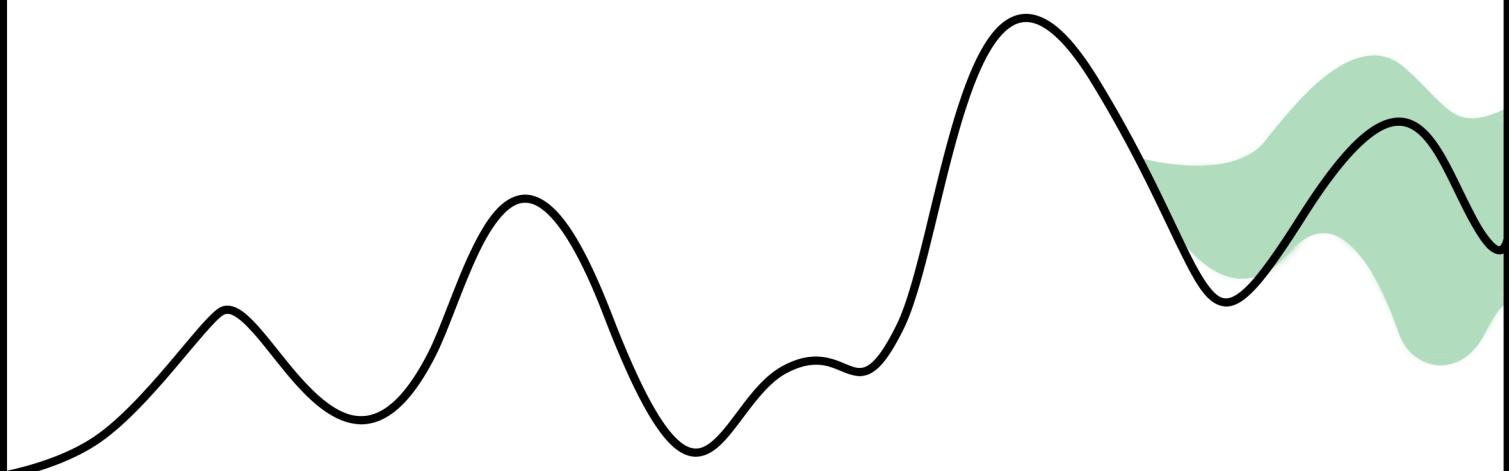


Comparación del desempeño de modelos estadísticos tradicionales, de aprendizaje automático y aprendizaje profundo para la predicción de series temporales

Tesina de grado



Alumno: Roncaglia Andrés Iván

Directora: Mag. Méndez Fernanda

Carrera: Licenciatura en Estadística



UNR Universidad
Nacional de Rosario

Agradecimientos

Resumen

La predicción de series temporales desempeña un rol crucial en contextos como la salud, la economía, la energía y la gestión de recursos, donde anticipar el comportamiento futuro de una variable resulta fundamental para la toma de decisiones. Esta tesis compara el desempeño de distintos enfoques para el pronóstico de series temporales, incluyendo modelos estadísticos tradicionales (ARIMA, SARIMA), algoritmos de aprendizaje automático (XGBoost, LightGBM), redes neuronales recurrentes (LSTM) y modelos fundacionales preentrenados basados en transformadores (TimeGPT y Chronos).

La evaluación se llevó a cabo sobre tres series reales representativas, con diferentes estructuras temporales: (i) número mensual de atenciones por patologías respiratorias en un hospital pediátrico, (ii) empleo privado en el sector educativo, y (iii) temperatura horaria durante el mes de marzo. Para cada caso, se implementaron y ajustaron los modelos mencionados, comparando sus resultados mediante métricas puntuales (MAPE) y probabilísticas (*Interval Score*), así como tiempos de cómputo y facilidad de implementación.

Los resultados muestran que no existe un modelo universalmente superior: mientras los modelos estadísticos ofrecen interpretabilidad y precisión en contextos con fuerte estacionalidad, los algoritmos de boosting presentan buena capacidad predictiva con bajo costo computacional. Las redes LSTM, aunque potentes, requieren entrenamiento cuidadoso y muestran sensibilidad al sobreajuste. Por su parte, los modelos fundacionales ofrecen ventajas en automatización y escalabilidad, aunque su comportamiento varía según la naturaleza de la serie y su estructura interna no siempre es accesible.

Este trabajo aporta evidencia empírica y conceptual para una selección informada de modelos de pronóstico en función del contexto, destacando el potencial de las herramientas recientes como TimeGPT y Chronos, así como la importancia de incorporar métricas probabilísticas y técnicas de conformal prediction para mejorar la estimación de la incertidumbre.

Palabras clave: series temporales, predicción, *conformal predictions*, aprendizaje automático, redes neuronales, *transformers*, TimeGPT, *interval score*, Chronos.

Índice

1. Introducción	1
2. Objetivos	2
2.1 Objetivo general	2
2.2 Objetivos específicos	2
3. Metodología	3
3.1 Conceptos básicos de series de tiempo	3
3.2 Modelos estadísticos tradicionales para series temporales	3
3.2.1 SARIMA	3
3.3 Modelos de aprendizaje automático	6
3.3.1 Introducción a árboles de decisión y ensamblado	6
3.3.2 Diferencias entre XGBoost y LightGBM	9
3.3.3 Intervalos de confianza en algoritmos de aprendizaje automático	10
3.4 Modelos de aprendizaje profundo	11
3.4.1 Introducción a redes neuronales	11
3.4.2 <i>Long Short Term Memory</i> (LSTM)	12
3.4.3 Modelos transformadores	15
3.4.4 Diferencias entre TimeGPT y Chronos	19
3.5 Métricas de evaluación	20
3.6 Selección de parámetros y validación del modelo	21
4. Aplicación	22
4.1 Series a utilizar	22
4.2 Ajuste y evaluación de modelos	24
4.2.1 ARIMA	24
4.2.2 Modelos de aprendizaje automático	30
4.2.3 Redes neuronales (LSTM)	33
4.2.4 Modelos fundacionales (TimeGPT y Chronos)	35
4.3 Comparación de resultados y análisis final	37
5. Conclusiones	39
6. Mejoras y extensiones a la investigación	40
7. Bibliografía	41
8. Anexo	43
8.1 Gráficos estacionales	43
8.2 Salidas de modelos arima	43
8.3 Comprobación de supuestos de modelos arima	49
NOTAS	51

1. Introducción

La predicción de valores futuros en series de tiempo es una herramienta clave en múltiples ámbitos, tales como la economía, el comercio, la salud, la energía y el medio ambiente. En estos contextos, anticipar el comportamiento de una variable permite mejorar la planificación, asignar recursos de forma más eficiente y reducir la incertidumbre.

Actualmente, la ciencia de datos se encuentra en una etapa de constante expansión e innovación, impulsada por la gran cantidad de datos generados diariamente, por lo que en un contexto creciente de complejidad y exigencia temporal, resulta conveniente contar con herramientas que faciliten y acorten los tiempos de trabajo. Si bien los métodos más conocidos para trabajar series de tiempo son precisos, los modelos tradicionales como ARIMA son difíciles de automatizar y requieren de amplios conocimientos para encontrar un buen ajuste, mientras que los algoritmos de aprendizaje automatizado que se utilizan actualmente pueden demandar largos tiempos de entrenamiento y un gran coste computacional. Frente a estas limitaciones, en los últimos años se han desarrollado modelos capaces de seleccionar de forma automática el mejor ajuste para una serie temporal dada, sin requerir entrenamiento previo ni conocimientos especializados en análisis de series de tiempo. Estos son los denominados modelos fundacionales preentrenados, tales como TimeGPT o Chronos.

Sin embargo, aún persisten interrogantes sobre el desempeño de estos nuevos modelos y la falta de acceso al código fuente de algunos de estos limita la posibilidad de auditar sus resultados o replicar su implementación. Por ello, esta tesina propone realizar una comparación sistemática de modelos de pronóstico para series de tiempo, abordando tres enfoques metodológicos: modelos estadísticos tradicionales, algoritmos de *machine learning* y modelos de aprendizaje profundo. El objetivo es evaluar su desempeño en distintos contextos, utilizando métricas como el porcentaje del error absoluto medio (MAPE) y el *Interval Score*, con el fin de analizar ventajas, limitaciones y potenciales usos de cada uno.

Este análisis busca aportar una mirada crítica e informada sobre el uso de nuevas tecnologías en la predicción de series de tiempo, contribuyendo a la toma de decisiones metodológicas más sólidas desde una perspectiva estadística.

2. Objetivos

2.1 Objetivo general

El objetivo de esta tesina es, en primer lugar, comparar la precisión, eficiencia y facilidad de pronosticar series de tiempo con distintos modelos, incluyendo enfoques estadísticos clásicos, algoritmos de *machine learning* y modelos de *deep learning*, analizando al mismo tiempo sus ventajas, limitaciones y condiciones de uso más apropiadas.

2.2 Objetivos específicos

- Implementar modelos clásicos de series de tiempo, como ARIMA y SARIMA, explicando y garantizando el cumplimiento de los fundamentos teóricos y supuestos que los sostienen.
- Aplicar modelos de aprendizaje automático supervisado, como XGBoost y LightGBM, explorando distintas configuraciones para garantizar el mejor ajuste.
- Desarrollar modelos de aprendizaje profundo, en particular redes LSTM, dando introducción a las redes neuronales y modelos de pronóstico más complejos.
- Realizar pronósticos con modelos fundacionales(TimeGPT,Chronos) y comprender su funcionamiento.
- Definir y aplicar métricas de evaluación (MAPE, *Interval Score*) para comparar el rendimiento de todos los modelos bajo un mismo conjunto de datos.
- Reflexionar valorativamente sobre los criterios de selección de modelos en función del contexto de aplicación, la complejidad computacional y la interpretabilidad de los resultados.

3. Metodología

El enfoque metodológico adoptado en esta tesina consiste en comparar el desempeño de distintos modelos de pronóstico aplicados a series temporales. Para ello, se seleccionan modelos representativos de tres enfoques principales: modelos estadísticos tradicionales, algoritmos de aprendizaje automático (*machine learning*) y modelos de aprendizaje profundo (*deep learning*).

El análisis se estructura en tres componentes fundamentales: una descripción conceptual de los modelos, su implementación práctica sobre series con diferentes características, y una evaluación cuantitativa comparativa a través de métricas de error.

3.1 Conceptos básicos de series de tiempo

Se denomina serie de tiempo a un conjunto de observaciones $\{z_1, z_2, \dots, z_t, \dots, z_n\}$ cuantitativas ordenadas en el tiempo, usualmente de forma equidistante, sobre una variable de interés. El análisis de series de tiempo tiene como objetivo sintetizar y extraer información estadística relevante, tanto para interpretar el comportamiento histórico de la variable como para generar pronósticos $\{z_{n+1}, \dots, z_{n+l}, \dots, z_{n+h}\}$.

Dado que las series temporales pueden exhibir diversos patrones subyacentes, resulta útil descomponerlas en componentes separadas, cada una de las cuales representa una característica estructural específica del comportamiento de la serie.

- Estacionalidad: corresponde a las fluctuaciones periódicas que se repiten a intervalos regulares de tiempo. Un ejemplo típico es la temperatura, que tiende a disminuir en invierno y aumentar en verano, repitiendo este patrón anualmente.
- Tendencia (o tendencia-ciclo): refleja la evolución a largo plazo de la media de la serie, asociada a procesos de crecimiento o decrecimiento sostenido. Por ejemplo, la población mundial exhibe una tendencia creciente a lo largo del tiempo.
- Residuos: representa las variaciones no sistemáticas que no pueden ser explicadas por la tendencia ni la estacionalidad. Estas fluctuaciones, que suelen deberse a eventos impredecibles o factores exógenos, se asumen como aleatorias.

3.2 Modelos estadísticos tradicionales para series temporales

Son llamados modelos estadísticos tradicionales a aquellos que surgen antes del auge del *machine learning* y los modelos de aprendizaje profundo. Son caracterizados por sus fuertes fundamentos estadísticos y su capacidad en capturar dependencias temporales en los datos.

3.2.1 SARIMA

Se dice que una serie es débilmente estacionaria si la media y la variancia se mantienen constantes en el tiempo y la correlación entre distintas observaciones solo depende de la distancia en el tiempo entre estas. Por comodidad, cuando se mencione estacionariedad se estará haciendo referencia al cumplimiento de estas propiedades.

Los modelos *ARIMA* (*AutoRegressive Integrated Moving Average*) son unos de los modelos de pronóstico tradicionales mejor establecidos. Son una generalización de los modelos autoregresivos (AR), que suponen que las observaciones futuras son función de las observaciones

pasadas, y los modelos promedio móvil (MA), que pronostican las observaciones como funciones de los errores de observaciones pasadas. Además, estos modelos pueden adaptarse a series no estacionarias mediante la aplicación de diferenciaciones de orden d , las cuales implican restar a cada observación el valor registrado d periodos anteriores.

Formalmente un modelo $ARIMA(p, d, q)$ se define como:

$$\psi_p(B)(1 - B)^d z_t = \theta_0 + \theta_q(B)\alpha_t \quad (1)$$

Donde z_t es la observación t -ésima, $\psi_p(B)$ y $\theta_q(B)$ son funciones de los rezagos (B), correspondientes a la parte autoregresiva y promedio móvil respectivamente, d es el grado de diferenciación y α_t es el error de la t -ésima observación.

Se debe tener en cuenta estos aspectos importantes:

- Se dice que una serie es invertible si se puede escribir cada observación como una función de las observaciones pasadas más un error aleatorio. Por definición, todo modelo AR es invertible.
- Por definición, todo modelo MA es estacionario.
- $\psi_p(B) = 1 - \psi_1 B - \psi_2 B^2 - \dots - \psi_p B^p$ es el polinomio característico de la componente AR y $\theta_q(B) = 1 - \theta_1 B - \theta_2 B^2 - \dots - \theta_q B^q$ de la componente MA. Si las raíces de los polinomios característicos caen fuera del círculo unitario, entonces un proceso AR se puede escribir de forma MA y es estacionario, y a su vez un proceso MA se puede escribir de forma AR y es invertible.
- Un proceso $ARIMA$ es estacionario e invertible si su componente AR y MA lo son respectivamente.

Sin embargo este tipo de modelos no tienen en cuenta la posible estacionalidad que puede tener una serie, es por esto que se introducen los modelos $SARIMA(p, d, q)(P, D, Q)_s$ que agregan componentes AR, MA y diferenciaciones a la parte estacional de la serie con período s .

Se denomina función de autocorrelación a la función de los rezagos, entendiendo por rezago a la distancia ordinal entre dos observaciones, que grafica la autocorrelación entre pares de observaciones. Es decir que para cada valor k se tiene la correlación entre todos los pares de observaciones a k observaciones de distancia. En su lugar, la función de autocorrelación parcial calcula la correlación condicional de los pares de observaciones, removiendo la dependencia lineal de estas observaciones con las que se encuentran entre estas. Estas funciones son necesarias para poder identificar los modelos $SARIMA$ y se definen como:

$$\rho_k = \text{Corr}(z_t, z_{t+k}) = \frac{\text{Cov}(z_t, z_{t+k})}{\sqrt{\text{Var}(z_t) \cdot \text{Var}(z_{t+k})}} \quad (2)$$

y

$$\phi_{kk} = \text{Corr}(z_t, z_{t+k} | z_{t+1}, \dots, z_{t+k-1}) \quad (3)$$

Los modelos AR se caracterizan por tener autocorrelaciones significativas que decaen lentamente y autocorrelaciones parciales significativas únicas. Los modelos MA se comportan de forma inversa, tienen autocorrelaciones significativas únicas y autocorrelaciones parciales que decaen progresivamente. Estos comportamientos también se evidencian en las componentes estacionales, presentándose en los rezagos estacionales.

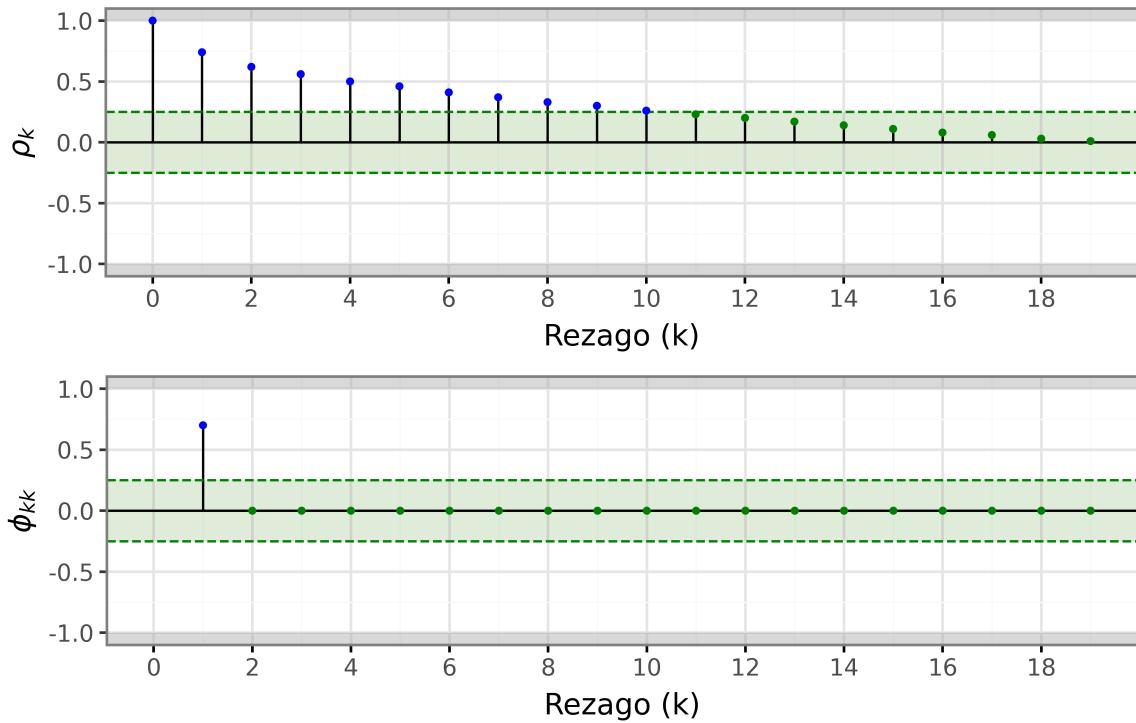


Figura 1: Ejemplo de las autocorrelaciones de un proceso $AR(1)$.

