8498	0.354336	0.8596427

Tabla 6.4 Resultados de cada modelo sobre el conjunto de validación con los datos diarios del cluster 2

En este cluster sí se logran predicciones con un CV-RMSE menor que 0.3. Las redes neuronales funcionan mejor que el Random Forest en este caso. Entre las dos redes neuronales, la LSTM funciona mejor, llegando a explicar cerca del 95% de la varianza.

6.4.1. Random Forest para regresión

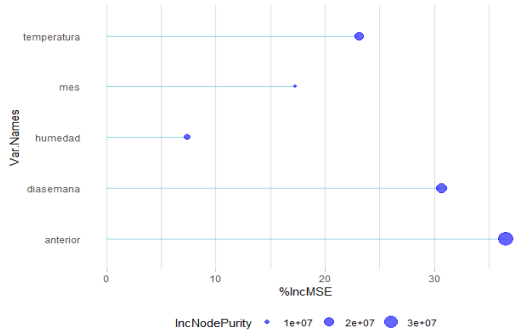


Ilustración 6.13 Importancia de las variables en el modelo diario Random Forest para los datos del cluster 2

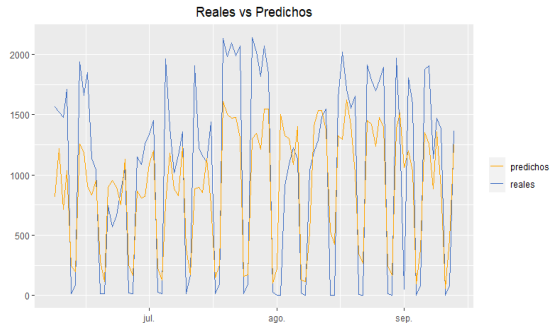


Ilustración 6.14 Gráfico de valores reales y valores predichos en el modelo diario Random Forest para los datos del cluster 2

En este modelo las variables más importantes son el día de la semana y el valor del mismo día de la semana anterior, mientras que la humedad pierde importancia.

El modelo predice los días con consumo, aunque al igual que en modelos anteriores, las predicciones de valores altos son demasiado bajas Esto podría deberse a un mayor consumo en el año predicho (2021) que en los datos del conjunto de entrenamiento.

6.4.2. Redes Neuronales Recurrentes

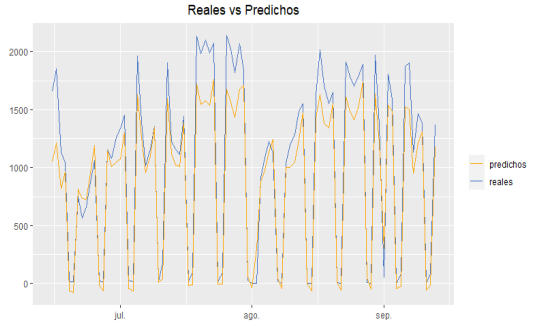


Ilustración 6.15 Ilustración 6.9 Gráfico de valores reales y valores predichos en el modelo diario LSTM para los datos del cluster 2

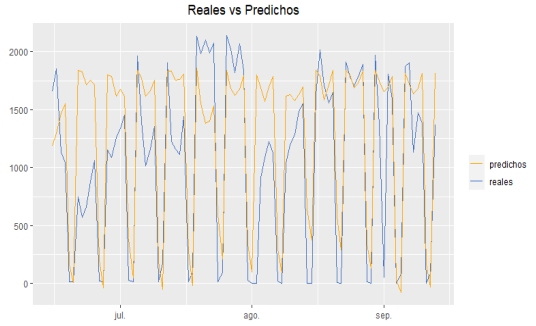


Ilustración 6.16 Ilustración 6.9 Gráfico de valores reales y valores predichos en el modelo diario GRU para los datos del cluster 2

La LSTM es el único modelo que consigue bajar del CV-RMSE objetivo. Por tanto, podemos decir que es un modelo válido para la predicción del consumo energético dentro del periodo delimitado por el cluster 2.

La GRU obtiene también buenos resultados, pero sin llegar al objetivo.

7 Conclusiones y trabajo futuro

7.1 Conclusiones sobre el conjunto de datos

El conjunto de datos contiene varios periodos irregulares. Los primeros meses de 2019 han sido descartados por su comportamiento inconsistente con el resto de los datos. Esto, sumado a otros problemas como outliers o valores vacíos dificultan su análisis.

La serie tiene una componente estacional muy elevada, donde la mayor parte de los datos tiene un valor casi nulo. Esto supone un problema evidente a la hora de modelizar la serie completa. La solución de segmentar los datos en función de la magnitud de su consumo permite analizar mejor los valores de mayor interés, es decir, los de altas demandas de energía. No obstante, los clusters de menor consumo tienen varios momentos puntuales de consumo los cuales no parecen depender de ningún factor conocido. Además, no se elimina del todo la componente estacional dentro de cada cluster, sobre todo en el cluster de consumo medio.

Pese a que un periodo de dos años debería ser suficiente para elaborar un modelo predictivo del consumo total según (5), el hecho de que una gran parte de los datos no aporte información sobre la sección especifica a analizar muestra la necesidad de ampliar el rango temporal del experimento.

7.2 Conclusiones sobre los modelos utilizados

La capacidad predictiva de los modelos se basa principalmente en el histórico de datos anteriores. Para la mejora de las predicciones se añaden las condiciones externas, siendo la temperatura la más importante. En el cluster de consumo medio, en el cual aumenta en función a la cercanía a temporadas de verano, adquiere mayor relevancia. Las variables de mes, día de la semana y hora aportan información sobre el uso y ocupación del edificio. Este es uno de los principales factores de consumo dentro de este (27).

Los modelos ARIMA no se ajustan correctamente a los datos disponibles. Esto se debe principalmente a dos factores: las diferencias de comportamiento en distintos periodos y el uso exclusivo de valores anteriores para la predicción. Los Random Forest funcionan mejor que las redes neuronales en datos horarios. Esto puede ser explicado por la no linealidad de la respuesta en función de las variables de tiempo (23). No obstante, en modelos diarios, las redes neuronales mejoran las predicciones de los Random Forest en el cluster de mayor consumo.

El mejor modelo en términos del CV-RMSE es la LSTM en el cluster 2, siendo el único que consigue bajar del umbral del 0.3 establecido por la ASHRAE. Aunque sólo predice valores dentro de este cluster, este representa un 87,6% del total del consumo del edificio.

Cuando los datos presentan algún tipo de patrón temporal (como en el caso de los datos completos o del cluster 2) los modelos de cluster son más precisos (con relación al CV-RMSE) que los modelos completos; sin embargo, si los clusters presentan patrones anómalos o muy distintos (como en los clusters 0 y 1) esta suposición no se cumple y los resultados parecen invitar a buscar otro tipo de modelos.

7.3 Futuras líneas de trabajo

- Aumentar el rango temporal de los datos mejoraría las capacidades predictivas de los modelos y permitiría desestacionalizar la serie mejor. Esto ayudaría a que las redes neuronales aprendan a sintetizar el comportamiento de la serie a pesar de sus irregularidades.
- Actualmente el edificio LUCIA recoge más variables relativas a las condiciones externas que podrán ser utilizadas en futuros modelos. Además, existen factores relacionadas con sistemas energéticos, mantenimiento y actividad del edificio o la calidad ambiental interior que ayudarían a explicar el consumo energético.

8 Bibliografía

1. **Department of Energy.** The impact of mandatory energy audits on building energy use. *About the Commercial Buildings Integration Program.* [En línea] 2016. https://energy.gov/eere/buildings/about-commercial-buildings-integration-program.

2. **Asdrubali, F.; Baggio, P.; Grazieschi, G.; Guattari, C.** *Dynamic life cycle assessment modelling of a NZEB building.* 116489, Roma : s.n., 2019, Vol. 191. 0360-5442.

3. **Zorita Lamadrid, Ángel L.; Moríñigo Sotelo, Daniel; Duque Pérez, Óscar; González González, Sergio L.; Valbuena García, Francisco J.; Osornio Ríos, Roque A.; Morales Velázquez, Luis** *Analisis de la contribución fotovoltaica en un edificio de cero emisiones.* Soria : s.n.

4. **Amasyali, Kadir y El-Gohary, Nora M.** *A review of data-driven building energy consumption prediction studies.* Urbana : ELSEVIER, 2018.

5. **ASHRAE.** Measurement of Energy, Demand and Water Savings. *ASHRAE Guideline 14-2014.* [En línea] 18 de 12 de 2014.

6. **Barón García, Alejandro.** *Energy load management in Smart Buildings with Deep Learning Techniques.* Trabajo de Fin de Grado de Ingeniería Informática. Universidad de Valladolid. Director: Alonso González, Carlos Javier y Pulido Junquera, José Belarmino. Valladolid, 2020.

7. **Arias-Requejo, D., Alonso-González, C. J., Pulido, B., & Keane, M. M.** *Advanced machine learning techniques for predictive maintenance of HVAC subsystems based on energy consumption prediction.* Loughborough : *Proceedings of BSO-V2020, 5.* 2020.

8. **García Sánchez, Cristina.** *Descubrimiento de conocimiento (clustering) en el consumo energético en el edificio inteligente LUCIA.* Trabajo de Fin de Grado de Estadística. Facultad de Ciencias. Universidad de Vallaodlid. Director: Pulido Junquera, José Belarmino. Valladolid, 2023.

9. **Rodríguez del Tío, María del Pilar.** Apuntes de la asignatura de Análisis de Series Temporales del Grado en Estadística. Universidad de Valladolid. Valladolid : s.n., 2021.

10. **Peña, Daniel.** *Análisis de series temporales.* Madrid : Alianza Editorial, 2010.

11. **Luger, George F. y Stubblefield, W. A.** *Artificial Intelligence. Structures and Strategies for Complex Problem Solving.* s.l. : The Benjamin/Cummings Publishing Company, Inc, 2005.

12. **Kaplan, Andreas.** *Artificial Intelligence, Business and Civilization: Our Fate Made in Machines.* s.l. : Routledge, 2022.

13. **Romero, Juan Jesús; Dafonte, Carlos; Gómez, Ángel; Penousal, Fernando Jorge** *Inteligencia artificial y computación avanzada.* Santiago de Compostela : Fundación Alfredo Brañas, 2007.

14. **Sierra Araujo, Basilio.** *Aprendizaje automático: conceptos básicos y avanzados.* s.l. : Pearson Prentice Hall, 2006.

15. **Bengio, Yoshua, Courville, Aaron y Vincent, Pascal.** *Representation learning: a review and new perspectives..* 8, s.l. : IEEE, 8 de 2013, Vol. 35.

16. **Clark, W. A.. y Farley, B. G.** Simulation of Self-Organizing Systems by Digital Computer. s.l. : IRE Transactions on Information Theory, 1954.

17. **Hochreiter, Sepp y Schmidhuber, Jürgen.** *Long Short-term Memory.* 8, s.l. : Neural Computation, 12 de 1997, Vol. 9.

18. **Torres, Jordi.** Deep learning: introducción práctica con Keras. Barcelona, 2018.

19. **Garzón, José Ignacio.** Cómo usar redes neuronales (LSTM) en la predicción de averías en las máquinas. [En línea] 6 de 11 de 2018. https://blog.gft.com/es/2018/11/06/como-usar-redes-neuronales-lstm-en-la-prediccion-de-averias-en-las-maquinas/.

20. **Hochreiter, Sepp y Schmidhuber, Jürgen.** *Long Short-term Memory.* MIT Press: Neural Computation, 1997, Vol. 9.

21. **Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun; Bengio, Yoshua.** *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.* 2014

22. **Smartdraw.** Decision Tree. *Smartdraw.* [En línea] https://wcs.smartdraw.com/decision-tree/img/structure-of-a-decision-tree.png?bn=15100111874.

23. **James, Gareth, Robert Tibshirani, Daniela Witten y Hastie, Trevor.** Tree-Based Methods. *An Introduction to Statistical Learning.* 2013.

24. **Chu, Zheng, Yu, Jiong y Hambdulla, Askar.***Throughput prediction based on ExtraTree for stream processing tasks.* 31, comsis.org : Computer Science and Information Systems, 8 de 6 de 2020, Vol. 18.

25. **Latinne, Patrice, Debeir, Oliver y Decaestecker, Christine.** *Limiting the Number of Trees in Random Forests.* [ed.] Josef Kitler y Fabio Roli. International Workshop on Multiple Classifier Systems : Springer, Berlin, Heidelberg, 2001.

26. **Marino, Daniel L., Amarasinghe, Kasun y Manic, Milos.** *Building Energy Load Forecasting using Deep Neural Networks.* IECON 2016 - 42nd Annual Conference of the

IEEE Industrial Electronics Society, Florence, Italy, 2016, pp. 7046-7051, doi: 10.1109/IECON.2016.7793413.

27. **Chong, Adrian, Augenbroe, Godfried y Yan, Da**. *Occupancy data at different spatial resolutions: Building energy performance and model calibration.*. Applied Energy, 2021, Vol. 286. 116492, ISSN 0306-2619.
https://doi.org/10.1016/j.apenergy.2021.116492.(https://www.sciencedirect.com/science/article/pii/S0306261921000532)

28. **Nwankpa, Chigozie; Ijomah, Winifred; Gachagan, Anthony; Marshall, Stephen.** Activation Functions: Comparison of trends in Practice and Research for Deep Learning. 2018.

29. **Maas, Andrew L., Hannun, Awni Y. y Ng, Andrew Y.** Rectifier Nonlinearities Improve Neural Network Acoustic Models. [En línea] 2013.
https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

9 ANEXOS

9.1 Documentación de los sistemas del edificio

*"Tras analizar las diferentes variables del raw data y en concreto las que se corresponden con los analizadores de red del edificio, se ha podido observar cómo los datos de consumo energético que hacen referencia únicamente al edificio van a depender de dos variables: por una parte tenemos el analizador de red 39 de la planta sótano que registra los valores de consumo energético del edificio completo incluyendo los consumos del CPD en MWh, y por otra parte el analizador de red 29 de la planta sótano que registra únicamente el consumo del CPD en kWh. El consumo energético que consideraremos únicamente del edificio LUCIA se calculará como la diferencia entre los datos de los dos analizadores de red, aunque requerirá de un procesado previo de los datos de ambas variables que se mostrará en los siguientes apartados. Consumo Edificio LUCIA(t) = AnlzRed39(t) * 1000 - AnlzRed29(t)".*

9.2 Métricas

- MAE

$$\frac{\sum_{i=1}^{n} |\hat{x}_i - x_i|}{n}$$

- RMSE

$$\sqrt{\frac{\sum_{i=1}^{n} (\hat{x}_i - x_i)^2}{n}}$$

- CV-RMSE

$$\frac{\sqrt{\frac{\sum_{i=1}^{n} (\hat{x}_i - x_i)^2}{n-1}}}{\frac{\sum_{i=i}^{n} x_i}{n}}$$

- R^2

$$1 - \frac{\sum_{i=1}^{n} (\hat{x}_i - x_i)^2}{\sum_{i=1}^{n} (\bar{x}_i - x_i)^2}$$

9.3 KNN

Los algoritmos de los K vecinos más próximos son modelos basados en memoria que no necesitan entrenamiento. La clasificación de una instancia dependerá de las k observaciones con menor distancia a esta. A pesar de esta simplicidad este algoritmo ha sido útil en numerosos problemas de clasificación. La medida de distancia más frecuente para este algoritmo es la euclídea.

El procedimiento de este algoritmo es el siguiente:

1. Seleccionar el número de vecinos (K)
2. Calcular la distancia de los valores del conjunto de entrenamiento
3. Tomar los K vecinos más cercanos según la distancia calculada
4. Entre estos K vecinos, contar el número de observaciones de cada categoría
5. Asignar los nuevos puntos de datos al valor más repetido

9.4 One-Hot encoding

Esta técnica convierte cada posible valor de una variable categórica en una variable binaria. Estas variables tomarán el valor 1 cuando la variable tome ese valor y 0 en caso contrario. Un problema derivado de este método es el aumento de la dimensionalidad que implica.

9.5 Funciones de activación

Estas funciones se eligen estratégicamente para mejorar la capacidad de la red o para simplificarla haciéndola computacionalmente más eficiente. Es importante que esta función sea no lineal en redes de más de una capa ya que en caso contrario las capas adicionales no aportarán ninguna información (28). Las funciones de activación a considerar para este trabajo son:

Sigmoide

Esta función tuvo una gran importancia en un principio ya que es fácilmente derivable y, por tanto, fácil de manipular matemáticamente. Al ser útil para reducir la carga computacional era frecuentemente usado en perceptrones multicapa. Sin embargo, actualmente existen estudios que muestran que no es la más indicada ya que al computar el gradiente por retropropagación esta tiende hacia cero.

La función sigmoide tiene la siguiente expresión:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{24}$$

Tangente Hiperbólica

Se ha demostrado que esta función proporciona mejores resultados que la función sigmoide, sobre todo en redes neuronales multicapa. Al ser una función centrada en cero, ayuda al proceso de retropropagación, aunque puede no solucionar el problema en la evanescencia del gradiente producido por la función sigmoide.

Es una función trigonométrica que es dada por:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{25}$$

Rectificación lineal

También conocida como ReLU. Esta función mejora la ratio de error de la sigmoide. Simplemente elimina todo valor negativo de la entrada. Su expresión viene dada por el siguiente sistema:

$$f(x) = \begin{cases} 0 \; si \; x \leq 0 \\ x \; si \; x > 0 \end{cases} \tag{26}$$

Los rectificadores lineales sufren de un problema. Al ser el gradiente 0 en algunas ocasiones cuando la neurona no se ha activado, sus pesos pueden no ser actualizados y, por tanto, no mejorar en sus predicciones (29). Para ello se puede añadir un parámetro a su fórmula:

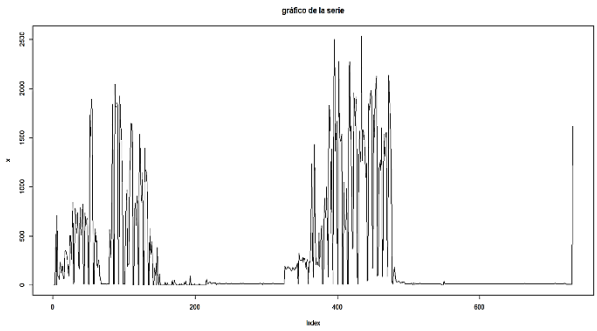
$$f(x) = \begin{cases} \alpha x \; si \; x \leq 0 \\ x \; si \; x > 0 \end{cases} \tag{27}$$

Esta variante se conoce como LeakyReLU.