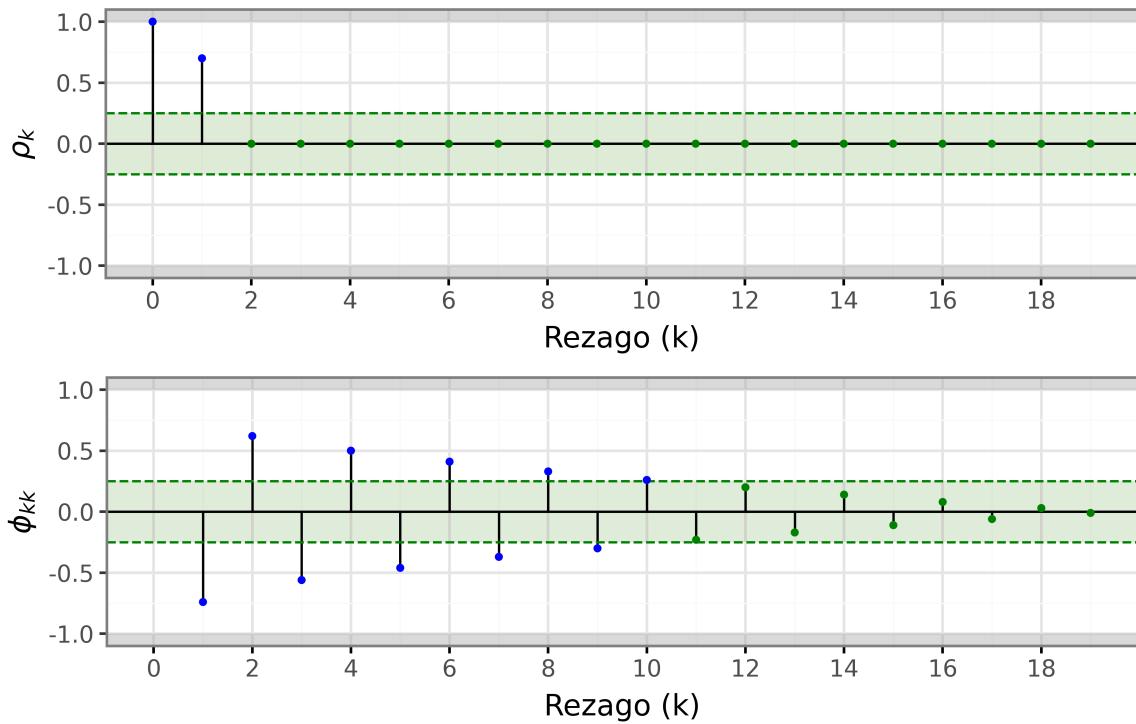


Figura 2: Ejemplo de las autocorrelaciones de un proceso $MA(1)$.

Un buen modelo $SARIMA$ debe cumplir las siguientes propiedades:

- Sus residuos se comportan como ruido blanco, es decir, están incorrelacionados y siguen una distribución normal, con media y variancia constantes.

- Es admisible, es decir, es invertible y estacionario.
- Es parsimonioso, en el sentido de que sus parámetros son significativos.
- Es estable en los parámetros, que se cumple cuando las correlaciones entre los parámetros no son altas.

3.3 Modelos de aprendizaje automático

El aprendizaje automático (*machine learning*) es una rama de la inteligencia artificial que permite a las computadoras aprender de los datos y realizar tareas de forma autónoma. Aunque los métodos presentados no fueron diseñados específicamente para datos temporales, han demostrado ser útiles en múltiples contextos mediante diversas pruebas empíricas.

Los métodos de *machine learning*, a diferencia de los modelos tradicionales, se enfocan principalmente en identificar los patrones que describen el comportamiento del proceso que sean relevantes para pronosticar la variable de interés, y no se componen de reglas ni supuestos que tengan que seguir. Para la identificación de patrones, estos modelos requieren la generación de características.

3.3.1 Introducción a árboles de decisión y ensamblado

Los árboles de decisión pueden ser explicados sencillamente como un conjunto extenso de estructuras condicionales *if-else*. El modelo pronosticará un cierto valor x si una cierta condición es verdadera, u otro valor y si es falsa. Es importante ver que no hay una tendencia lineal en este tipo de lógica, por lo que los árboles de decisión pueden ajustar tendencias no lineales. El resultado que se obtiene al aplicar esta técnica puede resumirse gráficamente como un camino que toma diferentes bifurcaciones (o un tronco con diferentes ramas), y de esta característica surge su nombre.

Un árbol puede tener distinta cantidad de divisiones en un mismo nivel, llamadas hojas, y profundidad, las cuales determinan en qué medida el modelo se ajusta a los datos con los que se entrena. Lógicamente árboles más profundos y con más hojas suelen generar sobreajuste, es decir, un modelo que se adapta demasiado a los datos de entrenamiento y generaliza mal.

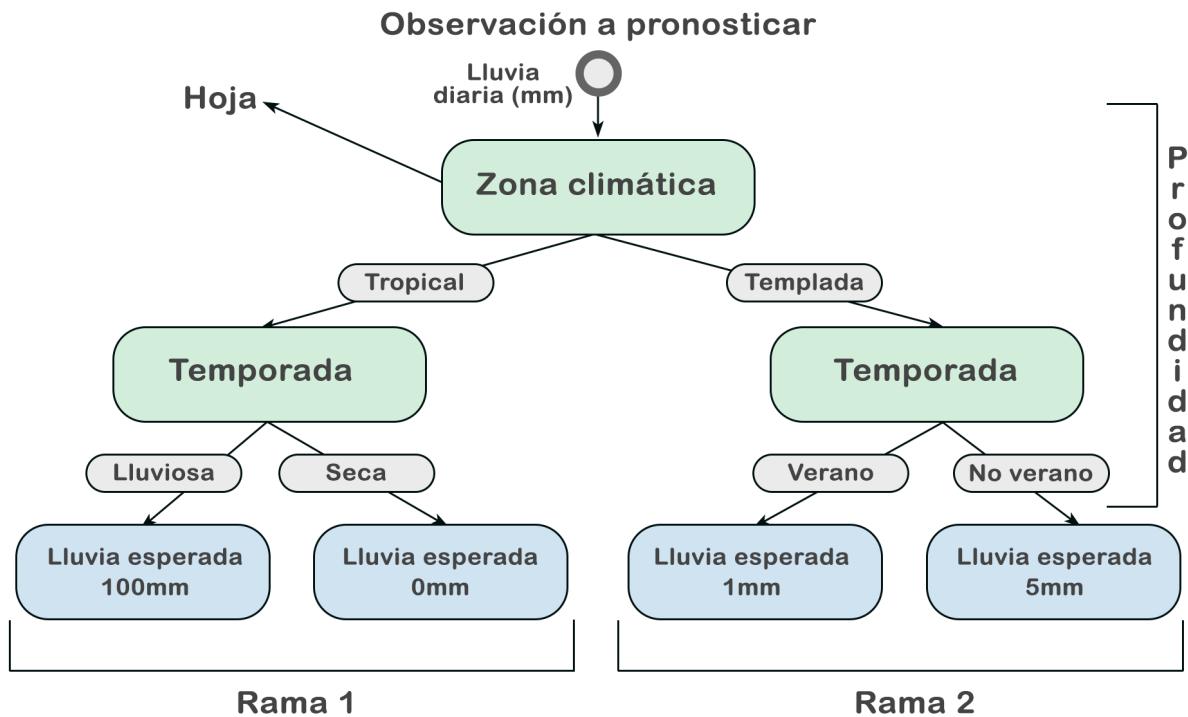


Figura 3: Ejemplo de árbol de decisión

Los métodos de ensamblaje combinan la predicción de varios estimadores base con el objetivo de mejorar la robustez de la predicción. Existen numerosos métodos de ensamblaje entre los cuales se encuentran los bosques de decisión y los árboles potenciados por gradiente (del inglés *Gradient-boosted trees*).

Boosting es un proceso iterativo, que consiste en la construcción de árboles de forma secuencial donde cada nuevo árbol busca predecir los residuos de los árboles anteriores. Es así entonces que el primer árbol buscará predecir los valores futuros de la serie, mientras que el segundo intentará predecir los valores reales menos los pronosticados por el primer árbol, el tercero tratará de inferir la diferencia entre los valores reales y el valor pronosticado del primer árbol menos los errores del segundo, y así sucesivamente. En cada iteración se pesan los puntos y se corrigen aquellos que tengan un mayor error por medio del descenso del gradiente.

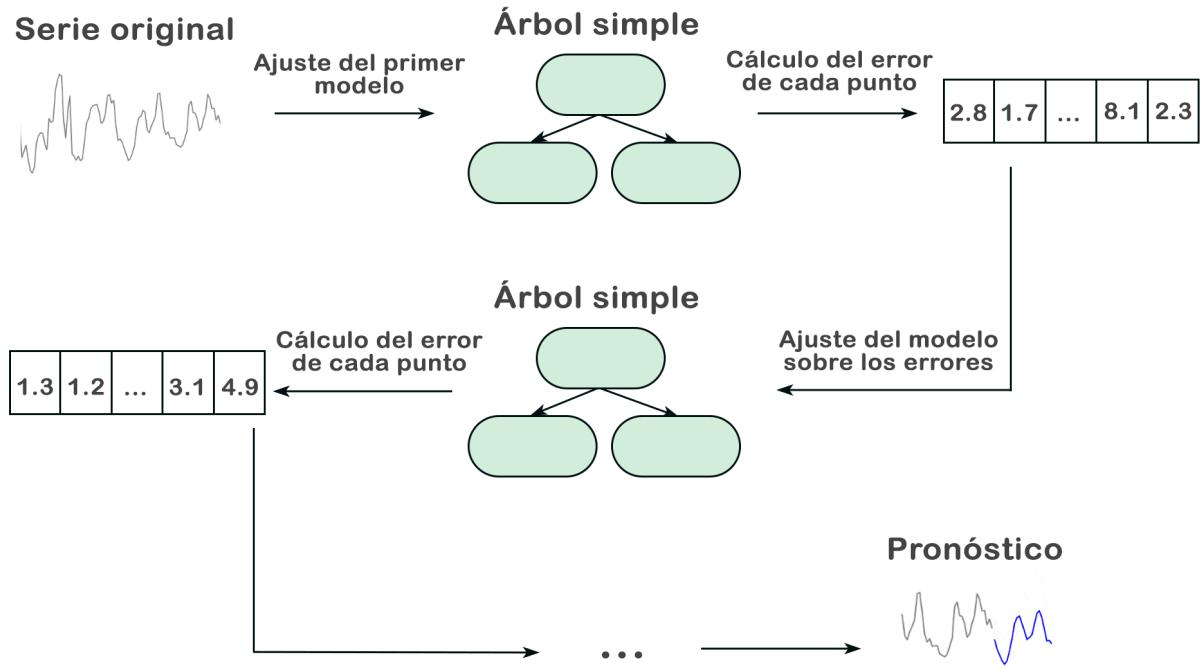


Figura 4: Proceso de ensamblado

La dirección de máximo crecimiento para una función está determinada por su gradiente, y del mismo modo, la dirección contraria a este gradiente es la dirección de máximo decrecimiento. De este modo, el descenso del gradiente busca encontrar los valores más bajos para una función de pérdida. El algoritmo de descenso de gradiente propone calcular el gradiente de la función de costo bajo el valor actual de parámetros, para luego modificarlo moviéndose en la dirección de mayor descenso. Este resultado se basa en derivadas, tasas de cambio instantáneo, por lo que conviene desplazarse una magnitud pequeña η , llamada “tasa de aprendizaje”. Es un algoritmo iterativo, en el que en cada paso la regla de actualización consiste en calcular la derivada de la función de costo respecto a cada parámetro y desplazarse una cierta magnitud η en la dirección contraria. Esto se repite un cierto número de veces hasta alcanzar la convergencia.

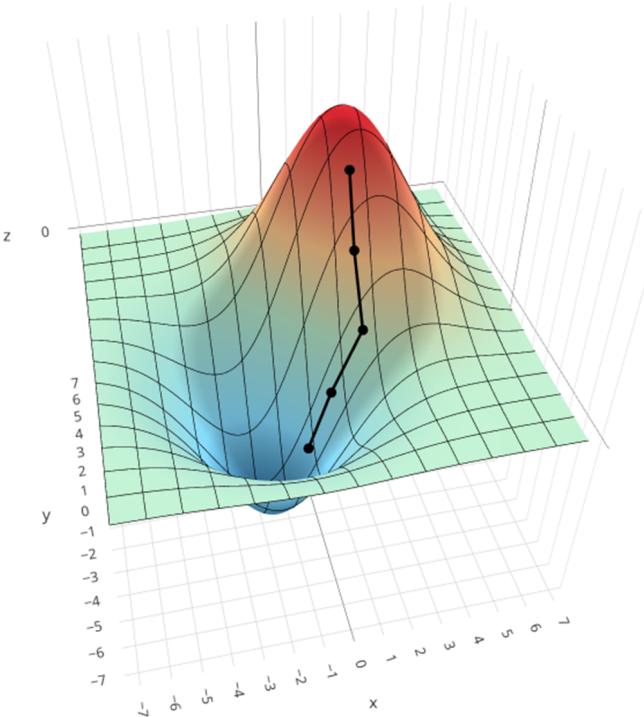


Figura 5: Ejemplo del descenso del gradiente en una función de pérdida

Sin embargo, los modelos no se construyen infinitamente, sino que se busca minimizar una función de pérdida que incluye una penalización por la complejidad del modelo, limitando así la cantidad de árboles que se producen. Existen múltiples métodos de ensamblaje (XGBoost, LightGBM, CatBoost, AdaBoost, entre otros) que se diferencian en la forma en la que se construyen los árboles. En esta tesina se usarán los algoritmos *eXtreme Gradient Boosting* (XGBoost) y *Light Gradient-Boosting Machine* (LightGBM).

3.3.2 Diferencias entre XGBoost y LightGBM

Las diferencias entre XGBoost y LightGBM radican en la forma en la que cada uno identifica las mejores divisiones dentro de los árboles y de que forma los hacen crecer.

Mientras que XGBoost usa un método en el que se construyen histogramas para cada una de las características generadas para elegir la mejor división por característica, LightGBM usa un método más eficiente llamado *Gradient-Based One-Side Sample* (GOSS). GOSS calcula los gradientes para cada punto y lo usa para filtrar afuera aquellos puntos que tengan un bajo gradiente, ya que esto significaría que estos están mejor pronosticados que el resto y no es necesario enfocarse tanto en ellos. Además, LightGBM utiliza un procedimiento que acelera el ajuste cuando se tienen muchas características correlacionadas de las cuales elegir.

A la hora de hacer crecer los árboles, XGBoost lo hace nivel a nivel, es decir que primero se crean todas las divisiones de un nivel, y luego se pasa al siguiente, priorizando que el árbol sea simétrico y tenga la misma profundidad en todas sus ramas. LightGBM, en cambio, se expande a partir de la hoja que más reduce el error, mejorando la precisión y eficiencia en series largas, pero arriesgándose a posibles sobreajustes si no se limita correctamente la profundidad de los árboles.

3.3.3 Intervalos de confianza en algoritmos de aprendizaje automático

Una de las principales ventajas de los modelos de aprendizaje automático respecto de los enfoques estadísticos tradicionales es que no requieren supuestos distribucionales estrictos sobre los errores. Sin embargo, esta flexibilidad también implica que, en general, no generan pronósticos probabilísticos de forma directa, lo que dificulta la construcción de intervalos de confianza.

Para subsanar esta limitación, se han desarrollado diversos métodos que permiten acompañar las predicciones puntuales con medidas de incertidumbre. A continuación, se describen dos de los enfoques más utilizados.

- Regresión cuantil con *boosting*: consiste en entrenar modelos para estimar directamente cuantiles de la distribución condicional de la variable objetivo, en lugar de su valor esperado. De este modo, pueden construirse intervalos de predicción utilizando, por ejemplo, los percentiles 5 y 95. Esta técnica está implementada en algunas variantes como LightGBM, pero no es directamente aplicable en XGBoost, lo que restringe su uso en ciertos entornos.
- Conformal prediction: se trata de una familia de métodos no paramétricos que permiten construir intervalos de predicción válidos bajo el supuesto de intercambiabilidad de las observaciones. Dado que este supuesto no se cumple en series temporales, donde existe dependencia temporal, se han desarrollado adaptaciones específicas para este tipo de datos.

Una de las más destacadas es el método *Ensemble Batch Prediction Intervals* (EnbPI), que permite aplicar conformal prediction en series temporales sin asumir independencia entre observaciones. Su procedimiento consiste en:

1. Seleccionar un modelo por ensamblado (como XGBoost o LightGBM).
2. Generar B muestras *bootstrap* por bloques, manteniendo así la estructura temporal de los datos.
3. Ajustar un modelo sobre cada una de las B muestras.
4. Para cada observación del conjunto de entrenamiento, calcular el residuo utilizando únicamente aquellos modelos que no la incluyeron.
5. Obtener las predicciones puntuales promediando los resultados de los B modelos.
6. Construir los intervalos de predicción sumando y restando los cuantiles empíricos de los residuos a las predicciones.

$$IC_{Z_{n+l};1-\alpha} = Z_n(l) \pm Q_{1-\alpha}(e) \quad (4)$$

Este método será el utilizado en esta tesina para estimar los intervalos de confianza en los algoritmos de aprendizaje automático y se aplica con la librería MAPIE.

3.4 Modelos de aprendizaje profundo

El *deep learning* (aprendizaje profundo) es una rama del *machine learning* que tiene como base un conjunto de algoritmos que intentan modelar niveles altos de abstracción en los datos usando múltiples capas de procesamiento, con complejas estructuras o compuestas de varias transformaciones no lineales.

Entre estos algoritmos se encuentran las redes neuronales, que imitan el funcionamiento del cerebro humano usando procesos que simulan la forma biológica en la que trabajan las neuronas para identificar fenómenos, evaluar opciones y llegar a conclusiones.

3.4.1 Introducción a redes neuronales

Una red neuronal está compuesta en grandes rasgos de 3 capas: entrada, oculta y salida. Dentro de cada capa se pueden encontrar neuronas y conexiones entre estas, donde cada neurona representa una variable y cada conexión un peso, y es por esto que a estas conexiones las llamaremos así en adelante. La suma de los pesos y las neuronas que no formen parte de la capa de entrada dan el total de parámetros que tiene que ajustar el modelo.

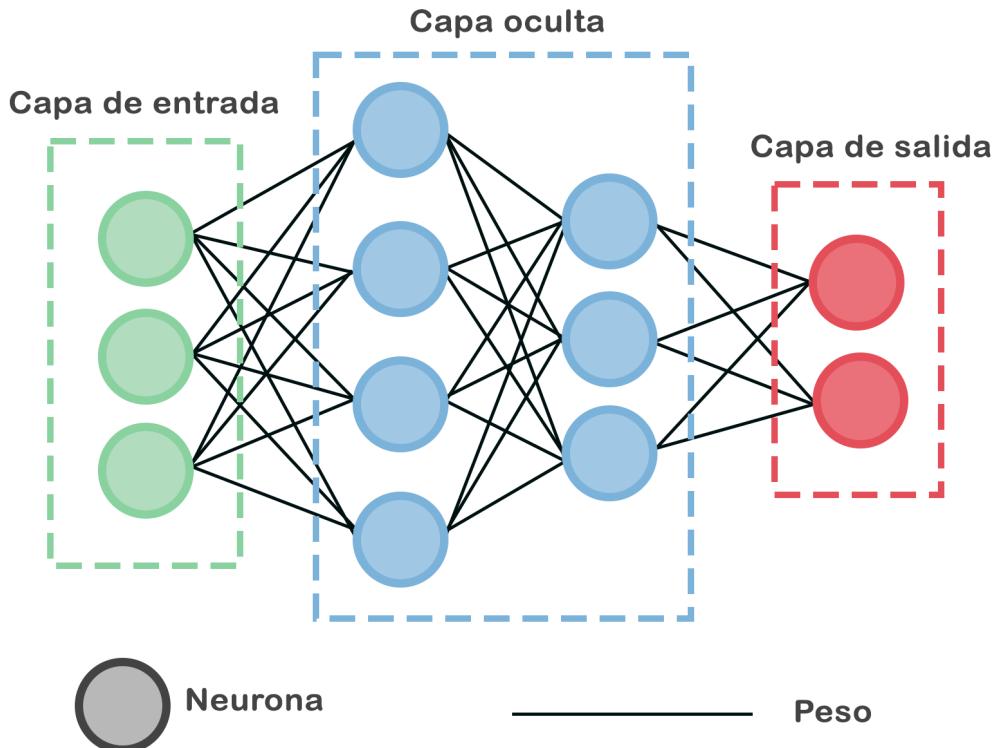


Figura 6: Red neuronal completamente conectada

En la capa de entrada se introducen las variables explicativas, y luego cada neurona fuera de esta capa es una función de las neuronas anteriores conectadas a la misma. Estas funciones son llamadas funciones de activación.

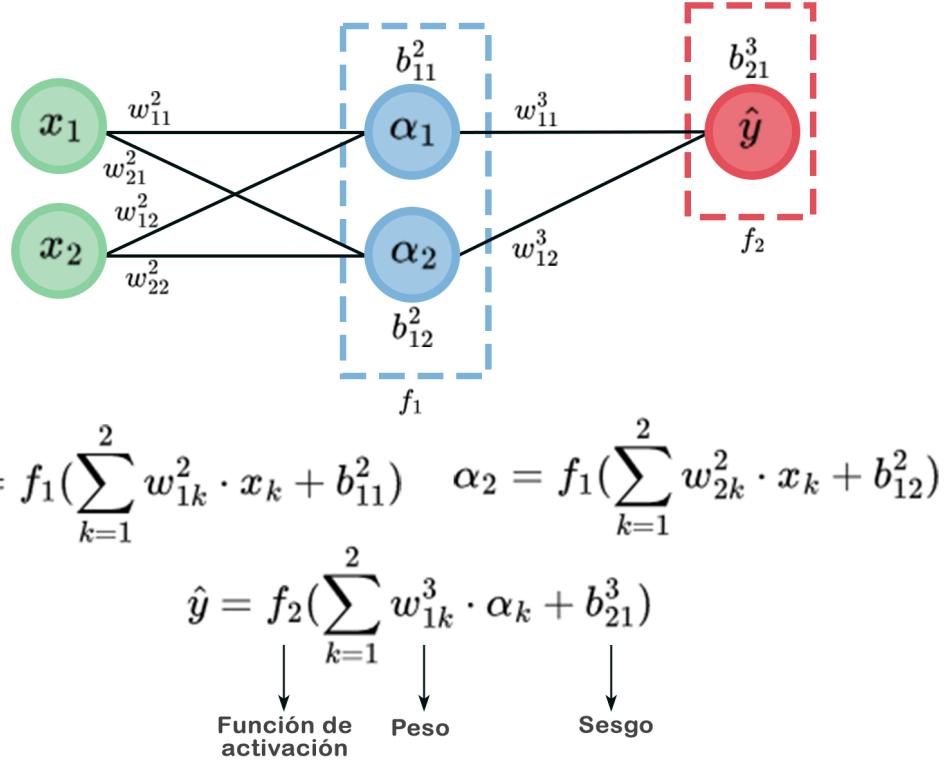


Figura 7: Red neuronal completamente conectada

Los parámetros se estiman buscando minimizar una función de costo, esto se logra con el descenso del gradiente y por medio de retropropagación. La retropropagación consiste en realizar una estimación inicial de la variable respuesta con los valores iniciales de la red neuronal, que pueden estar dados por ejemplo por una distribución normal, y de manera inversa a la dirección de la red neuronal calcular derivadas para encontrar la dirección de máximo decrecimiento de la función de costo para cada parámetro en la red neuronal.

Existen distintos tipos de redes neuronales según la forma en la que se conectan las neuronas. En esta tesina son de interés especialmente las *Convolutional Neural Networks* (CNN) y las *Recurrent Neural Networks* (RNN), en español, redes neuronales convolucionales y recurrentes respectivamente. Las primeras son útiles para el reconocimiento de patrones en los datos, mientras que las últimas son especialmente buenas en la predicción de datos secuenciales.

3.4.2 Long Short Term Memory (LSTM)

Lo que caracterizan a las redes neuronales recurrentes son los bucles de retroalimentación que se presentan en la figura 8. Mientras que cada neurona de entrada en una red neuronal completamente conectada es independiente, en las redes neuronales recurrentes se relacionan entre ellas y se retroalimentan.

Retroalimentación

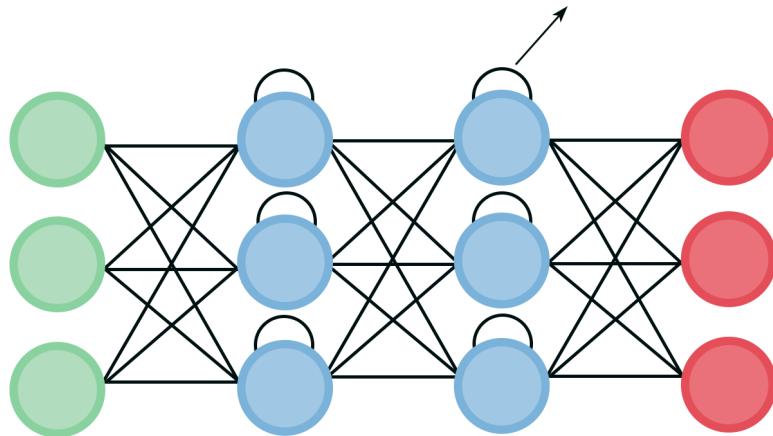


Figura 8: Ejemplo de RNN

Un problema frecuente en las RNN es su dificultad para capturar dependencias de largo plazo. Esto puede tener 2 causas, el desvanecimiento o la explosión del gradiente. El desvanecimiento del gradiente ocurre cuando, iteración tras iteración, el gradiente se aproxima a cero y se estabiliza, evitando que la red siga aprendiendo. Por el contrario, cuando el gradiente crece exponencialmente se habla de una explosión, esto lleva a inestabilidades en el aprendizaje, haciendo que las actualizaciones de los parámetros sean erráticas e impredecibles.

Las redes neuronales con memoria a corto y largo plazo (LSTM) son un tipo de RNN que solucionan este problema mediante un algoritmo logístico de 3 puertas, y usando una neurona que guarda la información histórica necesaria para la red.

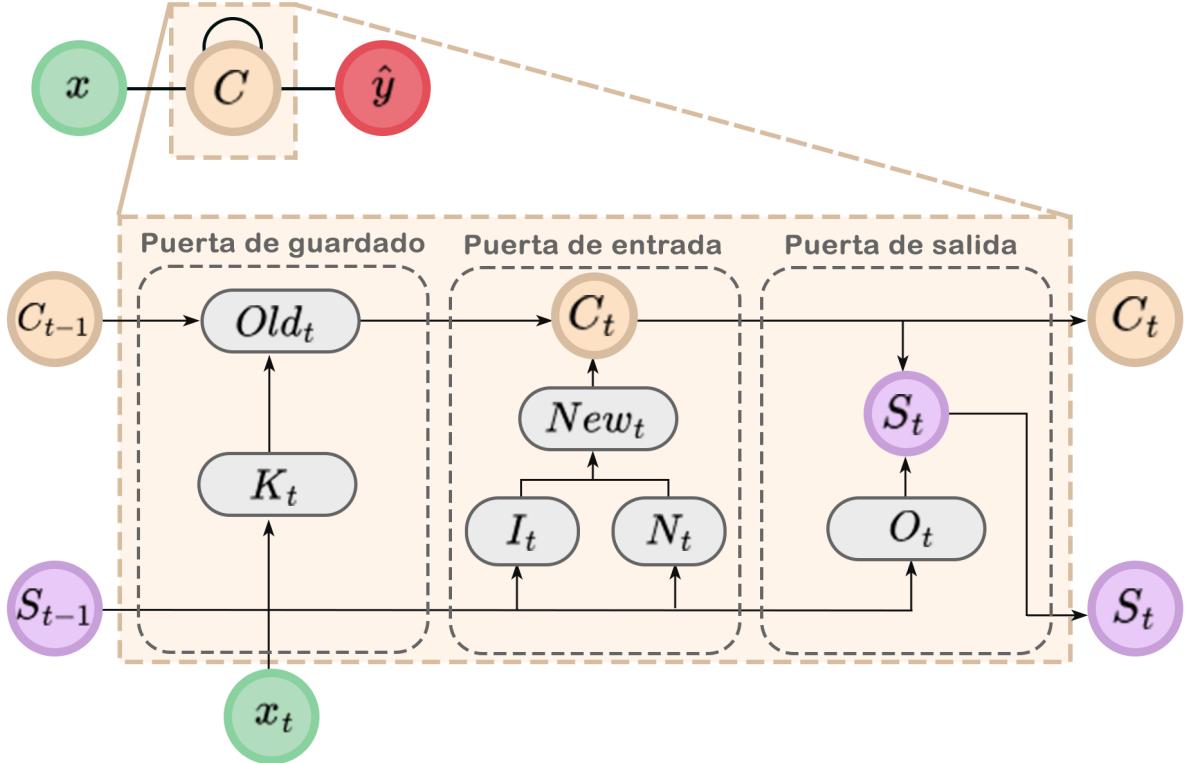


Figura 9: Estructura *Long Short Term Memory*

Puerta de guardado

La puerta de guardado se encarga de decidir si mantener o descartar la información actualmente guardada en la neurona de memoria de la red neuronal. Esta puerta recibe la entrada y el estado de la RNN, y las pasa como argumentos de una función sigmoide.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (5)$$

$$K_t = \sigma(W_k \times [S_{t-1}, x_t] + B_k) \quad (6)$$

Si K_t es igual a 1, significa que la información guardada debe ser mantenida perfectamente. Si K_t fuera igual a 0, la información guardada debe ser descartada completamente.

Sean S_{t-1} el estado actual de la RNN, x_t la entrada actual, y W_t y B_t los pesos y sesgos de la puerta de guardado respectivamente:

$$Old_t = K_t \times C_{t-1} \quad (7)$$

Donde C_{t-1} es la información guardada actualmente y Old_t lo que se mantendrá para la próxima iteración de la red.

Puerta de entrada

La puerta de entrada controla que información añadir a la neurona de memoria. Sean W_i y B_i los pesos y sesgos de la puerta de guardado:

$$I_t = \sigma(W_i \times [S_{t-1}, x_t] + B_i) \quad (8)$$

$$New_t = I_t \times N_t \quad (9)$$

Donde N_t es el nuevo valor propuesto por la red neuronal y New_t es la información que se va a agregar a la neurona de memoria. Luego, el nuevo valor de la neurona de memoria es:

$$C_t = Old_t + New_t$$

Puerta de salida

La puerta de salida se encarga de extraer la información más importante del estado actual de la neurona para usar como salida. Sean W_o y B_o los pesos y sesgos de la puerta de salida y $tanh(x)$ la función tangente:

$$O_t = \sigma(W_o \times [S_{t-1}, x_t] + B_o) \quad (10)$$

$$S_t = O_t \times \tanh(C_t) \quad (11)$$

Donde S_t es el nuevo estado de la red neuronal.

Las 3 puertas son lógicas para que sea sencillo aplicar la retropropagación. Este sistema de puertas evita los problemas de desvanecimiento y explosión del gradiente, y evita que se acumulen muchos estados por largos períodos de tiempo eligiendo que información es relevante guardar.

3.4.3 Modelos transformadores

Otro tipo de modelo de aprendizaje profundo son los *transformer models* (modelos transformadores), los cuales son significativamente más eficientes al entrenar y realizar inferencias que las RNNs; gracias al uso de mecanismos de atención, presentados en la publicación '*Attention is all you need*' de Google. La autoatención captura dependencias y relaciones en la secuencias de valores que se alimentan al modelo, logrando poner en contexto a cada observación.

Los modelos transformadores fueron creados originalmente con el propósito de generar texto. Sin embargo, tanto TimeGPT como Chronos explotan esta tecnología para el pronóstico de series de tiempo. Ambos modelos son preentrenados, lo cual significa que la optimización de parámetros y pesos fue realizada antes de usarse el modelo. Esto se logra entrenando y generalizando el modelo en un conjunto de datos extenso, por lo general de fuentes públicas. El preentrenamiento permite que el modelo adquiera conocimientos generales sobre la estructura y los patrones de los datos, los cuales luego pueden ser reutilizados en tareas concretas mediante técnicas como *fine-tuning* (ajuste fino). Los modelos preentrenados constituyen una gran innovación, lo que mejora la accesibilidad, precisión, eficiencia computacional y velocidad del pronóstico.

Dado que los modelos de lenguaje de texto utilizan diccionarios de *tokens*, que son segmentos de caracteres representados vectorialmente según ciertos parámetros, es necesario tokenizar los valores de la serie temporal. El diccionario de *tokens* con el que operan los modelos de lenguaje no es infinito, por lo tanto es necesario proyectar las observaciones a un set finito de *tokens*. Para cumplir esto, Chronos escala y discretiza las observaciones. TimeGTP por su parte

usa las mismas observaciones como *tokens*, esto dado que, si bien su arquitectura es la de un modelo transformador, esta no está basada en ningún modelo de lenguaje existente, y en cambio trabaja con un modelo especializado en series de tiempo entrenado para minimizar el error de pronóstico.

Para el escalado se aplica a las observaciones una transformación del tipo $f(x_i) = (x_i - m)/s$. Existen variadas técnicas de escalado eligiendo apropiadamente m y s , pero se opta por elegir $m = 0$ y $s = \frac{1}{C} \sum_{i=1}^C |x_i|$ debido a que preserva los valores iguales a cero, los cuales pueden ser importantes de destacar en numerosas aplicaciones.