9.6 Pruebas completas de los modelos ARIMA

9.6.1. Datos completos

Serie inicial

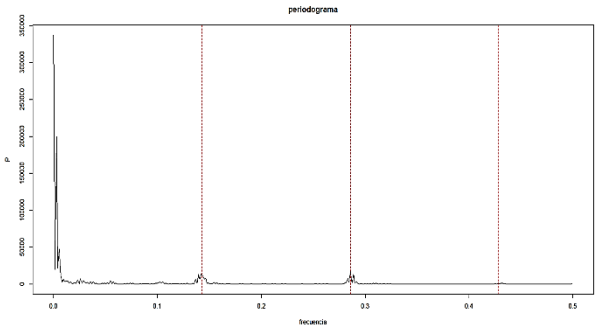


- Media: 290.3815
- Varianza: 295856

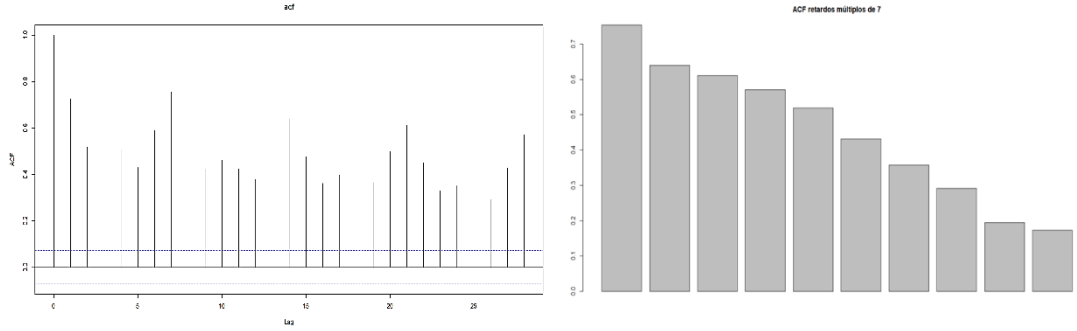
P-valores del test de Mann-Kendall para pendiente:

- Pendiente > 0: 0.9999
- Pendiente = 0: 0.0001029
- Pendiente < 0: 5.144e-05

Periodograma



ACF



PACF

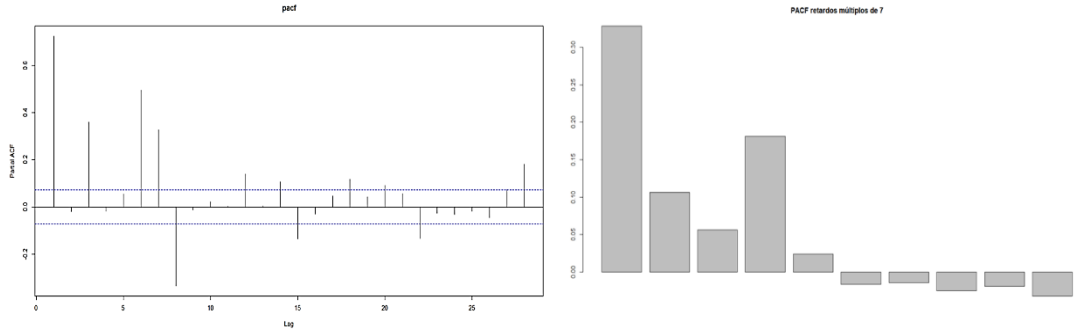
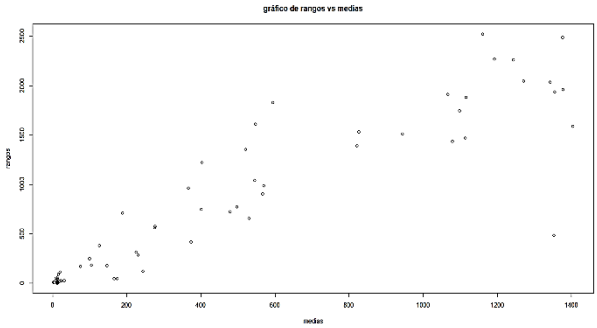
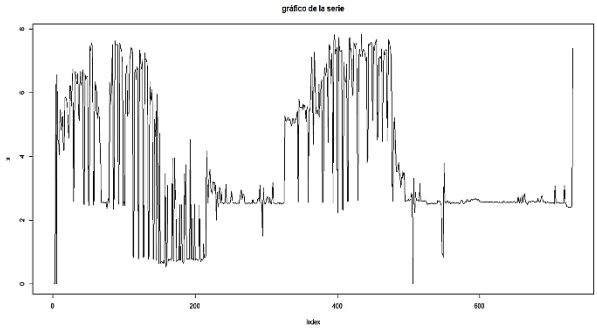


Gráfico de rangos vs medias



Serie transformada



- Media: 3.713533
- Varianza: 4.175921

P-valores del test de Mann-Kendall para pendiente:

- Pendiente > 0: 0.9999
- Pendiente = 0: 0.0001534
- Pendiente < 0: 9.835e-05

Periodograma

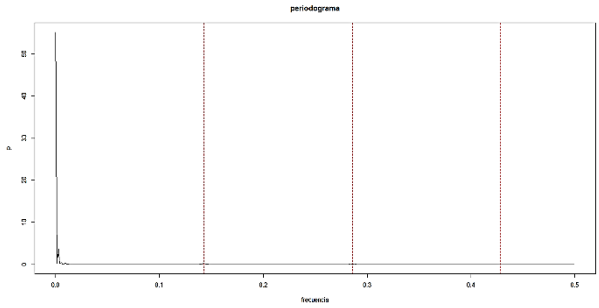
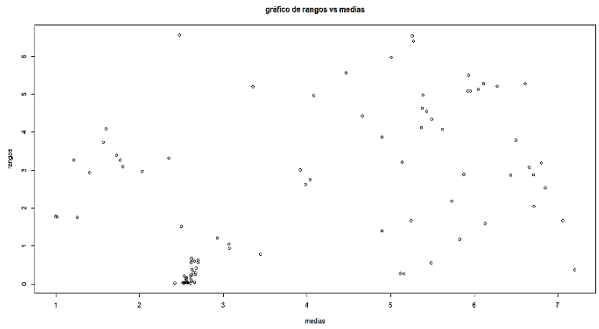
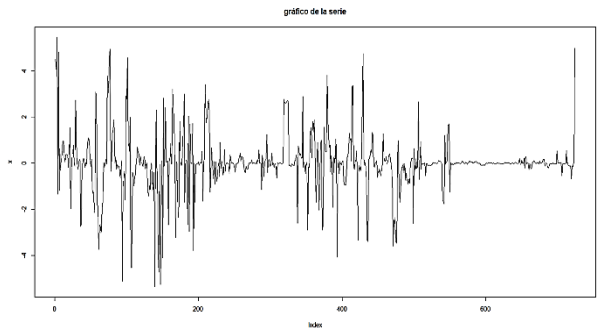


Gráfico de rangos vs medias



Serie diferenciada

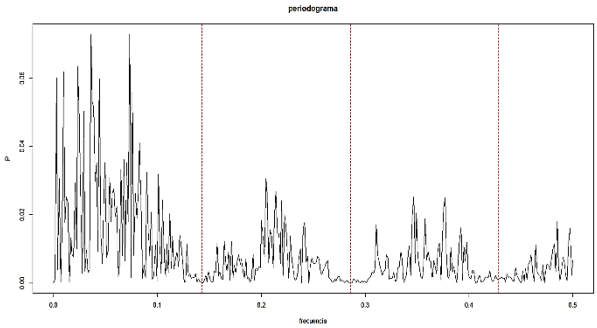


- Media: 0.00631561
- Varianza: 1.577565

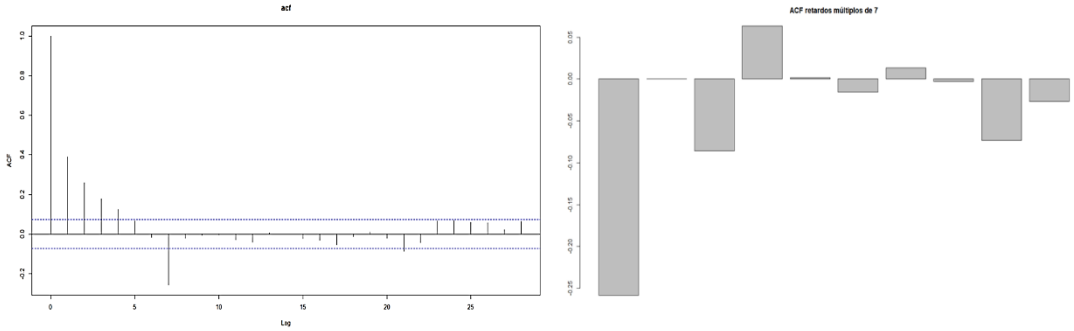
P-valores del test de Mann-Kendall para pendiente:

- Pendiente > 0: 0.8521
- Pendiente $= 0$: 0.2958
- Pendiente < 0: 0.1479

Periodograma



ACF



PACF

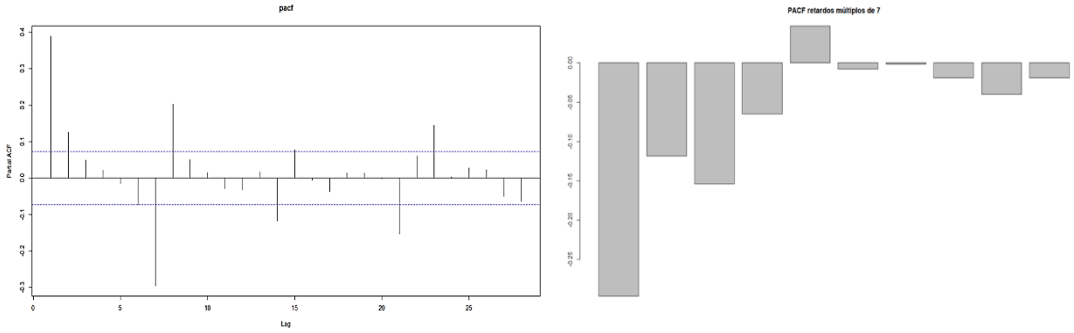
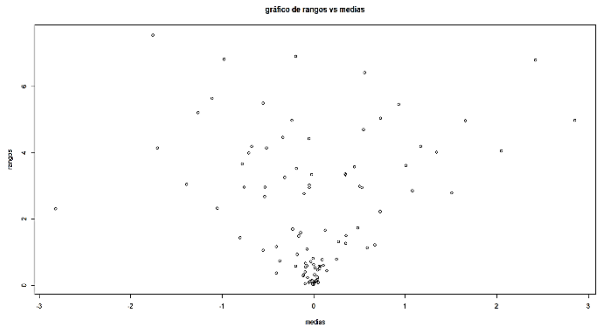
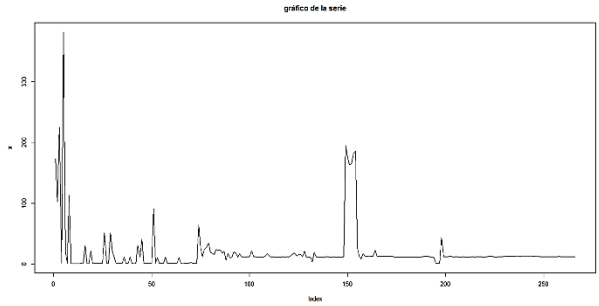


Gráfico de rangos vs medias



9.6.2. Datos del cluster 0

Serie inicial

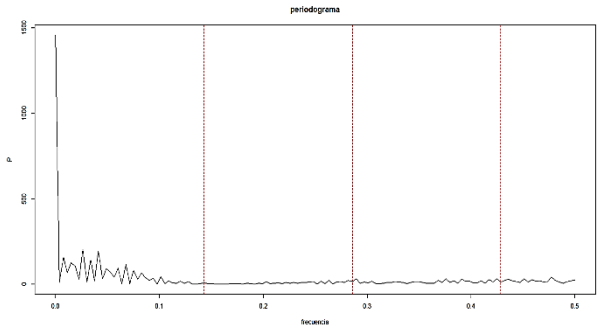


- Media: 19.09398
- Varianza: 1520.999

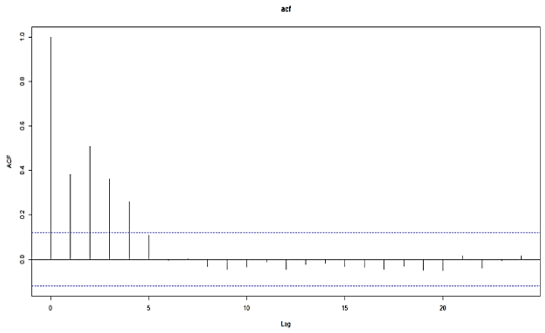
P-valores del test de Mann-Kendall para pendiente:

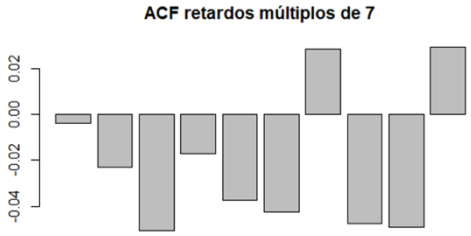
- Pendiente > 0: 1
- Pendiente = 0: 1.515e-08
- Pendiente < 0: 3.03e-08

Periodograma

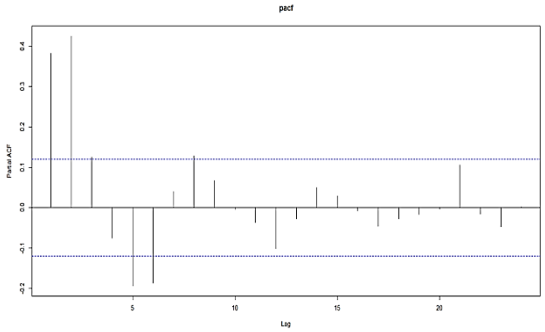


ACF





PACF



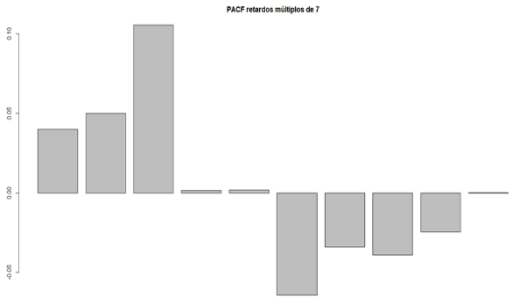
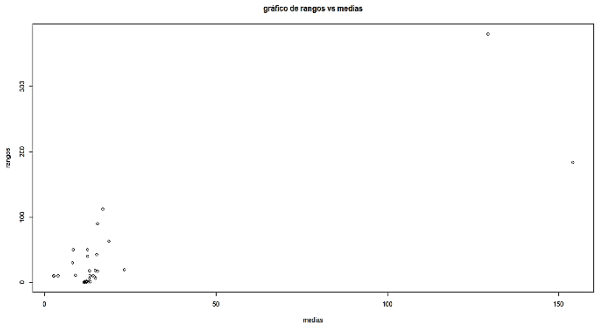
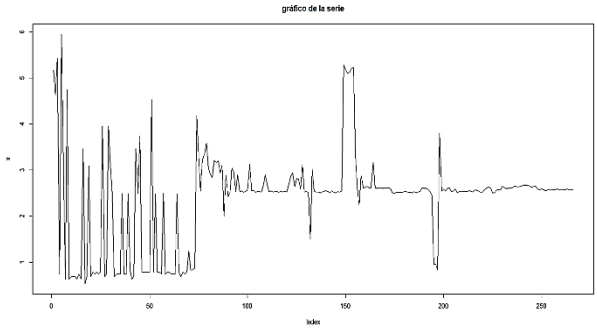


Gráfico de rangos vs medias



Serie transformada



- Media: 2.387729
- Varianza: 1.030185

P-valores del test de Mann-Kendall para pendiente:

- Pendiente > 0: 1.515e-08
- Pendiente = 0: 3.03e-08
- Pendiente < 0: 1

Periodograma

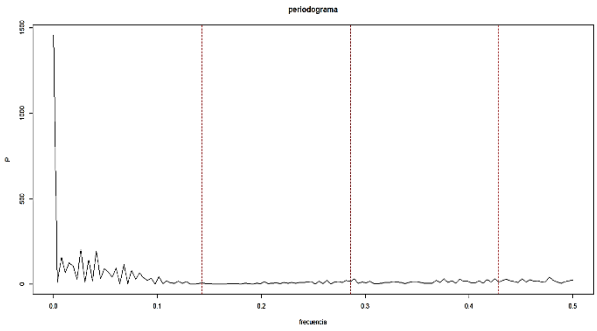
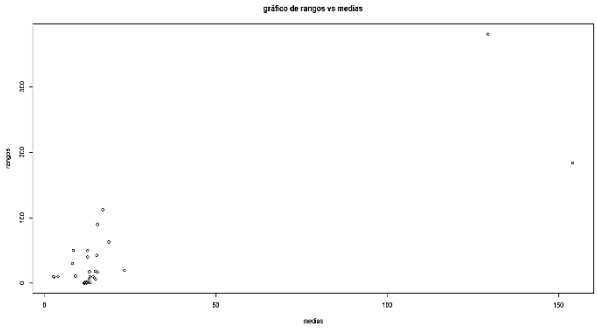
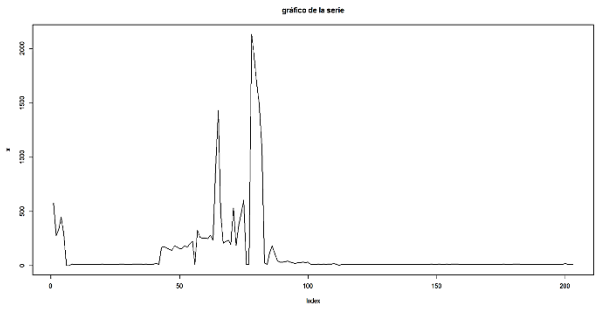


Gráfico de rangos vs medias



9.6.3. Datos del cluster 1

Serie inicial



- Media: 112.5463
- Varianza: 89567.27

P-valores del test de Mann-Kendall para pendiente:

- Pendiente > 0: 0.9999
- Pendiente = 0: 0.0001229
- Pendiente < 0: 6.143e-05

Periodograma

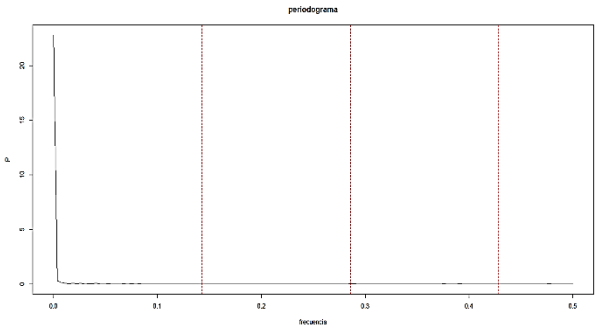
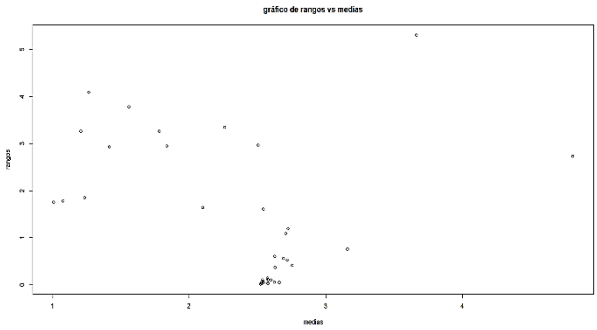
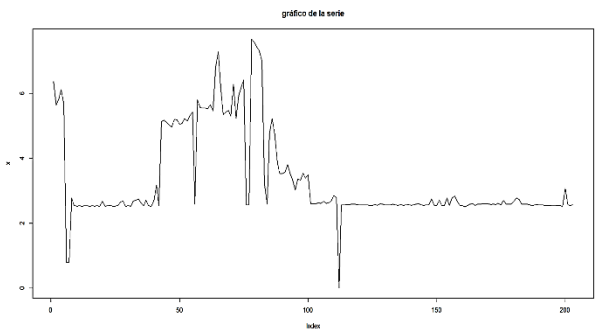


Gráfico de rangos vs medias



Serie transformada



- Media: 3.325035
- Varianza: 1.953429

P-valores del test de Mann-Kendall para pendiente:

- Pendiente > 0: 0.9999
- Pendiente = 0: 0.0001229
- Pendiente < 0: 6.143e-05

Periodograma

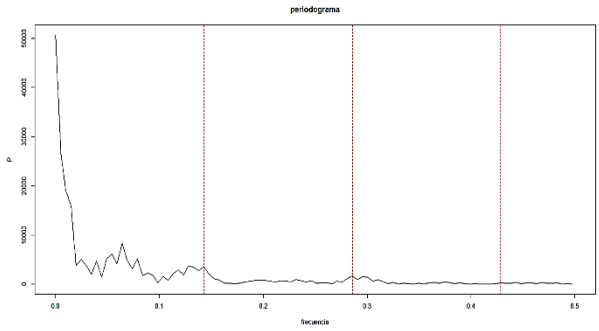
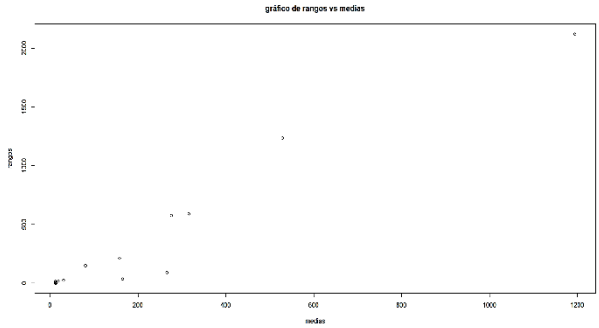
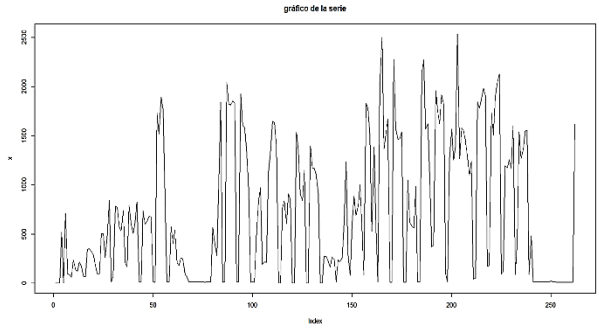


Gráfico de rangos vs medias



9.6.4. Datos del cluster 2

Serie inicial

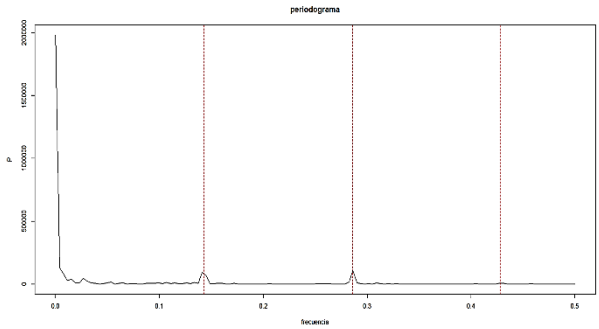


- Media: 703.5992
- Varianza: 485618

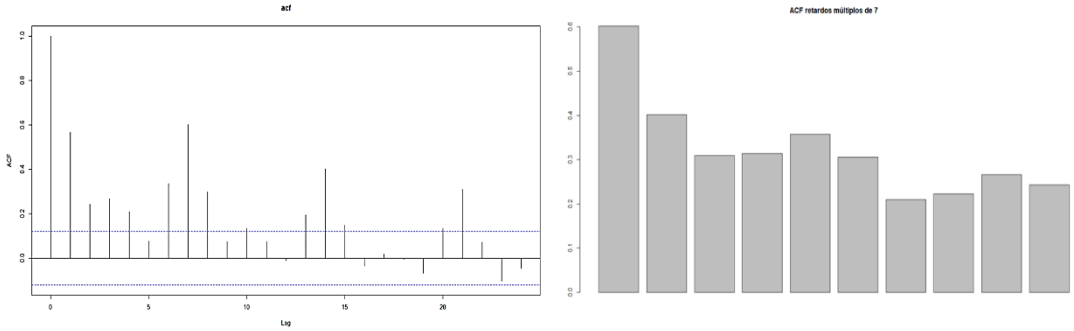
P-valores del test de Mann-Kendall para pendiente:

- Pendiente > 0: 0.003614
- Pendiente = 0: 0.007227
- Pendiente < 0: 0.9964

Periodograma



ACF



PACF

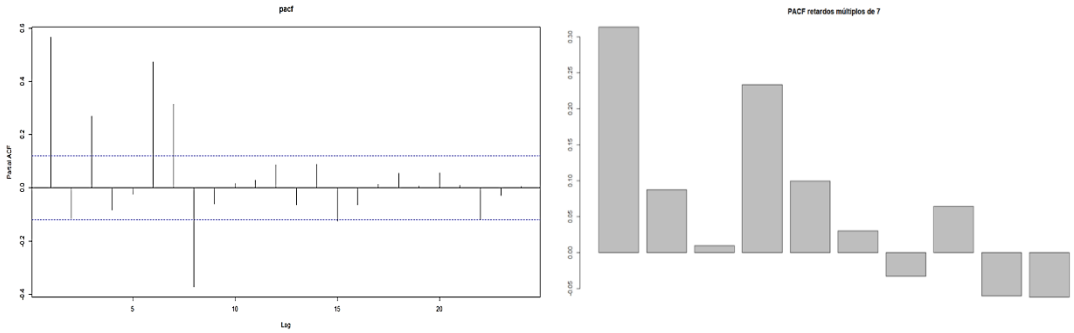
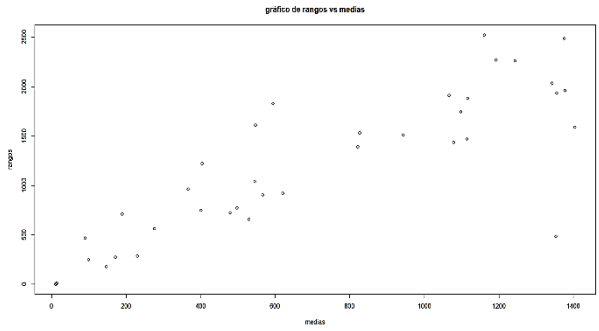
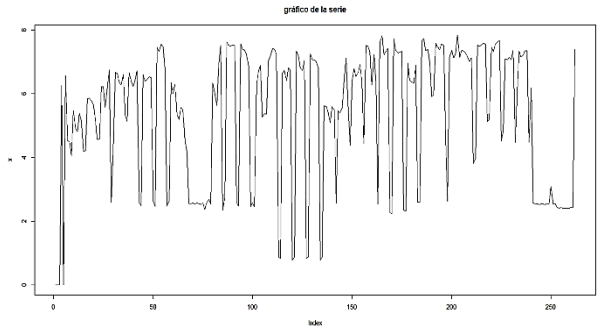


Gráfico de rangos vs medias



Serie transformada



- Media: 5.36059
- Varianza: 4.489946

P-valores del test de Mann-Kendall para pendiente:

- Pendiente $>$ 0: 0.003614
- Pendiente = 0: 0.007227
- Pendiente $<$ 0: 0.9964

Periodograma

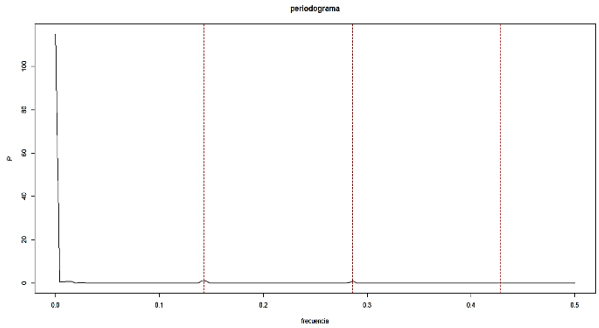
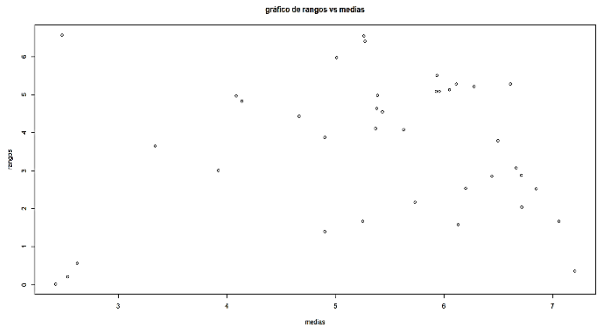
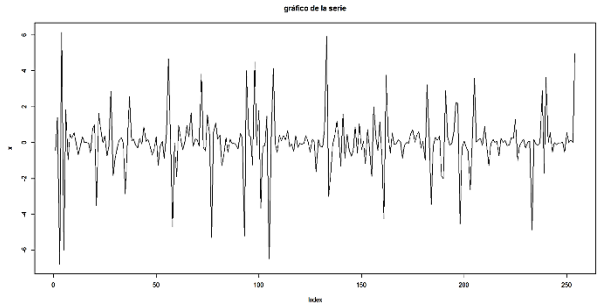