Sin embargo estos valores siguen siendo números reales y no pueden ser procesados directamente por un modelo de lenguaje. Es por esto que se discretizan las observaciones. Se seleccionan B centros de intervalos en la recta real, c_1, c_2, \dots, c_B , y $B - 1$ extremos b_i que los separan, $c_i < b_i < c_{i+1}$ para $i \in \{1, \dots, B - 1\}$. Las funciones de discretización $q : \mathfrak{R} \rightarrow \{1, 2, \dots, B\}$, y de descuantificación $d : \{1, 2, \dots, B\} \rightarrow \mathfrak{R}$ se definen como:

$$q(x) = \begin{cases} 1 & \text{si } -\infty \leq x < b_1 \\ 2 & \text{si } b_1 \leq x < b_2 \\ \vdots & \\ B & \text{si } b_{B-1} \leq x < \infty \end{cases} \quad \text{y} \quad d(j) = c_j \quad (12)$$

Una vez se hayan transformado las observaciones para poder ser leídas por el modelo, el funcionamiento de un transformador es el siguiente:

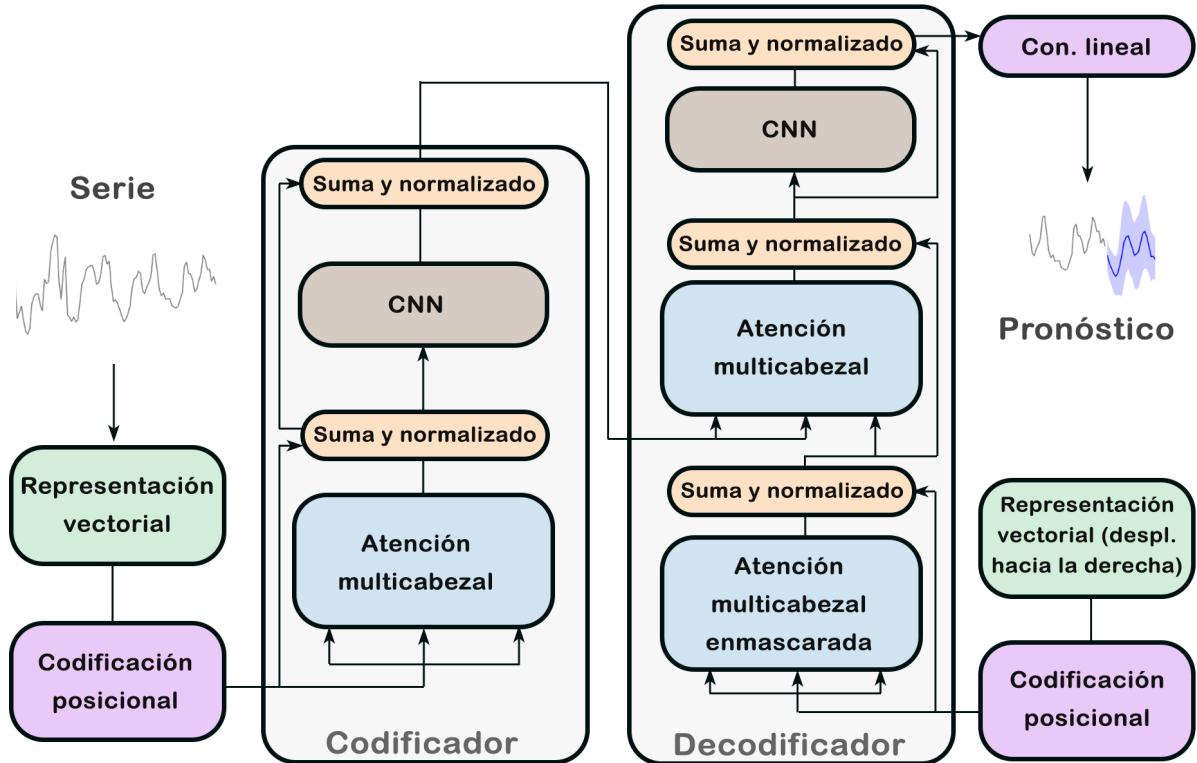


Figura 10: Diagrama de la estructura del modelo transformador de TimeGPT

Codificador

1. Representación vectorial: Cada *token* es transformado en un vector (vector de entrada) con muchas dimensiones (\vec{E}). Las dimensiones corresponden a diferentes características que el modelo definió en el preentrenado con una numerosa cantidad de parámetros.
2. Codificación posicional: Un set de valores adicionales o vectores son añadidos a los vectores de entrada antes de alimentarlos al modelo ($\vec{E} \leftarrow \vec{E} + \vec{P}$). Estas codificaciones posicionales tienen patrones específicos que agregan la información posicional del token.
3. Atención multi-cabezal (del inglés *Multi-Head Attention*): La autoatención opera en múltiples ‘cabezales de atención’ para capturar los diferentes tipos de relaciones entre tokens. Una cabeza de atención verifica el contexto en el que se presenta el token, y manipula los valores del vector que lo representa para añadir esta información contextual.

La verificación del contexto funciona gracias a una matriz denominada *Query* (W_Q) que examina ciertas características definidas con anterioridad en el preentrenado. El vector de entrada (\vec{E}) es multiplicado por esta matriz, resultando en un vector de consultas (\vec{Q}) para cada token. Una matriz de claves (W_K) que comprueba las relaciones con las características en la matriz de consultas, y tiene las mismas dimensiones que esta, es también multiplicada por \vec{E} generando así el vector de claves (\vec{K}). Luego, se forma una nueva matriz a partir de los productos cruzados entre los vectores \vec{K} y \vec{Q} de cada token, se divide por la raíz de la dimensión de los vectores¹ ($\sqrt{d_k}$) y se normaliza con softmax² por columna³ (\vec{S}), valores más altos indican que un token (de las columnas) está siendo influenciado por el comportamiento de otro token (de las filas).

	UN \downarrow \vec{E}_1 $\downarrow W_Q$ \vec{Q}_1	GRAN \downarrow \vec{E}_2 $\downarrow W_Q$ \vec{Q}_2	BLANCO \downarrow \vec{E}_3 $\downarrow W_Q$ \vec{Q}_3	AVIÓN \downarrow \vec{E}_4 $\downarrow W_Q$ \vec{Q}_4	EN \downarrow \vec{E}_5 $\downarrow W_Q$ \vec{Q}_5	EL \downarrow \vec{E}_6 $\downarrow W_Q$ \vec{Q}_6	AMPLIO \downarrow \vec{E}_7 $\downarrow W_Q$ \vec{Q}_7	CIELO \downarrow \vec{E}_8 $\downarrow W_Q$ \vec{Q}_8
UN $\rightarrow \vec{E}_1 \xrightarrow{W_K} \vec{K}_1$	-4.9	-23.2	-12.0	-5.4	-84.9	-48.7	-13.1	-52.9
GRAN $\rightarrow \vec{E}_2 \xrightarrow{W_K} \vec{K}_2$	-40.2	+4.7	-63.1	+82.3	-2.8	-40.6	-18.9	-20.9
BLANCO $\rightarrow \vec{E}_3 \xrightarrow{W_K} \vec{K}_3$	-13.9	-0.2	-12.3	+83.9	-85.1	-51.7	-23.6	+4.3
AVIÓN $\rightarrow \vec{E}_4 \xrightarrow{W_K} \vec{K}_4$	-91.6	-74.7	-16.5	-13.1	-62.7	-36.3	-48.9	-24.0
EN $\rightarrow \vec{E}_5 \xrightarrow{W_K} \vec{K}_5$	-17.3	-87.4	-21.0	-40.0	-18.4	-34.3	-72.8	-13.2
EL $\rightarrow \vec{E}_6 \xrightarrow{W_K} \vec{K}_6$	-97.9	-95.1	-73.4	-60.4	-84.0	-13.1	-34.3	-46.1
AMPLIO $\rightarrow \vec{E}_7 \xrightarrow{W_K} \vec{K}_7$	-54.1	-23.0	-84.7	-33.0	-67.7	-91.8	-10.6	+87.2
CIELO $\rightarrow \vec{E}_8 \xrightarrow{W_K} \vec{K}_8$	-21.9	-46.8	-67.0	-84.2	-75.5	-53.2	-84.6	-11.4

Figura 11: Verificación del contexto

¹Es útil para mantener estabilidad numérica, la cual describe cómo los errores en los datos de entrada se propagan a través del algoritmo. En un método estable, los errores debidos a las aproximaciones se atenúan a medida que la computación procede.

² $\text{softmax}(x) = \frac{e^{x_i/t}}{\sum_j e^{x_j/t}}$

³Aplicar softmax hace que cada columna se comporte como una distribución de probabilidad.

Ahora que se sabe que tokens son relevantes para otros tokens, es necesario saber como son afectados. Para esto existe otra matriz de valores (W_V) que es multiplicada por cada vector de entrada resultando en los vectores de valor (\vec{V}) que son multiplicados a cada columna. La suma por filas devuelven el vector ($\Delta \vec{E}$) que debe ser sumado al vector de entrada original de cada token.

	UN ↓ \vec{E}_1	GRAN ↓ \vec{E}_2	BLANCO ↓ \vec{E}_3	AVIÓN ↓ \vec{E}_4	EN ↓ \vec{E}_5	EL ↓ \vec{E}_6	AMPLIO ↓ \vec{E}_7	CIELO ↓ \vec{E}_8
UN $\rightarrow \vec{E}_1$	\vec{W}_V				0.00 \vec{V}_1			
GRAN $\rightarrow \vec{E}_2$		\vec{W}_V			0.17 \vec{V}_2			
BLANCO $\rightarrow \vec{E}_3$			\vec{W}_V		0.83 \vec{V}_3			
AVIÓN $\rightarrow \vec{E}_4$				\vec{W}_V	0.00 \vec{V}_4			
EN $\rightarrow \vec{E}_5$					0.00 \vec{V}_5			
EL $\rightarrow \vec{E}_6$					0.00 \vec{V}_6			
AMPLIO $\rightarrow \vec{E}_7$					0.00 \vec{V}_7			
CIELO $\rightarrow \vec{E}_8$					0.00 \vec{V}_8			

Figura 12: Influencia de tokens sobre otros luego de aplicar softmax

$$\Delta \vec{E}_j = \sum_i S_j \cdot \vec{V}_i \quad (13)$$

Con múltiples ‘cabezas’, cada una con sus propias matrices W_K , W_Q y W_V , se generan varios $\Delta \vec{E}$ que se suman y se añaden al vector de entrada original.

$$\vec{E} \leftarrow \vec{E} + \sum_h \Delta \vec{E}_h \quad (14)$$

Todo el mecanismo de atención se puede resumir con la siguiente función:

$$\text{Atencion}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (15)$$

Es importante notar que todas las matrices Q , K y V son preentrenadas.

4. Sumar y normalizar (*Add and norm*): En lugar de simplemente pasar los datos por las capas, las conexiones residuales se añaden sobre el vector de entrada en la salida de cada capa. Esto es lo que se hizo cuando se sumaron los cambios al vector de entrada, en lugar de directamente modificar el vector.

Dado que las redes neuronales profundas sufren de inestabilidad en los pesos al actualizarlos, una normalización al vector estabiliza el entrenamiento y mejora la convergencia.

5. Red neuronal convolucional (CNN): A diferencia de otros modelos transformadores tradicionales, TimeGPT incorpora *CNNs* para descubrir dependencias locales y patrones de corto plazo, tratando de minimizar una función de costo.

Decodificador

1. *Output embedding* (desplazado hacia la derecha): La entrada del decodificador son los tokens desplazados hacia la derecha.
2. Codificación posicional
3. *Masked multi-head attention* (Atención multicabezal enmascarada): Ahora que las predicciones deben hacerse únicamente con los valores previos a cada token, en el proceso de ‘atención’ y antes de aplicar la transformación softmax se debe reemplazar todos los valores debajo de la diagonal principal de la matriz QK por $-\infty$ para prevenir que los tokens sean influenciados por tokens anteriores.
4. *Multi-head attention*: En este caso, se usan las matrices de claves y valores que da como salida el codificador y la matriz de consultas es la salida de la capa de atención multicabezal enmascarada.
5. Conexión lineal: Es una capa completamente conectada que traduce las representaciones de atributos aprendidas en predicciones relevantes.

3.4.4 Diferencias entre TimeGPT y Chronos

TimeGPT es un modelo transformer preentrenado (de aquí las siglas GPT, *generative pre-trained transformer*) para el pronóstico de series de tiempo que puede producir predicciones en diversas áreas y aplicaciones con gran precisión y sin entrenamiento adicional. El mismo fue desarrollado por Nixtla y tuvo su primera beta privada en Agosto de 2023, volviéndose accesible a todo público desde el 18 de julio de 2024.

Chronos es una familia de modelos transformadores preentrenados para series de tiempo basados en arquitecturas de modelos de lenguaje, el cual fue desarrollado por Amazon y lanzado en marzo de 2024.

La primera diferencia entre ambos modelos es la accesibilidad al código fuente, mientras que Chronos es de código abierto, el modelo desarrollado por Nixtla no lo es.

Tal vez la diferencia más importante es como operan los modelos. Se mencionó anteriormente que TimeGPT utiliza la arquitectura transformer para diseñar un modelo que pueda trabajar directamente con series de tiempo. Chronos en cambio, aprovecha los modelos de lenguaje existentes para aplicarlos a datos temporales. Esto conlleva a otras diferencias clave, como que Chronos debe transformar los datos antes de poder procesarlos, y por trabajar con datos discretos, está entrenado para minimizar la entropía cruzada entre las distribuciones de las categorías reales contra las predichas.

La función de pérdida utilizada por Chronos está dada por:

$$\ell(\theta) = \sum_{l=1}^{h+1} \sum_{i=1}^{|\nu_{ts}|} 1_{z_{n+l+1}=i} \log p_\theta(z_{n+l+1} = i | z_{1:n+l}) \quad (16)$$

Donde $|\nu_{ts}|$ es el tamaño del diccionario de tokens, el cual depende del número de intervalos creados. z_{n+h+1} es la serie transformada en *tokens* de la cual las primeras n observaciones se usarán como entrenamiento para pronosticar las siguientes h , y se agrega al final un token EOS que se utiliza comúnmente en los modelos de lenguaje para denotar el final de la secuencia. p_θ es la probabilidad estimada por el modelo bajo la parametrización θ .

Es importante notar que no es una función que detecta distancias, por lo que se espera que el modelo asocie a los intervalos cercanos gracias a la información en el conjunto de entrenamiento. Es decir que Chronos aplica regresión por clasificación.

Otra diferencia entre TimeGPT y Chronos es el tipo de red neuronal que utilizan para detectar patrones en los datos, mientras que el primero hace uso de las CNN, el segundo aplica *Feed-Forward Networks*. Luego de la conexión lineal, Chronos necesita volver a aplicar softmax para obtener las probabilidades del pronóstico, procedimiento que no es necesario por parte de TimeGPT.

3.5 Métricas de evaluación

Para comparar el rendimiento de los modelos se utilizan métricas cuantitativas. Para los pronósticos puntuales se usará el porcentaje del error absoluto medio (MAPE), mientras que para los pronósticos probabilísticos se aplicará el *Interval Score*, propuesto por Gneiting y Raftery (2007), que penaliza tanto la amplitud de los intervalos como la falta de cobertura. Estas comparaciones permiten evaluar la precisión, la robustez y la eficiencia de cada enfoque.

Sea $e_l = Z_{n+l} - \hat{Z}_n(l)$ el error de la l -ésima predicción, donde $\hat{Z}_n(l)$ representa el pronóstico l pasos hacia adelante, algunas medidas del error para pronósticos h pasos hacia adelante son:

- Error Cuadrático Medio (*Mean Square Error, MSE*):

$$MSE = \frac{1}{h} \sum_{l=1}^h e_l^2 \quad (17)$$

- Error absoluto medio (*Mean Absolute Error, MAE*):

$$MAE = \frac{1}{h} \sum_{l=1}^h |e_l| \quad (18)$$

- Porcentaje del error absoluto medio (*Mean Absolute Percentage Error, MAPE*)

$$MAPE = \left(\frac{1}{h} \sum_{l=1}^h \left| \frac{e_l}{Z_{n+l}} \right| \right) \cdot 100\% \quad (19)$$

El problema de estos errores es que solo tienen en cuenta la estimación puntual, y por lo general es buena idea trabajar con pronósticos probabilísticos para cuantificar la incertidumbre de los valores futuros de la variable. Gneiting y Raftery (2007, JASA) propusieron en *Strictly Proper Scoring Rules, Prediction, and Estimation* una nueva medida del error que tiene en cuenta los intervalos probabilísticos de la estimación, llamándola *Interval Score*:

$$S = \frac{1}{h} \sum_{l=1}^h (W_l + O_l + U_l) \quad (20)$$

Donde:

$$W_l = IS_l - II_l \quad (21)$$

$$O_l = \begin{cases} \frac{2}{\alpha}(Z_n(l) - Z_{n+l}) & \text{si } Z_n(l) > Z_{n+l} \\ 0 & \text{en otro caso} \end{cases} \quad U_l = \begin{cases} \frac{2}{\alpha}(Z_{n+l} - Z_n(l)) & \text{si } Z_n(l) < Z_{n+l} \\ 0 & \text{en otro caso} \end{cases} \quad (22)$$

Siendo IS_l e II_l los extremos superior e inferior del intervalo del l -ésimo pronóstico respectivamente. Es fácil darse cuenta que W es una penalización por el ancho del intervalo, y que O y U son penalizaciones por sobre y subestimación respectivamente.

Se considera mejor al modelo que minimiza estas métricas.

3.6 Selección de parámetros y validación del modelo

La correcta elección de los parámetros del modelo constituye una de las tareas más importantes para lograr un buen ajuste de los datos. No resulta conveniente utilizar todos los datos disponibles para este fin, ya que esto puede conducir a un sobreajuste (*overfitting*). Se habla de sobreajuste cuando el modelo ase adapta excesivamente a los datos de entrenamiento, lo que perjudica su capacidad para generalizar sobre datos nuevos. Para evitar este problema se aplican técnicas específicas de validación que permiten seleccionar los parámetros sin comprometer la capacidad predictiva del modelo.

La validación *holdout* consiste en reservar una parte del conjunto de entrenamiento como validación, e ir probando las métricas de evaluación de las distintas configuraciones de parámetros del modelo, ajustado sobre el resto de los datos de entrenamiento, sobre este nuevo conjunto. Luego, para ajustar el modelo con la mejor combinación de parámetros, se utilizarán tanto los datos de entrenamiento como de validación. Por la ordinalidad de los datos, el conjunto de validación tendrá que ser más reciente que el conjunto de entrenamiento.

$$\text{Conjunto de entrenamiento total : } \left\{ \underbrace{z_1, \dots, z_c}_{\text{Entrenamiento}}, \underbrace{z_{c+1}, \dots, z_n}_{\text{Validación}} \right\} \quad (23)$$

Si alguna serie presentara estacionalidad, se priorizará que el conjunto de validación tenga el largo del ciclo, para poder evaluar el ajuste del modelo en todo el ciclo.