Gráfico de rangos vs medias



Serie diferenciada

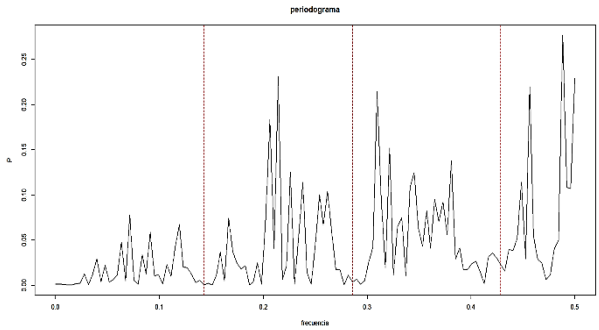


- Media: 0.001818405
- Varianza: 2.781604

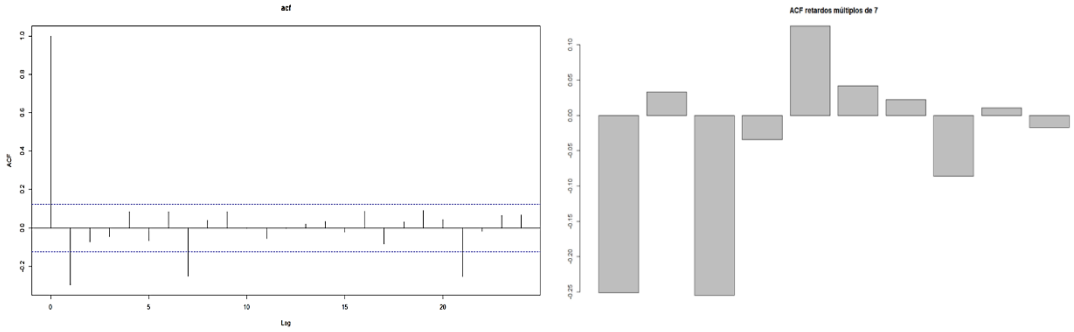
P-valores del test de Mann-Kendall para pendiente:

- Pendiente > 0: 0.5383
- Pendiente = 0: 0.9235
- Pendiente < 0: 0.4617

Periodograma



ACF



PACF

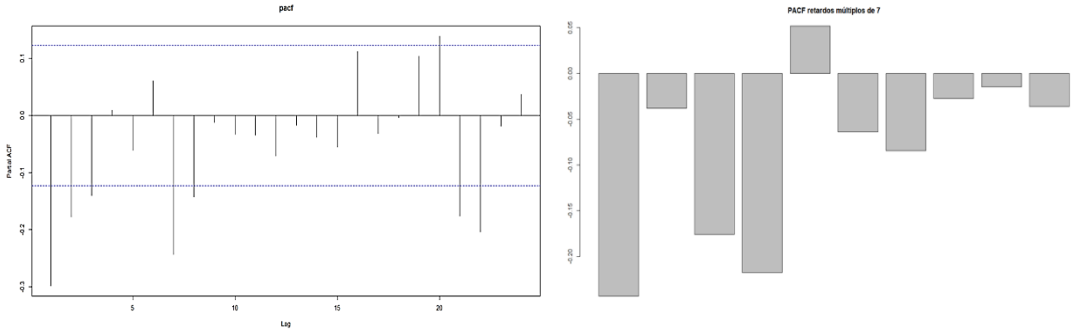
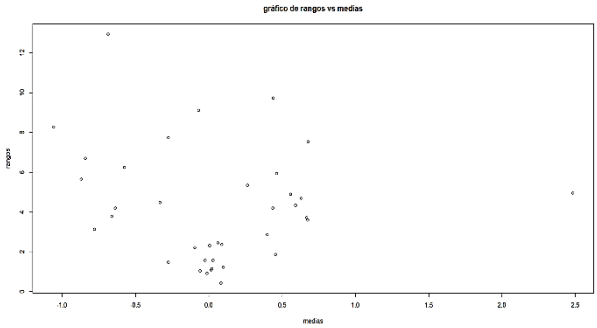


Gráfico de rangos vs medias



9.7 Resultados de los modelos en el conjunto de entrenamiento

9.7.1. Datos por horas

Datos completos

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	3.469907	9.17651	1.003884	0.9040342
LSTM	9.238111	31.15544	3.374903	$1.33728*10^{-7}$
GRU	9.253418	31.14044	3.373277	0.0002690582

Cluster 0

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	0.2477946	1.694824	6.679386	0.7341962
LSTM	0.1226451	1.836597	15.0117	$6.549301*10^{-9}$
GRU	0.1226569	1.836557	15.01137	<0

Cluster 1

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	1.074311	4.443066	1.53756	0.9264198
LSTM	2.692282	16.62752	6.177783	≈0
GRU	2.699816	16.6217	6.175624	0.0008264409

Cluster 2

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	8.496057	14.63523	0.6106112	0.8921041
LSTM	25.04097	51.35945	2.081601	<0
GRU	34.64057	45.07608	1.826935	<0

9.7.2. Datos por días

Datos completos

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	83.3851	159.1728	0.5435495	0.9150284
LSTM	80.12422	187.9477	0.6357999	0.8808725
GRU	111.5281	209.4895	0.7086728	0.8519997

Cluster 0

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	7.239922	16.14565	1.003862	0.6555456
LSTM	5.556839	12.76285	0.7881625	0.8929057
GRU	8.353466	24.76056	1.529074	0.5969193

Cluster 1

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	39.07291	91.98948	0.8193685	0.9050567
LSTM	21.15675	48.11758	0.4378867	0.9741501
GRU	38.285	103.4696	0.941609	0.8804702

Cluster 2

Métricas / Algoritmos	MAE	RMSE	CV-RMSE	R^2
Random Forest	194.3968	253.2788	0.342954	0.867783
LSTM	108.0688	169.4007	0.2229499	0.9409003
GRU	148.5249	216.1372	0.2844603	0.9034675

9.8 Código utilizado

En esta sección se adjuntará parte del código utilizado.

9.8.1. Limpieza de datos del consumo total

```
#librerias----
library(stringr)
library(univOutl)

#datos----

datos.totales.limp <- read.csv2("C:/Users/Javier/Google Drive/universidad/TFGestadistica/datosTotales-procesados.csv")

##analizador 39----
anlz.39 <- as.numeric(datos.totales.limp["AInt.PS.AnlzRed39.En1"][,1])
fechas.39 <- datos.totales.limp["date.150"][,1]

notna.39 <- !is.na(anlz.39)

anlz.39.notna <- anlz.39[notna.39]
fechas.39.notna <- fechas.39[notna.39]

horas.medidas.39 <- str_extract(fechas.39.notna,".*-.*-.*\\s[0-9]{2}") #Se obtiene la hora de cada observacion

horas.39 <- tapply(anlz.39.notna, horas.medidas.39, max) #Obtenemos el maximo de cada hora
horas.39.acotado <- horas.39[which(names(horas.39) == '2019-07-01 00'):length(horas.39)]
energia.39 <- c(0,diff(horas.39.acotado))

##analizador 29 ----
anlz.29 <- as.numeric(datos.totales.limp["AInt.PS.AnlzRed29.AI65"][,1])
fechas.29 <- datos.totales.limp["date.150"][,1]

notna.29 <- !is.na(anlz.29)

anlz.29.notna <- anlz.29[notna.29]
fechas.29.notna <- fechas.29[notna.29]

horas.medidas.29 <- str_extract(fechas.29.notna,".*-.*-.*\\s[0-9]{2}") #Obtencion de la hora de cada observacion

horas.29 <- tapply(anlz.29.notna, horas.medidas.29, max) #Obtenemos el maximo de cada hora
horas.29.acotado <- horas.29[which(names(horas.29) == '2019-07-01 00'):length(horas.29)]
energia.29 <- c(0,diff(horas.29.acotado))

#Inicializacion del dataset final----
horas<-c("00","01","02","03","04","05","06","07","08","09","10","11","12","13","14","15","16","17","18","19","20","21","22","23")
horas.totales <- paste(rep(seq(as.Date("2019-07-01 00"), as.Date("2021-05-21"), by="days"), each=24), horas)
consumo.por.horas <- rep(0,length(horas.totales))
consumo.por.horas <- as.array(consumo.por.horas)
dimnames(consumo.por.horas)<-list(horas.totales)

for(i in dimnames(horas.39.acotado)[[1]]){
```

```
47.   #print(energia.por.horas[i])
48.   consumo.por.horas[i] <- energia.39[i]*1000
49. }
50.
51. for(i in dimnames(horas.29.acotado)[[1]]){
52.   #print(energia.por.horas.2[i])
53.   consumo.por.horas[i] <- consumo.por.horas[i] - energia.29[i]
54. }
55.
56. consumo.por.horas.smooth <- consumo.por.horas
57. for(i in boxB(consumo.por.horas.smooth, k=1.5,
    method='asymmetric', weights=NULL, id=NULL, exclude=NA,
    logt=FALSE)$outliers){
58.   consumo.por.horas.smooth[i] <- (consumo.por.horas.smooth[i-
    1] + consumo.por.horas.smooth[i+1])/2
59. }
60. consumo.por.horas.smooth.no0 <- consumo.por.horas.smooth
61. for(i in which(consumo.por.horas.smooth<0)){
62.   consumo.por.horas.smooth.no0[i] <- 0
63. }
```

9.8.2. Clustering 1NN

```
1.  #librerias----
2.  library(stringr)
3.  library(class)
4.  #funciones----
5.  mode <- function(x) {
6.    return(as.numeric(names(which.max(table(x)))))
7.  }
8.
9.  #...
10.
11. #Conjunto de entrenamiento----
12. train.1nn <- lucia[lucia$semana<=83, c('humedad','temperatura')]
13. cl.1nn <- lucia[lucia$semana<=83, 'cluster']
14. test.1nn <- lucia[lucia$semana>83, c('humedad','temperatura')]
15.
16. #Modelo
17. clus <- knn1(train = train.1nn, test=test.1nn, cl=cl.1nn)
18.
19. #Se aplica la moda para cada semana
20. tapply(clus, lucia[lucia$semana>83, 'semana'], mode)
21.
22.
23. #Conjunto de prueba----
24. lucia.valid <- read.csv2("C:/Users/Javier/Google
    Drive/universidad/TFGestadistica/consumo_refr_test_dias.csv")
25. #Se extraen las marcas de tiempo
26. lucia.valid$diasemana <- rep(c(2:7, 1), ceiling(nrow(lucia.valid)/7))[1:
    nrow(lucia.valid)]
27. lucia.valid$mes <- as.numeric(substring(lucia.valid$fecha,6, 7))
28. #Se obtiene el valor de la semana anterior
29. lucia.valid$anterior <- c(rep(NA, 7),
    lucia.valid$consumo[1:(nrow(lucia.valid)-7)])
30. lucia.valid$semana <- c(rep(1,6), rep(2:52, each=7), rep(53,2))
31.
32. test.1nn.2 <- lucia.valid[, c('humedad','temperatura')]
33. #Modelo
34. clus.2 <- knn1(train = train.1nn, test=test.1nn.2, cl=cl.1nn)
35. tapply(clus.2, lucia.valid[, 'semana'], mode)
```

9.8.3. Gráficos de los datos

```
#graficos----
lucia.valid$fechaP <- as.POSIXct(lucia.valid$fecha, format = "%Y-%m-%d
%H")
ggplot(data=lucia.valid, aes(x=fechaP, y = humedad)) +
  geom_line(color='#5B9BD5') +
  labs(title = 'Humedad por horas') + xlab('')+ ylab('') +
  theme(plot.title = element_text(hjust = .5))


ggplot(data=lucia.valid, aes(x=fechaP, y = temperatura)) +
  geom_line(color='#ff6010') +
  labs(title = 'Temperatura por horas') + xlab('')+ ylab('') +
  theme

ggplot(data= lucia.valid, aes(x=fechaP, y= consumo)) +
  geom_line(color='#4472C4') +
  labs(title = 'Consumo por horas') + xlab('')+ ylab('') +
  theme(plot.title = element_text(hjust = .5))


ggplot(lucia.valid, aes(x=fechaP, y=consumo, color = factor(cluster))) +
  geom_line(aes(group = 1))+
  scale_color_manual(values=c('#0000ff','#00ff00','#ff0000')) +
  scale_x_datetime(date_labels = "%b", date_breaks = '1 month') +
  labs(title = 'Consumo por horas',
color= 'cluster') + xlab('')+ ylab('') +
  theme(plot.title = element_text(hjust = .5))

ggplot(lucia.valid, aes(x=fechaP, y=humedad, color = factor(cluster))) +
  geom_line(aes(group = 1))+
  #scale_color_manual(values=c('#8080ff','#40409f','#20206f')) +
  scale_color_manual(values=c('#aaaaff','#4488ff','#0000ff')) +
  scale_x_datetime(date_labels = "%b", date_breaks = '1 month') +
  labs(title = 'Humedad', color= 'cluster') + xlab('')+ ylab('') +
  theme(plot.title = element_text(hjust = .5))

ggplot(lucia.valid, aes(x=fechaP, y=temperatura,
color = factor(cluster))) +
  geom_line(aes(group = 1))+
  scale_color_manual(values=c('#803000','#ff8000','#ff0000')) +
  scale_x_datetime(date_labels = "%b", date_breaks = '1 month') +
  labs(title = 'Temperatura', color= 'cluster') + xlab('')+ ylab('') +
  theme(plot.title = element_text(hjust = .5))
```

9.8.4. Ejemplo de identificación SARIMA

```
#Lectura de datos----
datos.conjuntos <- read.csv2("C:/Users/Javier/Google
Drive/universidad/TFGestadistica/consumo_refr_abs_cluster_dias.csv")
#Funciones----
library(trend)
rango <-function(x){
  return(diff(range(x)))
}

periodograma.1 <- function(x, xlim=c(0,0.5),printp=F){
  long <- length(x)
  FF <- abs(fft(x)/sqrt(long))^2
```

```
12.   P <- (4/long)*FF[1:(long/2+1)] # Lee los primeros (n/2)+1 valores del
    resultado de FFT
13.   f <- (0:(long/2))/long # Crea armoicos desde 0 hasta 0.5 en pasos de
    1/128
14.   plot(f, P, type="l", xlim=xlim) # Grafico del periodograma
15.   if(printp){
16.     print(p)
17.   }
18. }
19.
20. analisis <- function(x){  #Extrae la informacion y graficos necesarios
    para la identificacion de una serie
21.   print(mean(x))
22.   print(var(x))
23.   plot(x, type='l', main='gráfico de la serie')
24.
25.   # Test de Mann-Kendall
26.   print(mk.test(x, alternative = "greater"))
27.   print(mk.test(x, alternative = "two.sided"))
28.   print(mk.test(x, alternative = "less"))
29.
30.   periodograma.1(x)
31.   abline(v=1:3/7, col='red', lty=2) #Señala posibles armónicos
    significativos
32.
33.   acf(x, main='acf')
34.   acf.larga <- acf(x, 10*7)
35.   barplot(acf.larga$acf[1+7*1:10], main = 'ACF retardos múltiplos de 7')
36.
37.   pacf(x, main='pacf')
38.   pacf.larga <- pacf(x, 10*7)
39.   barplot(pacf.larga$acf[7*1:10],  main = 'PACF retardos múltiplos de
    7')
40.
41.   #Grafico de Rangos VS medias
42.   periodos <- rep(1:length(x)/7, each= 7)[1:length(x)]
43.   rangos <- tapply(x, as.factor(periodos), rango)
44.   medias <- tapply(x, as.factor(periodos), mean)
45.   plot(medias,type='l', main= 'gráfico de medias')
46.   plot(rangos,type='l', main= 'gráfico de rangos')
47.   plot(rangos~medias, main= 'gráfico de rangos vs medias')
48.
49. }
50.
51. #Pruebas----
52. x <- datos.conjuntos[['consumo']]
53. analisis(x)
54.
55. log1x <- log(1+x)
56. analisis(log1x)
57.
58. log1x.1 <- diff(log1x)
59. analisis(log1x.1)
60.
61. log1x.7 <- diff(log1x, 7)
62. analisis(log1x.7)
63.
64. log1x.1.7 <- diff(log1x.7)
65. analisis(log1x.1.7)
```

9.8.5. Estimación y validación de modelos SARIMA

Código SAS

```
data vent;
        input energia @@;
        *hour = intnx('dthour', '02jun19:00:00:00'dt, _n_-1);
        *format         hour datetime12.;
        logaritmo = log1px(energia);
cards;
*...;

proc arima data =vent;
identify var=logaritmo(1,7); run;
estimate  p=(7) q=(1,7) method=ml noconstant; run;
estimate  q=(1,7) method=ml noconstant; run;
estimate  p=(7) q=(1) method=ml noconstant; run;
```