Para ARIMA se usará el método de Box-Jenkins. En este método se compararán aquellos modelos que cumplan con los supuestos, y se elegirá como mejor combinación de parámetros aquella que minimize el AIC, medida de ajuste que penaliza por la cantidad de parámetros.

4. Aplicación

La aplicación empírica de esta tesina tuvo como objetivo implementar, ajustar y comparar los modelos presentados en la sección 3, utilizando un conjunto de series temporales seleccionadas. Para ello, se empleó el lenguaje de programación Python, junto con diversas librerías de código abierto.

Se trabajó con series temporales reales, obtenidas de bases de datos públicas y confiables. Cada serie fue modelada desde tres enfoques metodológicos distintos: modelos estadísticos tradicionales, algoritmos de aprendizaje automático y modelos de aprendizaje profundo. Para cada uno de ellos, se exploraron distintas configuraciones de parámetros, explicando su significado y función en el ajuste.

- Los modelos estadísticos clásicos, como ARIMA y SARIMA, serán ajustados utilizando la librería `pmdarima`. La selección de parámetros se realizó manualmente y mediante procedimientos automáticos, tomando como criterio principal el Criterio de Información de Akaike (AIC).
- Para el enfoque de aprendizaje automático, se emplearon algoritmos de boosting, específicamente XGBoost y LightGBM, implementados con las librerías `xgboost` y `lightgbm` respectivamente.
- En el caso de los modelos de aprendizaje profundo, se entrenaron redes LSTM utilizando la librería `scalecast`.
- Finalmente, se exploraron dos modelos fundacionales preentrenados: TimeGPT, accedido a través de la API de Nixtla mediante la librería `nixtla`, y Chronos, una familia de modelos preentrenados desarrollada por *Amazon Web Services* cuya implementación se llevó a cabo con la librería `autogluon`.

Para cada modelo se obtuvieron tanto pronósticos puntuales como probabilísticos, con una confianza del 80%, y se evaluaron utilizando métricas como el error absoluto porcentual medio (MAPE) y el *Interval Score*. La mejor combinación de hiperparámetros para cada modelo se eligió en base al MAPE, con excepción de los modelos ARIMA. Además, se consideraron aspectos adicionales como el tiempo de cómputo, la facilidad de implementación y la interpretabilidad de los resultados. Los resultados fueron presentados en tablas comparativas y visualizaciones gráficas, acompañadas de un análisis crítico.

4.1 Series a utilizar

Con el objetivo de ver las capacidades de los modelos bajo distintas circunstancias, se decidió pronosticar sobre 3 series con distintas características. Estas son:

- Atenciones de guardia mensuales por patologías respiratorias en un hospital de Rosario.
- Personas registradas con empleo asalariado por mes en el sector educativo privado de Argentina.
- Temperatura horaria en la ciudad de Rosario.

La primera serie analizada corresponde al número mensual de atenciones de guardia por patologías respiratorias (Códigos CIE10: J09–J18, J21, J22 y J44) en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario. Esta información fue provista por la Dirección General de Estadística de la Municipalidad de Rosario. La serie mostró una marcada estacionalidad, con

picos en los meses de invierno, y una fuerte caída en 2020, atribuida a las medidas sanitarias adoptadas durante la pandemia de COVID-19. Esta estacionalidad es más discernible en el gráfico 30 del anexo.

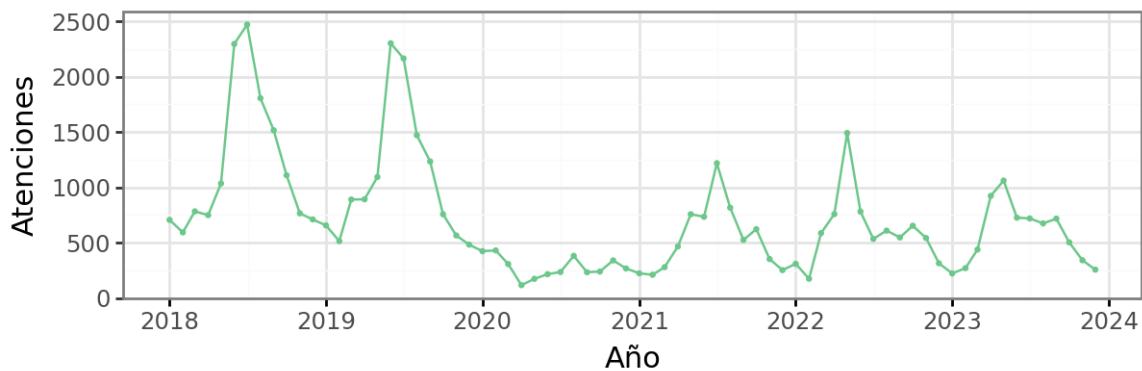


Figura 13: Atenciones en guardia por enfermedades respiratorias

La cantidad de personas en Argentina con empleo asalariado en el área de enseñanza y registradas en el sector privado aumenta casi constantemente año tras año. Presenta descensos drásticos en los meses de diciembre y enero. En esta serie también se evidencian las consecuencias de la pandemia. Los datos fueron relevados del informe de [Situación y evolución del Trabajo Registrado](#) elaborado por el Instituto Nacional de Estadísticas y Censos (INDEC).

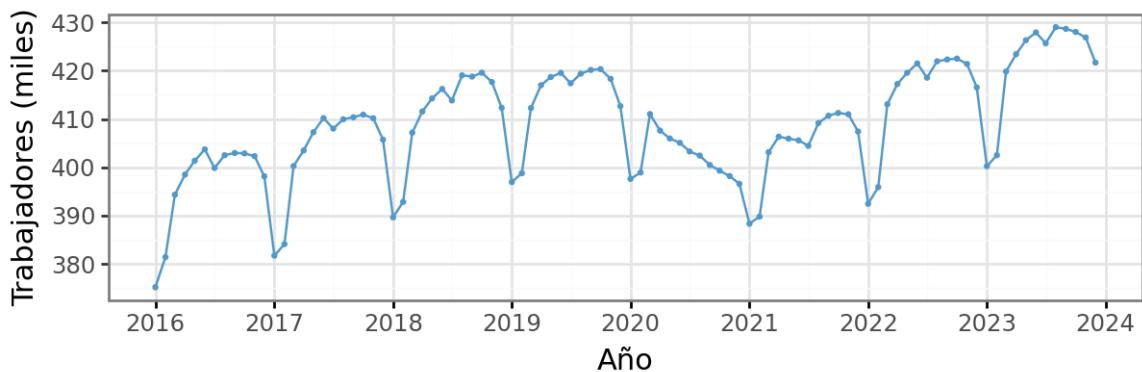


Figura 14: Personas con empleo asalariado en el área de enseñanza y registradas en el sector privado

Por último, se analizarán las temperaturas por hora a lo largo de los días del mes de marzo 2025, estudiando sus comportamientos en relación a la humedad relativa y a la presión atmosférica estándar. Gracias al gráfico 31 del anexo, se puede observar el patrón estacional diario que tiene la temperatura, la cual se mantiene constante entre la noche y la mañana, pero a la tarde sube pronunciadamente. La información necesaria fue extraída de la página del [Servicio Meteorológico Nacional](#).

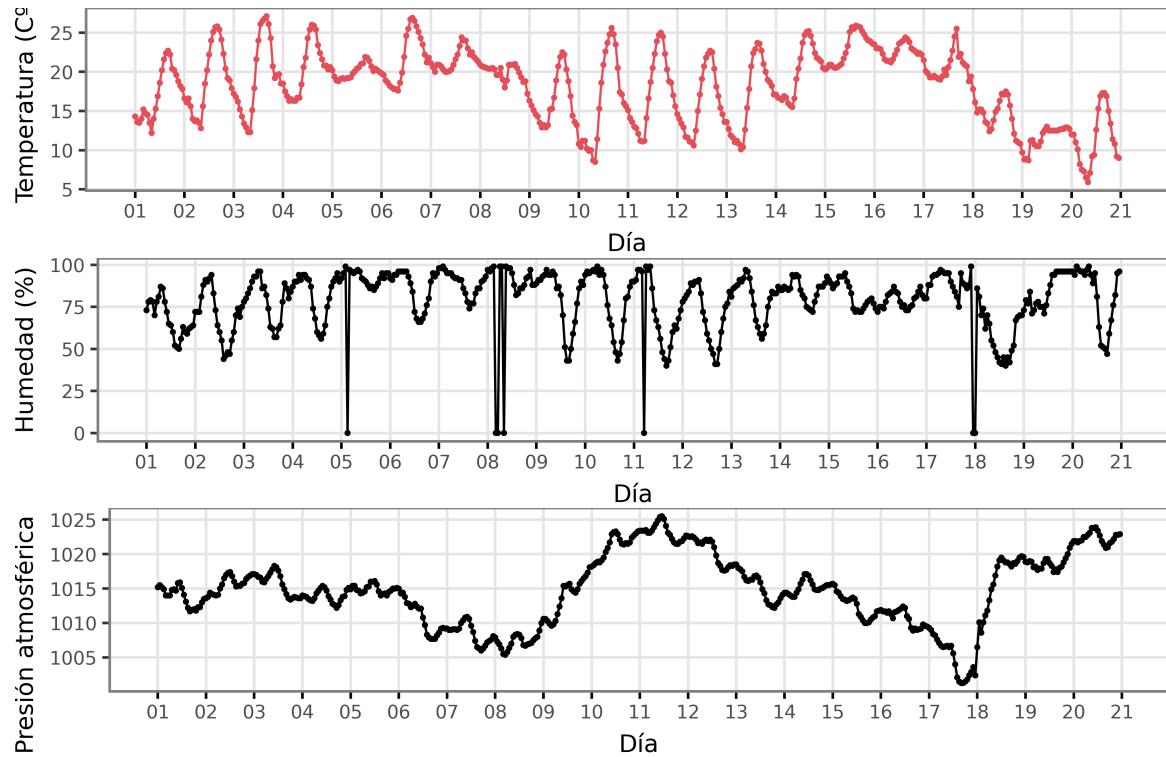


Figura 15: Temperatura (C°), humedad (%) y presión atmosférica (hPa) en Rosario por hora.

4.2 Ajuste y evaluación de modelos

4.2.1 ARIMA

Debido a la complejidad que implica modelar series de alta frecuencia con variables exógenas mediante ARIMA, la serie de temperatura se analizará únicamente a través del método automático de selección de modelos, descrito más adelante.

Antes de proponer modelos, se debe hacer un breve análisis exploratorio de las series. En primer lugar se debe observar que la variancia en cada período es la misma. De lo contrario se debería transformar la variable que se desea pronosticar.

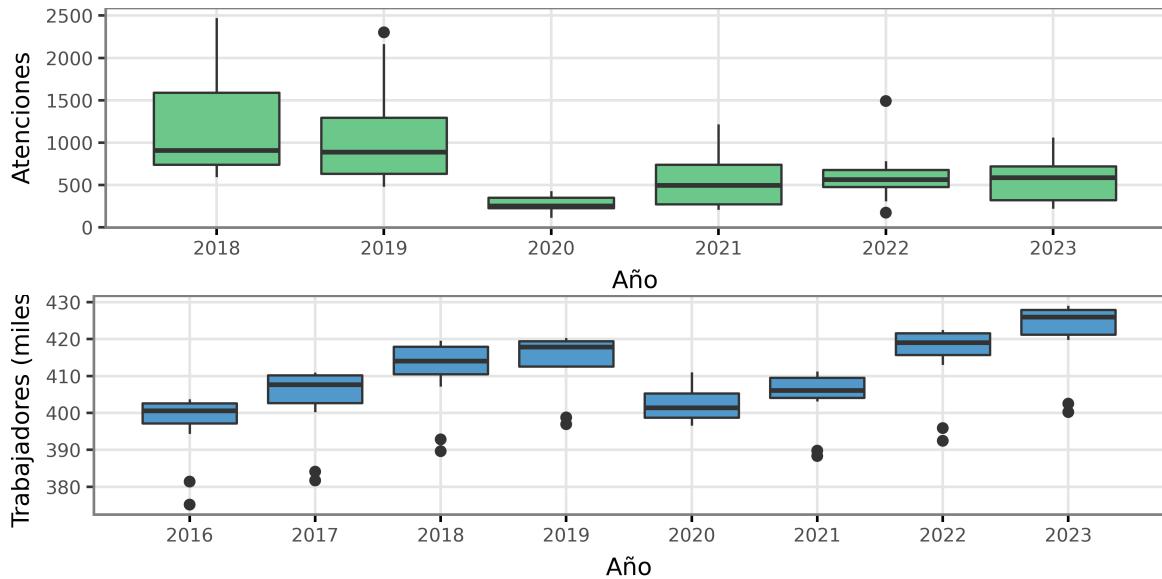


Figura 16: Variancias periódicas de las series.

Dadas las diferencias en la variabilidad anual de las atenciones por año, se calculó λ para la transformación de Box y Cox concluyendo que las atenciones en guardia por patologías respiratorias necesitan ser transformadas aplicando la función exponencial ($\lambda = -0.04$).

El primer paso de la modelización es estudiar las estacionalidades. Se mencionó anteriormente que las 2 series tienen estacionalidad, es por esto que se diferencian y se grafican, buscando que no sea visible ningún patrón estacional.

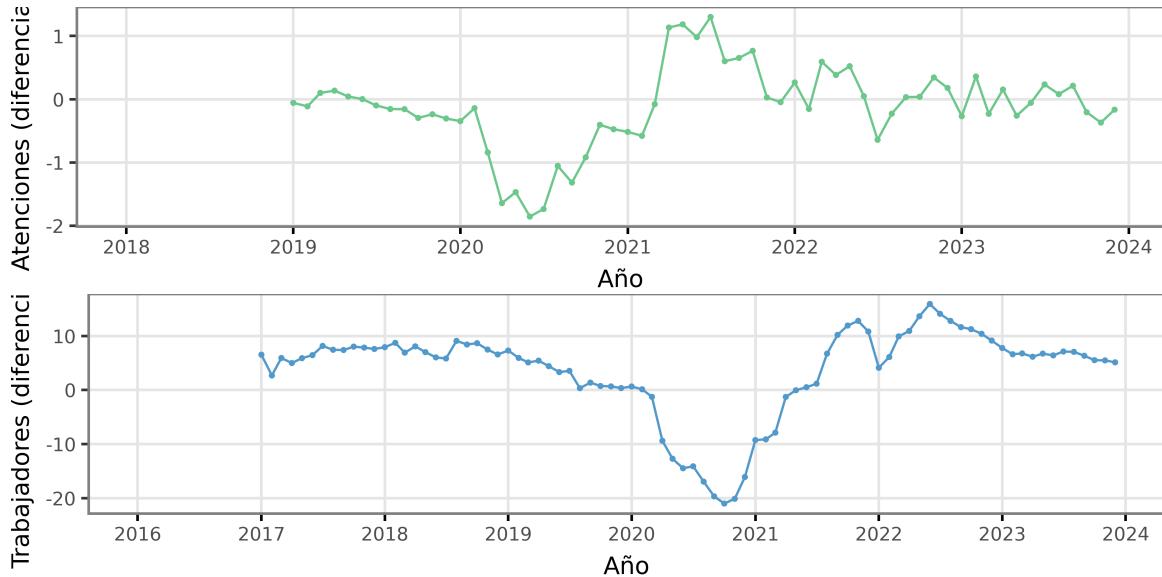


Figura 17: Series diferenciadas estacionalmente.

Una vez diferenciadas estacionalmente, es importante que no exista una tendencia clara, por lo que se estandarizan estacionariamente.

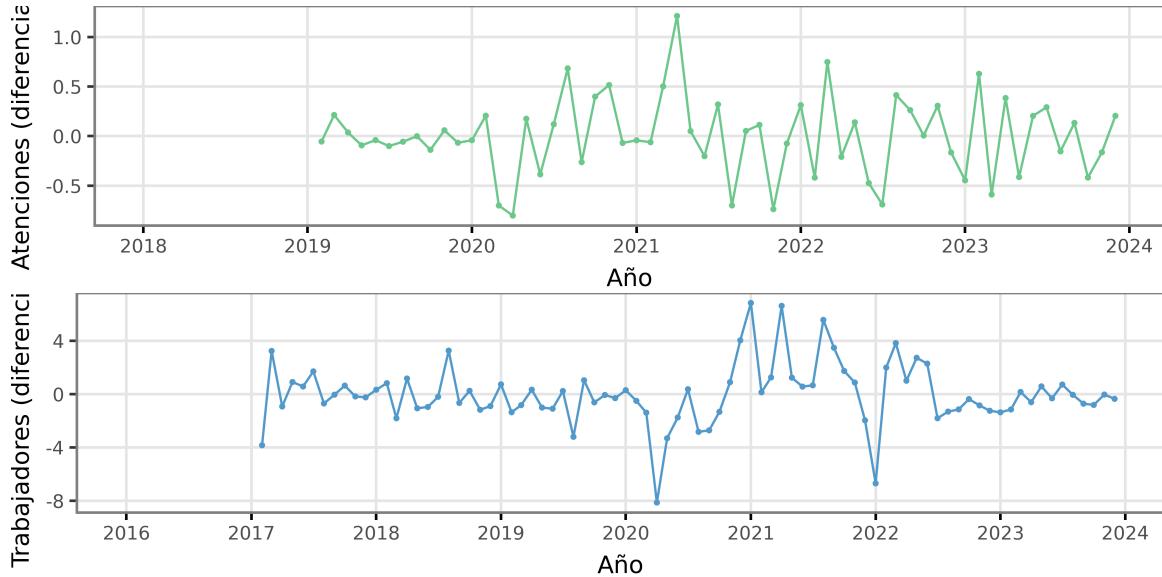


Figura 18: Series diferenciadas estacionariamente.

Para descubrir las componentes AR y MA que afectan a las series se grafican las autocorrelaciones y autocorrelaciones parciales de cada una.

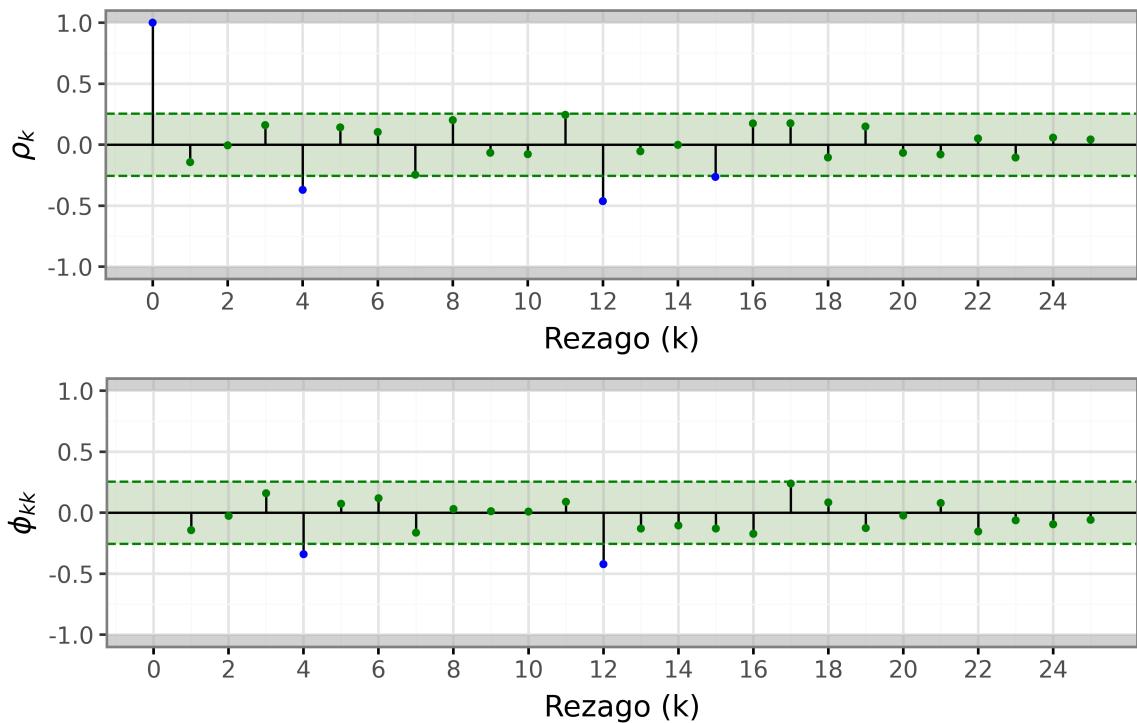


Figura 19: Autocorrelaciones de la serie de atenciones.

En la serie de atenciones se puede ver que tanto en las autocorrelaciones como en las autocorrelaciones parciales, hay valores significativos en el quinto y doceavo rezago, lo que podrían significar que existen componentes MA y AR en la parte estacional o estacionaria de la serie.

Se proponen entonces los modelos $SARIMA(0, 1, 1)(0, 1, 0)_{12}$ y $SARIMA(0, 1, 0)(0, 1, 1)_{12}$.

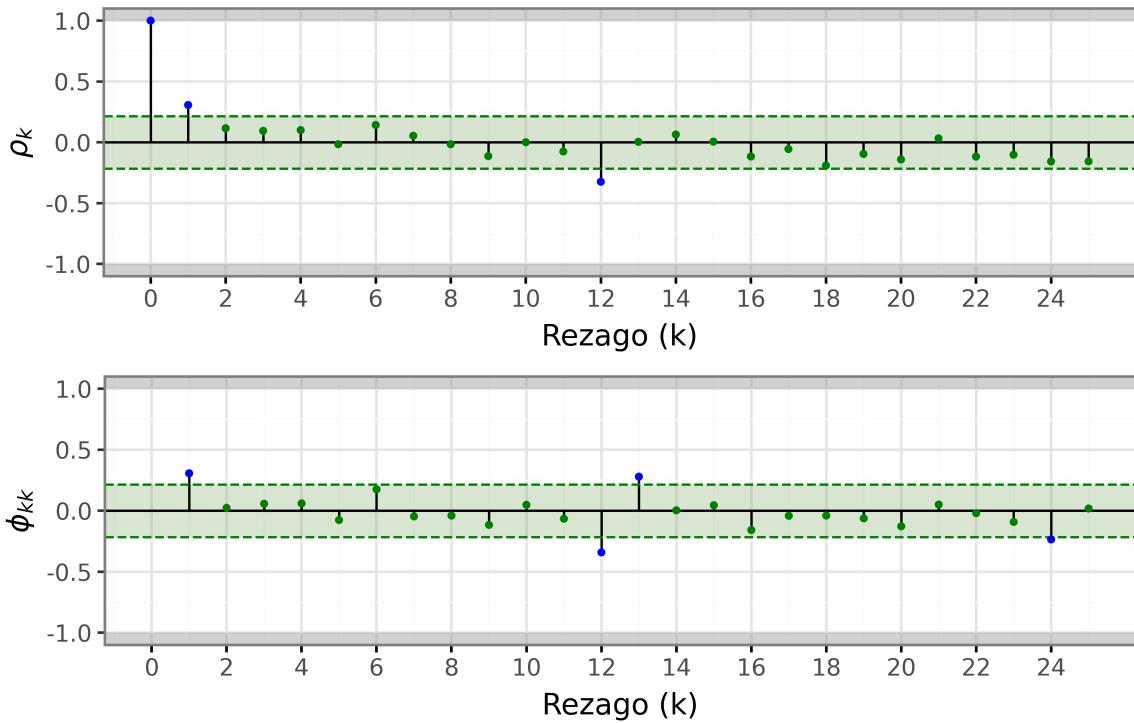


Figura 20: Autocorrelaciones de la serie de trabajadores.

En las autocorrelaciones de la serie de trabajadores resultan significativos el primer y doceavo rezago, mientras que en las autocorrelaciones parciales se destacan los rezagos 1, 12, 13 y 24. Se puede suponer que el rezago 13 no es en realidad significativo y que la serie presenta una componente MA en la parte estacionaria y una AR en la estacional, formando el modelo $SARIMA(0, 1, 1)(1, 1, 0)_{12}$.

Además, por medio de la función `auto_arima` de la librería `pmdarima` se propusieron modelos automáticos. Se elegirá como mejor modelo aquel que minimice el Criterio de Información de Akaike (AIC) y cumpla con las propiedades del modelo ARIMA.

Tabla 1: Cumplimiento de las condiciones de estacionariedad e invertibilidad de los modelos ajustados.

Modelo	AIC	Parte regular		Parte estacional	
		Est.	Inv.	Est.	Inv.
Atenciones en guardia					
$SARIMA(0, 1, 1)(0, 1, 0)_{12}$	857.6	Si	Si	Si	Si
$SARIMA(0, 1, 0)(0, 1, 1)_{12}$	852.1	Si	Si	Si	Si
$SARIMA(0, 1, 0)(1, 0, 0)_{12}$	1013.5	Si	Si	Si	Si
Trabajadores					
$SARIMA(0, 1, 1)(1, 1, 0)_{12}$	359.9	Si	Si	Si	Si
$SARIMA(2, 0, 0)(2, 1, 0)_{12}$	356.7	Si	Si	Si	Si
Temperatura					
$SARIMAX(1, 1, 1)(2, 0, 1)_{24}$	1067.1	Si	Si	No	Si

Para la serie de atenciones se seguirá trabajando con el segundo modelo propuesto y con el seleccionado de manera automática, ya que el parámetro MA del primer modelo propuesto no es significativo. Por otro lado, los dos modelos probados para la serie de

trabajadores seguirán siendo utilizados. El modelo automático para la serie de temperaturas no goza de estacionariedad, por lo que se proponen los modelos $SARIMAX(1, 1, 1)(1, 0, 1)_{24}$, $SARIMAX(1, 1, 0)(2, 0, 1)_{24}$ y $SARIMAX(1, 1, 0)(1, 1, 0)_{24}$, de los cuales únicamente el último cumple con todas las propiedades buscadas, con un AIC de 1119.3.

Las salidas de los modelos se pueden ver en el [anexo](#) para comprobar los resultados.

Lo siguiente es comprobar que los residuos estandarizados se comporten como ruido blanco.

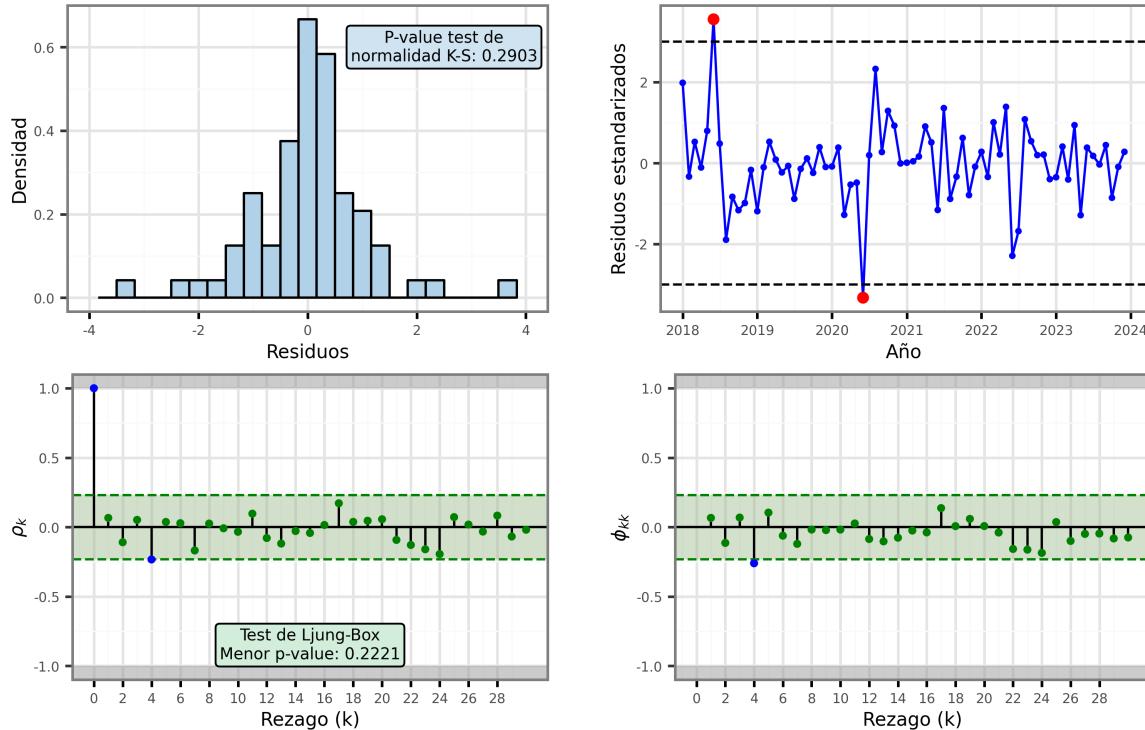


Figura 21: Comprobación de supuestos del segundo modelo arima manual para la serie de atenciones.

El histograma muestra la distribución de los residuos, pudiendo ver y comprobar con el test de Kolmogorov-Smirnov que es normal. La serie de los residuos muestra como estos no tienen un patrón particular ni variabilidad dependiente del tiempo, solo 2 *outlayers* fáciles de ignorar. Por último, los gráficos inferiores muestran las autocorrelaciones y autocorrelaciones parciales, esperando que no haya ninguna significativa. Por medio del test de Ljung-Box se comprueba que ninguna lo es y se concluye que los residuos no están correlacionados. Por lo tanto todos los supuestos se cumplen para este modelo.

Con el propósito de no poblar de gráficos el documento, el resto de los gráficos para la comprobación de supuestos de los demás modelos se encuentran en el [anexo](#). En estos se destaca que todos los modelos, con excepción del seleccionado automáticamente para los trabajadores, superaron la comprobación de supuestos. El problema de este modelo es la existencia de correlación en los residuos, supuesto que es sumamente importante al ajustar modelos ARIMA. Si bien los residuos del modelo seleccionado manualmente para los trabajadores no cumplen con normalidad, es sencillo ver que esto se debe a unos pocos valores extremos, y que sin estos es muy probable que sí se distribuyan normalmente.

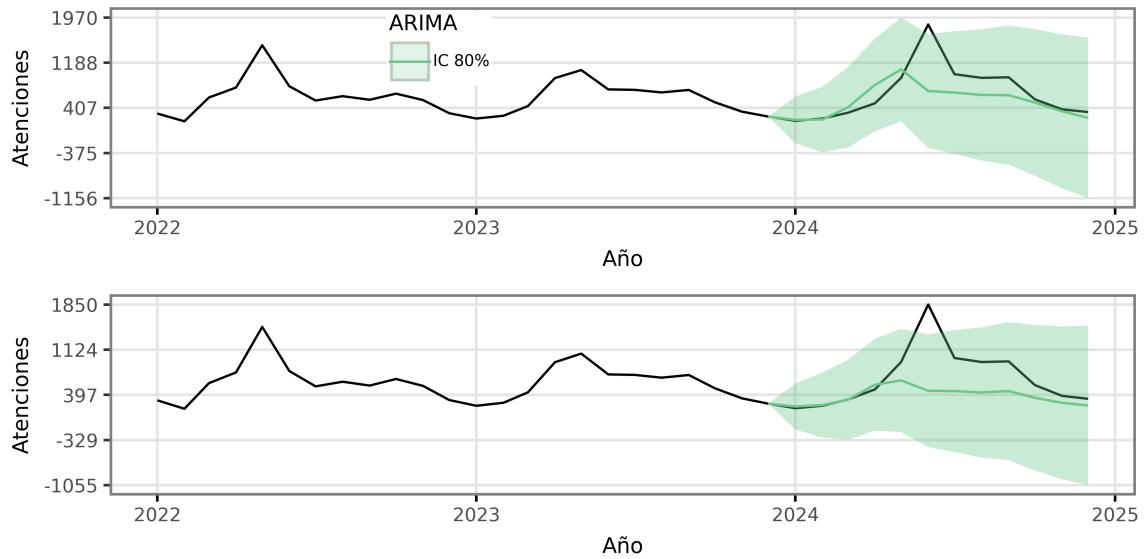


Figura 22: Pronóstico de atenciones en guardia con ARIMA.

Tabla 2: Métricas de evaluación para la serie de atenciones por guardia con modelos ARIMA.

Modelo	Horizonte	MAPE	<i>Interval Score</i>
<i>SARIMA</i> (0, 1, 0)(0, 1, 1) ₁₂	3	0.1529	1108.9433
	6	0.3157	1730.9364
	12	0.2794	2096.9139
<i>SARIMA</i> (0, 1, 0)(1, 0, 0) ₁₂	3	0.0694	1023.0534
	6	0.2391	2138.0755
	12	0.3328	2205.1062

Se puede observar en la tabla 2 que para la serie de atenciones en guardia por patologías respiratorias el modelo automático parece pronosticar mejor en los primeros 6 meses. Sin embargo, este no capta el aumento de casos en invierno, y es por esto que en el pronóstico a 12 meses el segundo modelo manual es mejor.

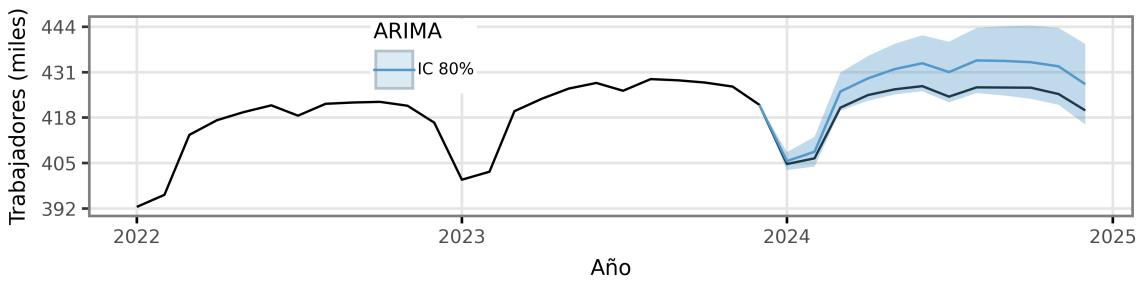


Figura 23: Pronóstico para la serie de trabajadores.

Tabla 3: Métricas de evaluación para la serie de trabajadores registrados con modelos ARIMA

Modelo	Horizonte	MAPE	<i>Interval Score</i>
<i>SARIMA</i> (0, 1, 1)(1, 1, 0) ₁₂	3	0.0059	24.4803
	6	0.0096	11.2827
	12	0.0137	15.7902

El modelo ARIMA elegido para la serie de trabajadores registrados en el sector privado de enseñanza logra captar muy bien el patrón de datos y el pronóstico es muy cercano a los valores reales.

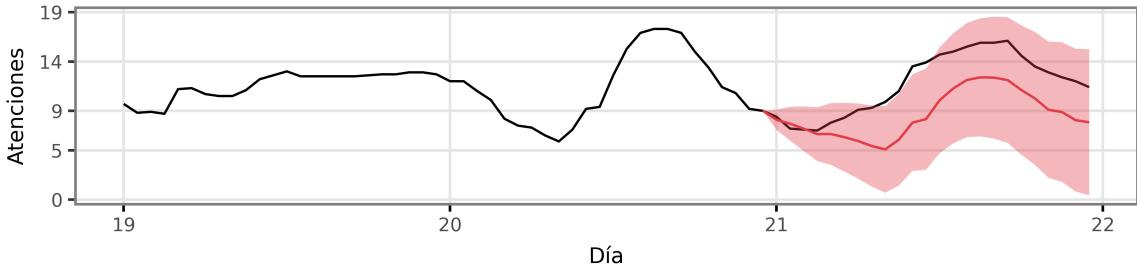


Figura 24: Pronóstico para la serie de temperaturas

Tabla 4: Métricas de evaluación para la serie de temperaturas con modelos ARIMA.