9.8.6. Ejemplo de Random Forest

```
#Librerias----
library(randomForest)
library(ggplot2)
library(stringr)

#Funciones ----
importancia <- function(x){
  # Extrae la importancia de las variables del modelo ajustado
  Importancia_extraida <- as.data.frame(importance(x))
  Importancia_extraida$Var.Names <- row.names(Importancia_extraida)

  ggplot(Importancia_extraida, aes(x=Var.Names, y=`%IncMSE`)) +
    geom_segment( aes(x=Var.Names, xend=Var.Names, y=0, yend=`%IncMSE`),
  color="skyblue") +
    geom_point(aes(size = IncNodePurity), color="blue", alpha=0.6) +
    theme_light() +
    coord_flip() +
    theme(
      legend.position="bottom",
      panel.grid.major.y = element_blank(),
      panel.border = element_blank(),
      axis.ticks.y = element_blank()
    )
}

metricas<- function(reales, predichos, fechas){
  #Extrae la informacion necesaria para evaluar el modelo

  resid <- predichos-reales

  print('MAE')
  print(mean(abs(resid)))#MAE
  print('RMSE')
  print(sqrt(mean((resid)^2)))#RMSE
  print('CV-RMSE')
  print(sqrt(sum((resid)^2)/(length(reales)-1))/mean(reales)) #CV-RMSE
  print('R2')
  print(1-var(resid)/var(reales))

  #plot(reales, type='l', col='blue')
  #lines(predichos, type='l', col='orange')

  Group <- c(0, cumsum(diff(fechas > 1)))+1

  p <-ggplot()  +
    geom_line(data = data.frame(reales, fechas, Group), aes(x=fechas,
  y=reales, color = 'reales') ) +
    facet_grid(~ Group, scales = "free_x") +
```

```
47.     geom_line(data = data.frame(predichos, fechas, Group), aes(x=fechas,
    y=predichos, color='predichos') ) +
48.     facet_grid(~ Group, scales = "free_x") +
49.     scale_color_manual(values=c('orange', '#4472C4')) +
50.     scale_x_datetime(date_labels = "%b", date_breaks = '1 month') +
51.     labs(title = 'Reales vs Predichos',
    color = '') + xlab('')+ ylab('') +
52.     theme(plot.title = element_text(hjust = .5))
53.
54.   print(p)
55.
56.   p <-ggplot()  +
57.     geom_line(data = data.frame(reales, fechas, Group), aes(x=fechas,
    y=reales, color = 'reales') ) +
58.     geom_line(data = data.frame(predichos, fechas, Group), aes(x=fechas,
    y=predichos, color='predichos') ) +
59.     scale_color_manual(values=c('orange', '#4472C4')) +
60.     scale_x_datetime(date_labels = "%b", date_breaks = '1 month') +
61.     labs(title = 'Reales vs Predichos',
    color = '') + xlab('')+ ylab('') +
62.     theme(plot.title = element_text(hjust = .5))
63.
64.   print(p)
65.
66.   plot(resid, type='l', col='brown', main='grafico de residuos')
67. }
68.
69. #Modelos----
70. rf.fit <- randomForest(consumo ~
    humedad + temperatura + diasemana + mes  + hora + anterior, data=lucia[-
    (1:(7*24)),], ntree=200,
71.                          importance=TRUE)
72.
73. rf.fit
74. importancia(rf.fit)
75.
76. predichos <- predict(rf.fit, lucia[-
    (1:(7*24)),c('humedad', 'temperatura', 'diasemana', 'mes', 'anterior', '
    hora')])
77. reales <- lucia$consumo[-(1:(7*24))]
78. fechas <- lucia$fechaP[-(1:(7*24))]
79. metricas(reales, predichos, fechas)
```

9.8.7. Ejemplo de Redes Neuronales Recurrentes

```
1.  #Librerias----
2.  library(stringr)
3.  library(dplyr)
4.  library(keras)
5.  library(tensorflow)
6.
7.  #Preparacion de datos----
8.  actores_escalado <- c(mean(lucia[lucia$cluster==2,]$consumo), sd(lucia[l
    ucia$cluster==2,]$consumo))
9.
10. entrenamiento.escal <- lucia[lucia$cluster==2,] %>%
11.   dplyr::select(consumo) %>%
12.   dplyr::mutate(consumo = (consumo - factores_escalado[1]) / factores_es
    calado[2])
13. entrenamiento.escal <- as.matrix(entrenamiento.escal)
14.
15. retardos <-7
16. predic <-7
17.
```

```
18.
19. #obtenemos los valores anteriores
20. x.train <- t(sapply(
21.   1:(length(entrenamiento.escal) - retardos - predic + 1),
22.   function(x) entrenamiento.escal[x:(x + retardos - 1), 1]
23. ))
24.
25. x.train.arr <- array(
26.   data = as.numeric(unlist(x.train)),
27.   dim = c(
28.     nrow(x.train),
29.     retardos,
30.     1
31.   )
32. )
33.
34. y.train <- t(sapply(
35.   (1 + retardos):(length(entrenamiento.escal) - predic + 1),
36.   function(x) entrenamiento.escal[x:(x + predic - 1)]
37. ))
38.
39. y.train.arr <- array(
40.   data = as.numeric(unlist(y.train)),
41.   dim = c(
42.     nrow(y.train),
43.     predic,
44.     1
45.   )
46. )
47.
48.
49. x.test <- lucia[lucia$cluster==2,]$consumo[(nrow(entrenamiento.escal) -
    predic + 1):nrow(entrenamiento.escal)]
50. x.test.escal <- (x.test - factores_escalado[1]) / factores_escalado[2]
51.
52.
53. x.pred.arr <- array(
54.   data = x.test.escal,
55.   dim = c(
56.     1,
57.     retardos,
58.     1
59.   )
60. )
61.
62. variables.dias <- matrix(nrow=dim(lucia[lucia$cluster==2,])[1], ncol = 7
    )
63. for(i in 1:7){
64.   variables.dias[,i] <- as.numeric(rep(c(6,7,1:5), nrow(lucia[lucia$clus
    ter==2,])/7)==i)
65. }
66.
67. variables.meses <- matrix(nrow=dim(lucia[lucia$cluster==2,])[1], ncol =
    12)
68. for(i in 1:12){
69.   variables.meses[,i] <- as.numeric(as.numeric(substring(lucia[lucia$clu
    ster==2,]$fecha,6, 7))==i)
70. }
71.
72. factores.escalados.hum <- list(
73.   mean = mean(lucia[lucia$cluster==2,]$humedad),
74.   sd = sd(lucia[lucia$cluster==2,]$humedad)
75. )
76. humedad.escalada <- (lucia[lucia$cluster==2,]$humedad - factores.escalad
    os.hum$mean)/factores.escalados.hum$sd
```

```
77. humedad.escalada.pred <- (lucia[lucia$cluster==2,]$humedad[(length(lucia
    [lucia$cluster==2,]$humedad) -
    retardos): length(lucia[lucia$cluster==2,]$humedad)] - factores.escalado
    s.hum$mean) /factores.escalados.hum$sd
78.
79. factores.escalados.temp <- list(
80.   mean = mean(lucia[lucia$cluster==2,]$temperatura),
81.   sd = sd(lucia[lucia$cluster==2,]$temperatura)
82. )
83. temperatura.escalada <- (lucia[lucia$cluster==2,]$temperatura - factores
    .escalados.temp$mean)/factores.escalados.temp$sd
84. temperatura.escalada.pred <- (lucia[lucia$cluster==2,]$temperatura[(leng
    th(lucia[lucia$cluster==2,]$temperatura) -
    retardos): length(lucia[lucia$cluster==2,]$temperatura)] - factores.esca
    lados.temp$mean) /factores.escalados.temp$sd
85.
86.
87. #Se unen los datos escalados en una matriz
88. x.train.reg <- cbind(entrenamiento.escal, humedad.escalada,
    temperatura.escalada, variables.dias, variables.meses)
89. x.train.data.reg <- list()
90.
91. #Se obtienen las distintas secuencias dentro de los datos de
    entrenamiento
92. for (i in 1:ncol(x.train.reg)) {
93.   x.train.data.reg[[i]] <- t(sapply(
94.     1:(length(x.train.reg[, i]) - retardos - predic + 1),
95.     function(x) x.train.reg[x:(x + retardos - 1), i]
96.   ))
97. }
98.
99. #Se prepara la estructura para el entrenamiento
100.      x.train.reg.arr <- array(
101.        data = as.numeric(unlist(x.train.data.reg)),
102.        dim = c(
103.          nrow(x.train.data.reg[[1]]),
104.          retardos,
105.          22
106.        )
107.      )
108.
109.      #Modelo----
110.       lstm_reg <- keras_model_sequential()
111.
112.       lstm_reg %>%
113.         layer_lstm(units = 64, # size of the layer
114.                    batch_input_shape = c(1, retardos, 22), # batch
    size, timesteps, features
115.                    return_sequences = TRUE,
116.                    stateful = TRUE) %>%
117.         # fraction of the units to drop for the linear transformation
    of the inputs
118.         layer_dropout(rate = 0.5) %>%
119.         layer_lstm(units = 32,
120.                    return_sequences = TRUE,
121.                    stateful = TRUE) %>%
122.         layer_dropout(rate = 0.5) %>%
123.         time_distributed(keras::layer_dense(units = 1))
124.
125.       lstm_reg %>%
126.         compile(loss = "mae", optimizer = "adam",
    metrics =  metric_root_mean_squared_error())
127.       summary(lstm_reg)
128.
129.       lstm_reg %>% fit(
```

```
130.          x = x.train.reg.arr,
131.          y = y.train.reg.arr,
132.          batch_size = 1,
133.          epochs = 64,
134.          verbose = 1,
135.          shuffle = FALSE
136.        )
137.        save_model_tf(lstm_reg, "C:/Users/Javier/Google
    Drive/universidad/TFGestadistica/modelo_dias_tiempo_cluster2")
138.        #lstm_reg <- load_model_tf("C:/Users/Javier/Google
    Drive/universidad/TFGestadistica/modelo_dias_tiempo_cluster2")
139.
140.      #Prediccion sobre el conjunto de entrenamiento----
141.      lstm.reg.fore.train <- lstm_reg %>%
142.        predict(x.train.reg.arr, batch_size = 1) %>%
143.        .[, , 1]
144.
145.      lstm.reg.fore.train <- lstm.reg.fore.train * factores_escalado[2]
     + factores_escalado[1]
146.
147.      ##prediccion del conjunto de prueba----
148.      lstm.reg.fore.val <- lstm_reg_32 %>%
149.        predict(x.val.arr, batch_size = 1) %>%
150.        .[, , 1]
151.
152.      lstm.reg.fore.val <- lstm.reg.fore.val * factores_escalado[2] + f
    actores_escalado[1]
```