Modelo	Horizonte	MAPE	<i>Interval Score</i>
<i>SARIMA</i> (1, 1, 0)(1, 1, 0) ₂₄	6	0.0926	4.7732
	12	0.2572	8.6444
	24	0.2606	10.7507

El pronóstico de la temperatura es muy bueno en las primeras 6 horas, pero empeora ligeramente en horizontes más largos, sin embargo, los intervalos de confianza incluyen casi constantemente a los valores reales y se adapta bien a los cambios estacionales.

Un problema que se encuentra es que el intervalo de confianza para las atenciones por guardia toma valores negativos, algo claramente ilógico. Esto se puede resolver ajustando modelos sobre el logaritmo de la variable, y luego transformando los valores pronosticados a la escala original. Esta solución queda propuesta para un futuro trabajo.

4.2.2 Modelos de aprendizaje automático

Los modelos de pronóstico basados en el aprendizaje automático no tienen supuestos que cumplir, por lo que el modelaje se vuelve mucho más sencillo y automatizable. Los únicos aspectos en los que se debe tener especial cuidado y atención es en la selección de características y parámetros del modelo.

Tanto para XGBoost como para LightGBM las características elegidas para todas las series fueron las siguientes:

- Identificación temporal

- El promedio de las 3 observaciones anteriores
- El desvío estándar de las 3 observaciones anteriores
- El valor del primer rezago
- El valor del segundo rezago
- El valor del rezago estacional

Con el objetivo de seleccionar los mejores hiperparámetros, se elaboró una grilla con diferentes combinaciones de valores para cada parámetro. Se ajustaron modelos con cada combinación en la grilla y se evaluaron sobre un conjunto de validación.

XGBoost y Lightgbm permiten parametrizar los modelos de varias formas, los parámetros que se decidieron probar en este trabajo son los siguientes:

- Número de árboles que se construyen paralelamente en cada iteración (A). Opciones: 20, 50, 100, 150.
- Profundidad máxima del árbol (P). Opciones: 2, 3, 4, 5.
- Número máximo de hojas del árbol (H). Opciones: 2, 4, 8, 16.
- Tasa de aprendizaje en el método del gradiente (η). Opciones: 0.001, 0.1, 0.2.
- Proporción de características que se usa en cada árbol (C). Opciones: 0.7, 1.

En las tablas 5 y 6 se muestran para cada serie que combinación de parámetros, de las 384 posibles, fue la que menor MAPE produjo sobre el conjunto de validación aplicando XGBoost o LightGBM respectivamente. Además, se presenta el MAPE e *Interval Score* que devuelve el modelo entrenado con todos los datos de entrenamiento y evaluado sobre el conjunto de prueba.

Tabla 5: Modelos XGBoost seleccionados y métricas de evaluación.

Serie	Hor.	A	P	H	η	C	MAPE	Interval Score
Atenciones	3	50	2	2	0.2	0.7	0.8615	1454.4965
	6	150	2	2	0.001	0.7	1.2669	1849.6949
	12	100	2	2	0.2	1.0	0.3684	1515.9289
Trabajadores	3	150	2	2	0.2	1.0	0.0083	20.514
	6	150	2	2	0.2	1.0	0.0091	20.514
	12	150	5	2	0.2	1.0	0.0069	21.9308
Temperatura	6	50	3	2	0.1	1.0	0.2775	35.6635
	12	50	2	2	0.2	1.0	0.2995	36.3026
	24	50	2	2	0.2	1.0	0.0583	2.7636

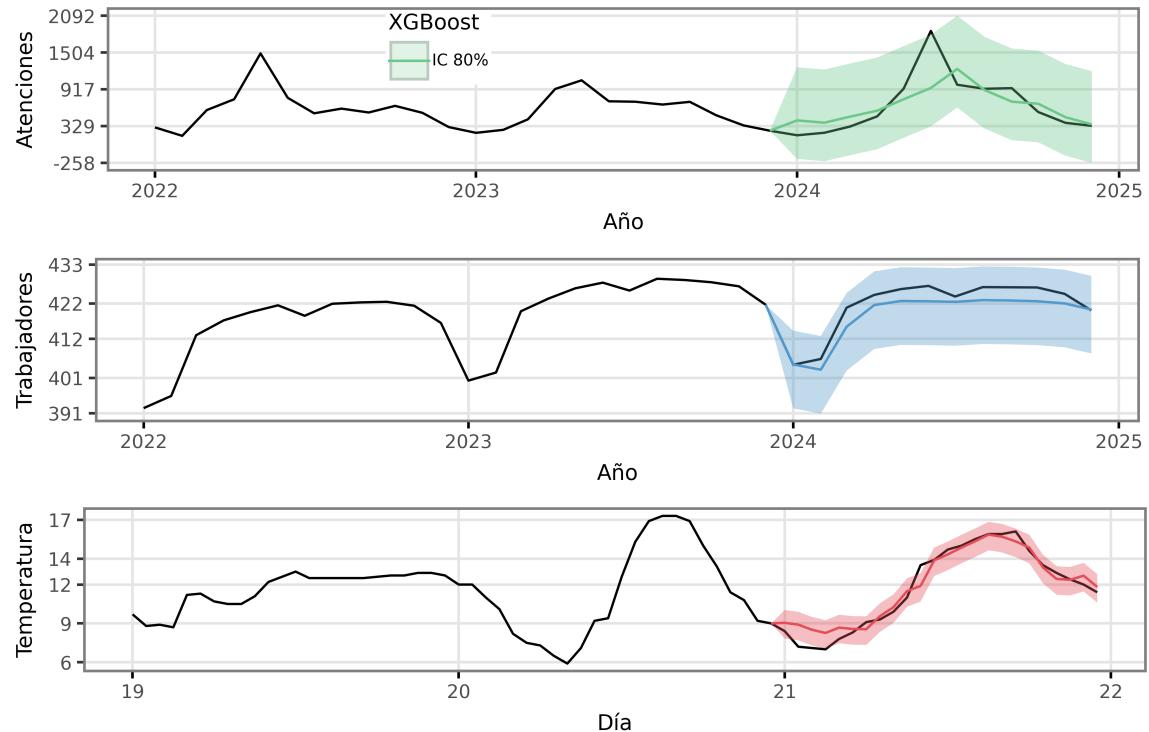


Figura 25: Pronósticos con XGBoost.

Los pronósticos con XGBoost fueron muy buenos sobre las 3 series. Si algo se puede remarcar es que en la serie de atenciones el modelo no capta del todo el pico de atenciones en invierno.

Tabla 6: Modelos LightGBM seleccionados y métricas de evaluación.

Serie	Hor.	<i>A</i>	<i>P</i>	<i>H</i>	η	<i>C</i>	MAPE	Interval Score
Atenciones	3	150.0	2.0	4.0	0.2	0.7	0.8639	1559.4147
	6	150.0	2.0	4.0	0.001	0.7	1.2655	1846.2058
	12	100.0	2.0	4.0	0.2	1.0	0.3781	1537.1091
Trabajadores	3	20.0	2.0	2.0	0.001	1.0	0.0144	32.417
	6	150.0	2.0	4.0	0.2	1.0	0.0092	23.8802
	12	150.0	3.0	4.0	0.2	0.7	0.0089	25.2876
Temperatura	6	150.0	5.0	16.0	0.1	0.7	0.265	33.1287
	12	100.0	5.0	16.0	0.1	1.0	0.2507	29.8732
	24	100.0	2.0	2.0	0.2	1.0	0.0931	4.6673

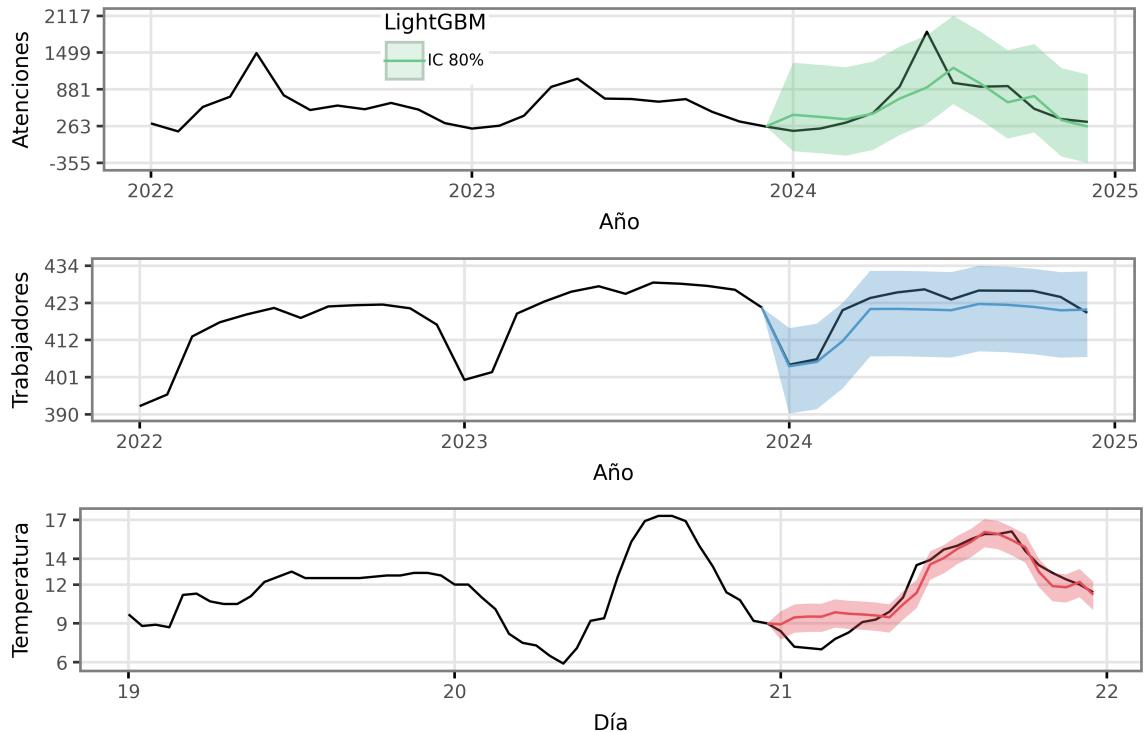


Figura 26: Pronósticos con LightGBM.

Los pronósticos con LightGBM son ligeramente peores a los conseguidos con XGBoost en todos los casos. Esta diferencia es especialmente importante en la serie de temperaturas, ya que en el pronóstico a 24 horas los intervalos de confianza en las primeras 6 horas no suelen cubrir a las observaciones. Este modelo tampoco pudo identificar el pico de atenciones en guardia en los meses invernales.

Algo interesante de notar sobre los modelos a partir de las tablas 5 y 6 es que XGBoost suele elegir bosques más sencillos, con menos árboles, que a su vez son menos profundos y tienen menos hojas. También se evidenció la tendencia de LightGBM a expandir los árboles por las hojas. Otro aspecto importante de mencionar es que los modelos optaron en general por elegir una tasa de aprendizaje y una proporción de características utilizadas altas, elecciones que podrían llevar a un sobreajuste. Aún así, esto no se vio evidenciado en los resultados.

4.2.3 Redes neuronales (LSTM)

Los modelos basados en aprendizaje profundo se encargan de definir las características más relevantes y los parámetros más adecuados de forma automática. En las redes neuronales, como es el caso de LSTM, solo se tendrá que elegir la forma del modelo.

Para evitar el sobreajuste en redes neuronales existen numerosas alternativas. Entre estas se puede optar por frenar el entrenamiento antes de completar todas las iteraciones si es que no se ve mejora en la función de pérdida, esto se conoce como *early stopping*. Otra opción es “ignorar” una cierta proporción de neuronas en cada iteración, que es equivalente a entrenar distintas redes neuronales. Esta última técnica es denominada *dropout*.

Para cada serie se entrenaron modelos LSTM con 300 iteraciones y una tasa de aprendizaje de 0.001. Se adoptó una paciencia para el *early stopping* de 10, esto quiere decir que si la función de

pérdida no muestra mejoras en 10 iteraciones seguidas se detiene el entrenamiento. Además, se probaron las siguientes características para el modelo:

- Capas y neuronas en el modelo (N). Opciones: [32], [12,24], [24,42].
- Rezagos con los que se entrena el modelo (R). Opciones: 1, 12, 24.
- *Dropout* en cada capa (D). Opciones: 0.1, 0.3.
- Función de activación (A). Opciones: ReLu, Tangente Hiperbólica (tanh).

Tabla 7: Modelos LSTM seleccionados y métricas de evaluación.

Serie	Hor.	N	R	D	A	MAPE	Interval Score
Atenciones	3	[12, 24]	24	0.3	relu	0.7485	759.3464
	6	[24, 42]	1	0.3	tanh	0.244	1318.8364
	12	[32]	1	0.3	tanh	0.2996	913.0587
Trabajadores	3	[32]	24	0.1	tanh	0.0075	20.8112
	6	[24, 42]	24	0.1	tanh	0.0058	5.6342
	12	[32]	24	0.1	relu	0.007	14.3802
Temperatura	6	[32]	24	0.1	tanh	0.0804	1.9593
	12	[24, 42]	12	0.1	relu	0.1896	6.6552
	24	[32]	12	0.3	tanh	0.169	7.2241

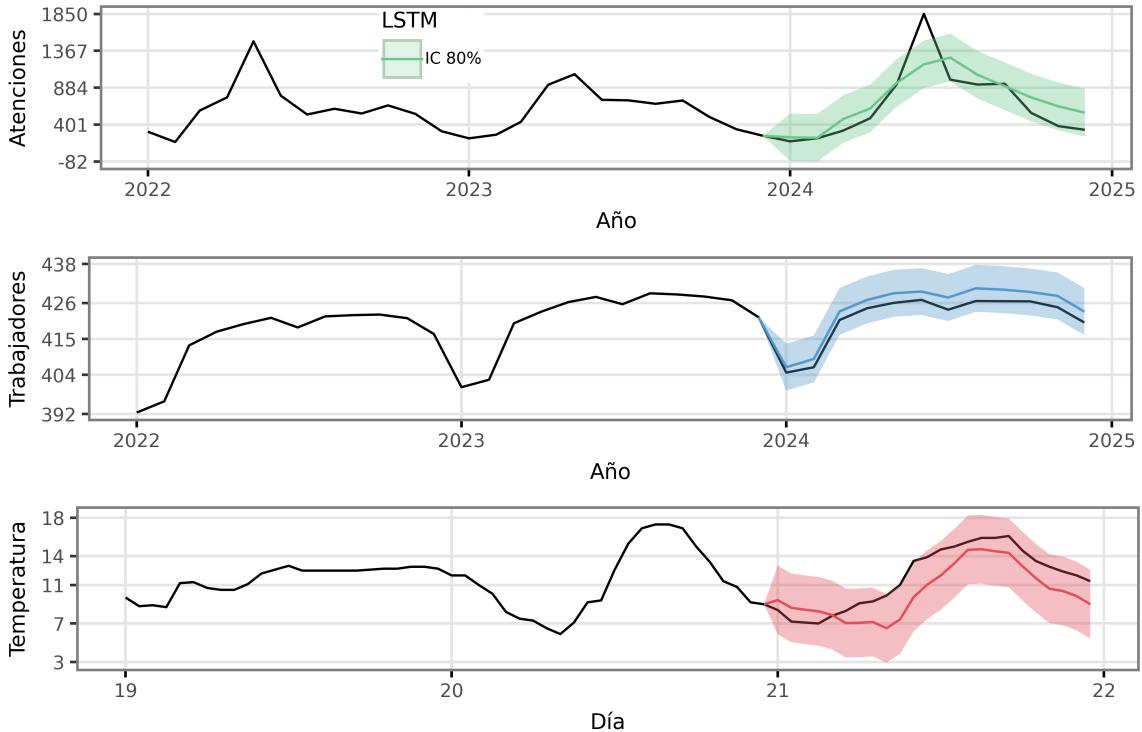


Figura 27: Pronósticos con LSTM.

Estos intervalos de confianza también fueron obtenidos gracias a *conformal predictions* pero se realizaron con funciones ya integradas en la librería **scalecast**.

4.2.4 Modelos fundacionales (TimeGPT y Chronos)

La ventaja de los modelos fundacionales por sobre las redes neuronales convencionales radica en que el ajuste de parámetros se realiza con un preentrenamiento en grandes conjuntos de datos. Esto significa que no es necesaria ninguna intervención manual. No es necesario definir características, ajustar parámetros, ni especificar la forma del modelo. Este modo de pronóstico se conoce como *zero-shot*.

Si se desea un ajuste más controlado sobre la serie se podría hacer uso del “ajuste fino” (*fine tuning*). El ajuste fino consiste en evaluar la función de perdida con los parámetros preestablecidos y realizar iteraciones extra de entrenamiento para ajustar el modelo específicamente al conjunto de datos dado con el objetivo de minimizar aún más el error del pronóstico. Sin embargo, luego de numerosas pruebas, esta herramienta no logró cambios significativos en el pronóstico de las series estudiadas por sobre las configuraciones base, logrando en ciertos casos resultados peores y aumentos excesivos los tiempos de procesamiento. Es por esto que no se utiliza esta característica en los resultados finales pero se deja propuesta para futuras aplicaciones.

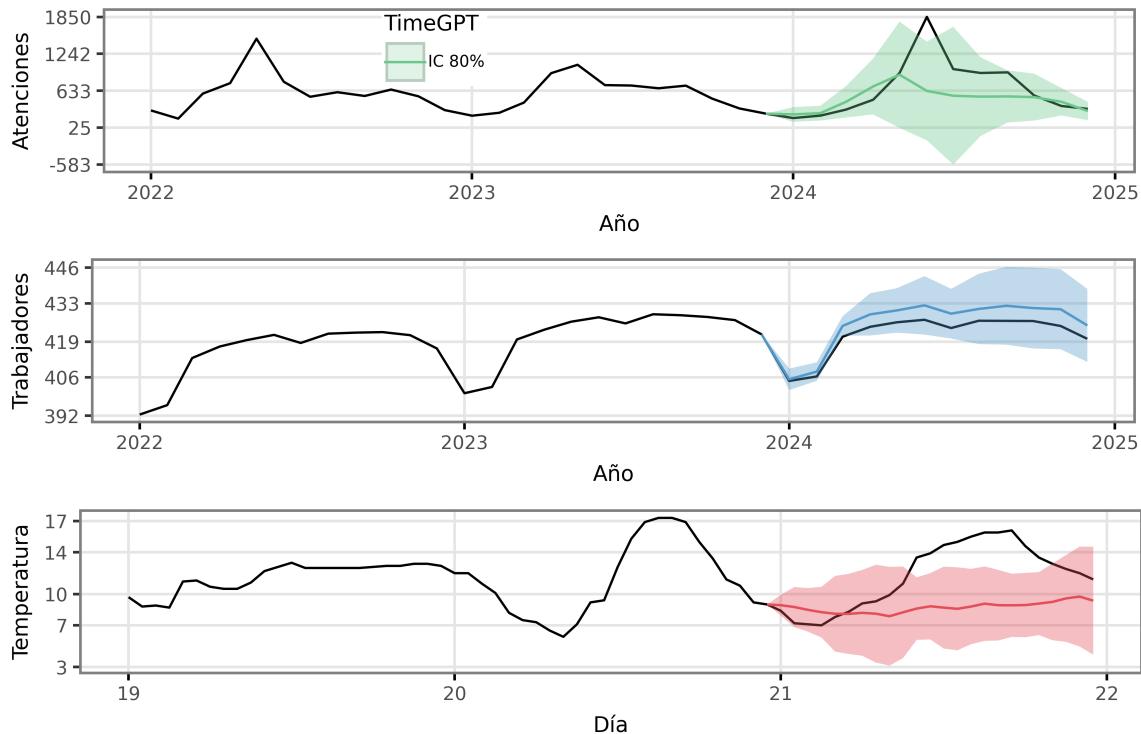


Figura 28: Pronósticos con TimeGPT.

Tabla 8: Métricas de evaluación para los ajustes con TimeGPT.

Serie	Hor.	MAPE	Interval Score
Atenciones	3	0.6542	1168.9787
	6	0.3651	793.6251
	12	0.3063	1276.4925
Trabajadores	3	0.0015	2.6232
	6	0.0025	11.1076

Serie	Hor.	MAPE	<i>Interval Score</i>
Temperatura	12	0.0101	19.6513
	3	0.238	9.212
	6	0.1219	5.1091
	12	0.2625	18.0171

Los pronósticos con TimeGPT sobre las series de temperatura y atenciones fueron regulares, en ninguno de los 2 casos pudo detectar los patrones estacionales de las series. Por otro lado, si pudo pronosticar correctamente las observaciones futuras en la serie de trabajadores registrados. Los problemas de pronóstico de TimeGPT parecen estar en el corto y largo plazo, ya que en la tabla 8 se puede ver como para los pronósticos a medio plazo las métricas de evaluación son menores que para corto y largo plazo.

Chronos al ser ser una familia de modelos cuenta con múltiples opciones para realizar pronósticos *zero-shot*. En esta tesina se utilizó el modelo **bolt-small**, el cual cuenta con 48 millones de parámetros.

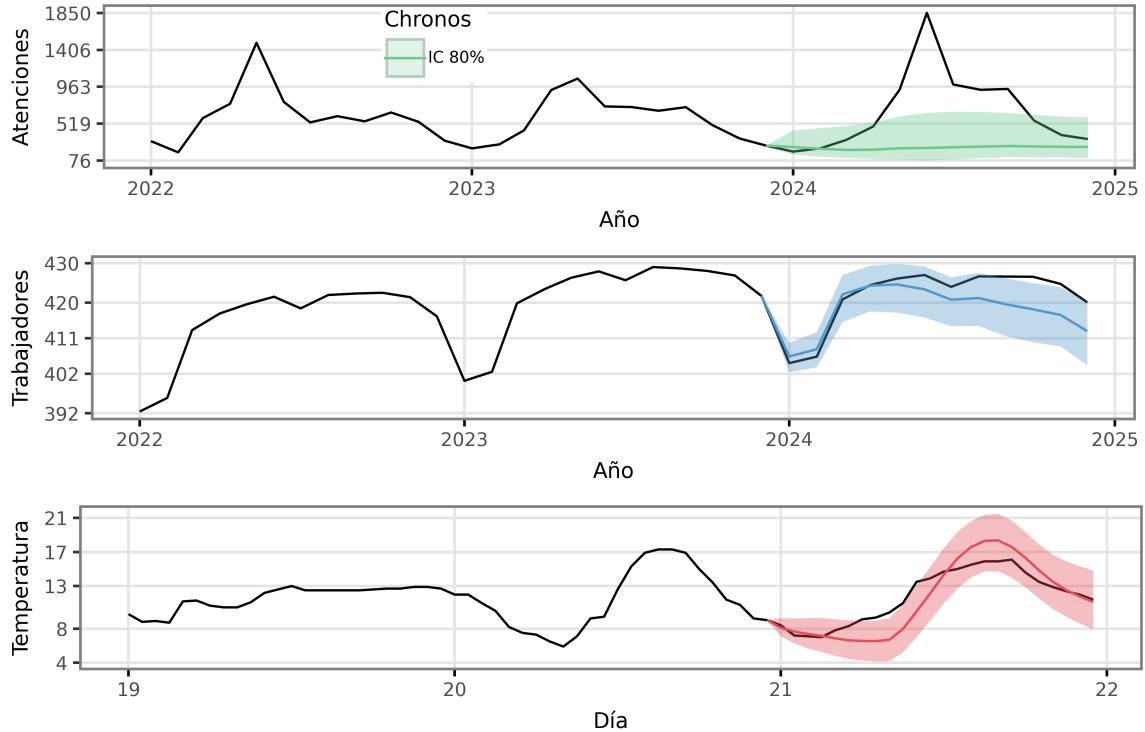


Figura 29: Pronósticos con Chronos.

Tabla 9: Métricas de evaluación para los ajustes con Chronos.

Serie	Hor.	MAPE	<i>Interval Score</i>
Atenciones	3	0.2312	338.2622
	6	0.4844	2961.5729
	12	0.5292	2479.0359
	3	0.0318	117.8766

Serie	Hor.	MAPE	<i>Interval Score</i>
Trabajadores	6	0.0212	67.4687
	12	0.0097	15.1275
Temperatura	3	0.3489	26.5617
	6	0.4502	49.5046
	12	0.125	6.2749

Chronos pudo detectar correctamente los cambios de temperatura en las 24 horas, pero por otro lado, hace un pésimo trabajo intentando pronosticar la serie de atenciones. En las dos series Chronos tiene un comportamiento inverso, mientras que en la serie de atenciones a medida que se incrementa el horizonte las métricas empeoran, para la serie de temperaturas mejoran. También se puede ver como el pronóstico sobre la serie de trabajadores es peor que con otros modelos.

4.3 Comparación de resultados y análisis final

Para las comparaciones en la primera serie se utiliza el modelo arima que mejor ajusto, este es el 2do modelo manual

Serie 1

	Modelo	Horizonte	MAPE	Interval Score	Tiempo
0	ARIMA		0.332828	2205.106225	6.652498
1	XGBoost	3	0.861528	1454.496494	0.527997
2	XGBoost	6	1.266862	1849.694872	0.814831
3	XGBoost	12	0.368438	1515.928887	0.689517
4	LightGBM	3	0.863908	1559.414681	0.680754
5	LightGBM	6	1.265549	1846.205816	0.665758
6	LightGBM	12	0.378141	1537.109101	0.525749
7	LSTM	3	0.748516	759.346380	22.389853
8	LSTM	6	0.243965	1318.836436	19.025867
9	LSTM	12	0.299630	913.058731	20.020719
10	TimeGPT	3	0.654243	1168.978727	1.897844
11	TimeGPT	6	0.365136	793.625100	1.276090
12	TimeGPT	12	0.306254	1276.492486	0.902465
13	Chronos	3	0.231202	338.262207	0.017940
14	Chronos	6	0.484421	2961.572856	0.021994
15	Chronos	12	0.529232	2479.035950	0.026991

Serie 2

	Modelo	Horizonte	MAPE	Interval Score	Tiempo
0	ARIMA	12	0.007843	15.533570	34.145489
1	XGBoost	3	0.008311	20.513982	0.866825
2	XGBoost	6	0.009068	20.513982	0.853317
3	XGBoost	12	0.006866	21.930821	0.995508
4	LightGBM	3	0.014368	32.417041	0.257646

	Modelo	Horizonte	MAPE	Interval Score	Tiempo
5	LightGBM	6	0.009150	23.880157	0.704516
6	LightGBM	12	0.008923	25.287597	0.766685
7	LSTM	3	0.007452	20.811196	17.835651
8	LSTM	6	0.005834	5.634199	15.575858
9	LSTM	12	0.007049	14.380212	21.808970
10	TimeGPT	3	0.001512	2.623167	0.959170
11	TimeGPT	6	0.002516	11.107600	1.135880
12	TimeGPT	12	0.010058	19.651347	1.349156
13	Chronos	3	0.031763	117.876635	0.026006
14	Chronos	6	0.021180	67.468704	0.025005
15	Chronos	12	0.009683	15.127507	0.028998

Serie 3

	Modelo	Horizonte	MAPE	Interval Score	Tiempo
0	ARIMA	24	0.241747	9.332746	536.194742
1	XGBoost	6	0.277503	35.663474	0.994445
2	XGBoost	12	0.299514	36.302561	0.757746
3	XGBoost	24	0.058265	2.763566	0.761943
4	LightGBM	6	0.264958	33.128715	1.564913
5	LightGBM	12	0.250733	29.873215	1.206677
6	LightGBM	24	0.093115	4.667262	0.500564
7	LSTM	6	0.080360	1.959271	34.023242
8	LSTM	12	0.189597	6.655205	24.162187
9	LSTM	24	0.168984	7.224146	16.002245
10	TimeGPT	6	0.238018	9.212010	3.285518
11	TimeGPT	12	0.121859	5.109111	1.046026
12	TimeGPT	24	0.262536	18.017061	1.133642
13	Chronos	6	0.348883	26.561669	0.048005
14	Chronos	12	0.450168	49.504552	0.024992
15	Chronos	24	0.124971	6.274898	0.036354

5. Conclusiones

A medida que se avanzó por el documento, las diferencias, las ventajas y desventajas de cada método se fueron evidenciando. Con ARIMA el ajuste fue completamente manual, tomó su requerido tiempo y aplicarlo a otras series implicaría empezar de cero. Luego, en los algoritmos de aprendizaje automático se tuvieron que definir una serie de características para ayudar al modelo a ajustarse automáticamente a los datos, si se deseara pronosticar otra serie, sería prudente cambiar las características a otras más acordes a los datos. Con LSTM simplemente se tuvo que definir la forma del modelo, y tanto con TimeGPT como con Chronos solo fue necesario brindarles los datos y su periodicidad. En estos últimos 3 modelos ningún cambio es necesario para pronosticar otra serie.

En este documento se presentan 3 contribuciones claves para el campo de la estadística. En primer lugar y como objetivo principal de esta tesina, la introducción de los modelos transformadores para el pronóstico de series temporales. Además se incluyeron aportes como evaluar el desempeño de los modelos con una nueva medida del error, el *interval score*, el cual no tiene en cuenta únicamente el pronóstico puntual sino también probabilístico. El último gran aporte es la construcción de intervalos de pronóstico por medio de *conformal predictions*, que no depende de conocer la distribución de los residuos.

6. Mejoras y extenciones a la investigación

En esta tesina se buscó abordar el tema de la forma más amplia posible sin sacrificar profundidad. Sin embargo, por la amplitud del mismo se dejaron fuera muchos temas interesantes que se podrían tratar en otros trabajos.

En primer lugar, se podrían estudiar numerosas otras series con distintas características y aplicar las respectivas correcciones a aquellas series acotadas, como el caso de la cantidad de atenciones en guardia que no puede ser negativa.

En la selección de los modelos se eligió como mejor aquel que minimizara el MAPE, pero se podría haber hecho la elección en base al *Interval Score* o alguna otra medida de error. También se podría indagar sobre múltiples medidas de error probabilísticas diferentes al *Interval Score*, tales como *Scaled quantile loss*, *Weighted quantile loss* o *Implicit quantile loss*.

Boosting y el ensamblaje de modelos no está limitado únicamente a los árboles de decisión, y se podría explorar como funcionan estas técnicas en otros modelos, como en redes neuronales.

Para obtener pronósticos probabilísticos en los algoritmos de aprendizaje automático se optó por EnbPI, pero queda propuesto probar otros métodos o alternativas, como *Natural Gradient Boosting* (NGBoost).

Otra expansión a la investigación se puede dar en el ajuste de hiperparámetros, y características en el caso de los algoritmos de aprendizaje automatizado. Si bien con la búsqueda de parámetros se intentó explorar múltiples opciones, por cuestiones de tiempo y exigencia computacional es imposible explorarlas todas, es por esto que aún queda un amplio campo de investigación en este aspecto. A su vez, en los modelos fundacionales también es posible indagar sobre el ajuste fino.

7. Bibliografía

- Alammar, J.** (27 de junio de 2018). *The Illustrated Transformer*. Jay Alammar. <https://jalammar.github.io/illustrated-transformer/>
- Ansari et al.** (2024). *Chronos: Learning the Language of Time Series*. Transactions on Machine Learning Research. <https://arxiv.org/abs/2403.07815>
- Awan, A.** (2 de septiembre de 2024). *Time Series Forecasting With TimeGPT*. Datacamp. <https://www.datacamp.com/tutorial/time-series-forecasting-with-time-gpt>
- Bermejo, J.** (21 de mayo de 2024). *Redes neuronales*. Facultad de Ciencias Económicas y Estadística de la Universidad Nacional de Rosario.
- Chen, Y., Yao, X.** (2023). *Conformal prediction for time series*. Proceedings of Machine Learning Research. <https://arxiv.org/abs/2010.09107>
- Elhariri, K.** (1 de marzo de 2022). *The Transformer Model*. Medium. <https://medium.com/data-science/attention-is-all-you-need-e498378552f9>
- GeeksforGeeks.** (s.f.). *What is LSTM – Long short term memory?*. Recuperado el 15 de julio de 2025 de <https://www.geeksforgeeks.org/deep-learning/deep-learning-introduction-to-long-short-term-memory/>
- Gilliland, M., Sglavo, U., & Tashman, L.** (2016). *Forecast Error Measures: Critical Review and Practical Recommendations*. John Wiley & Sons Inc.
- Gneiting, T., & Raftery A. E.** (2007). *Strictly Proper Scoring Rules, Prediction, and Estimation*. Journal of the American Statistical Association, 102(477), 359–378. <https://doi.org/10.1198/016214506000001437>
- Hyndman, R. J., & Athanasopoulos, G.** (2021). *Forecasting: principles and practice (3rd ed.)*. OTexts. <https://otexts.com/fpp3/>
- Hyndman, R.J., Athanasopoulos, G., Garza, A., Challu, C., Mergenthaler, M., & Olivares, K.G.** (2024). *Forecasting: Principles and Practice, the Pythonic Way*. OTexts. [OTexts.com/fpppy](https://otexts.com/fpppy).
- IBM.** (s.f.). *Explainers*. Recuperado el 14 de marzo de 2025 de <https://www.ibm.com/think/topics>
- Kamtziris, G.** (27 de febrero de 2023). *Time Series Forecasting with XGBoost and LightGBM: Predicting Energy Consumption*. Medium. <https://medium.com/@geokam/time-series-forecasting-with-xgboost-and-lightgbm-predicting-energy-consumption-460b675a9cee>
- Korstanje, J.** (2021). *Advanced Forecasting with Python*. Apress.
- Nielsen, A.** (2019). *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*. O'Reilly Media.
- Nixtla.** (s.f.). *About TimeGPT*. Recuperado en diciembre de 2024 de https://docs.nixtla.io/docs/getting-started-about_timegpt
- Parmezan, A., Souza, V., & Batista, G.** (1 de mayo de 2019). *Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model*. Information Sciences. <https://www.sciencedirect.com/science/article/abs/pii/S0020025519300945>

Prunello, M., & Marfetán, D. (12 de mayo de 2024). *Árboles de decisión*. Facultad de Ciencias Económicas y Estadística de la Universidad Nacional de Rosario.

Sabino Parmezan, A. R., Souza, V. M. A., & Batista, G. E. A. P. A. (2019). *Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model*. Information Sciences, 484, 302–337. <https://doi.org/10.1016/j.ins.2019.01.076>

Sanderson, G. [3Blue1Brown]. (2024). *Attention in transformers, step-by-step / DL6* [Video]. Youtube. <https://www.youtube.com/watch?v=eMlx5fFNoYc&t=1204s>

Sanderson, G. [3Blue1Brown]. (2024). *Transformers (how LLMs work) explained visually / DL5* [Video]. Youtube. <https://www.youtube.com/watch?v=wjZofJX0v4M>

Shastri, Y. (26 de abril de 2024). *Attention Mechanism in LLMs: An Intuitive Explanation*. Datacamp. <https://www.datacamp.com/blog/attention-mechanism-in-lmms-intuition>

Silberstein, E. (7 de noviembre de 2024). *Tracing the Transformer in Diagrams*. Medium. <https://medium.com/data-science/tracing-the-transformer-in-diagrams-95dbeb68160c>

Valeriy, M. (11 de agosto de 2023). *Demystifying EnbPI: Mastering Conformal Prediction Forecasting*. Medium. <https://valeman.medium.com/demystifying-enbpi-mastering-conformal-prediction-forecasting-d49e65532416>

Vaswani et al. (2017). *Attention is all you need*. Google. <https://arxiv.org/pdf/1706.03762>

8. Anexo

8.1 Gráficos estacionales

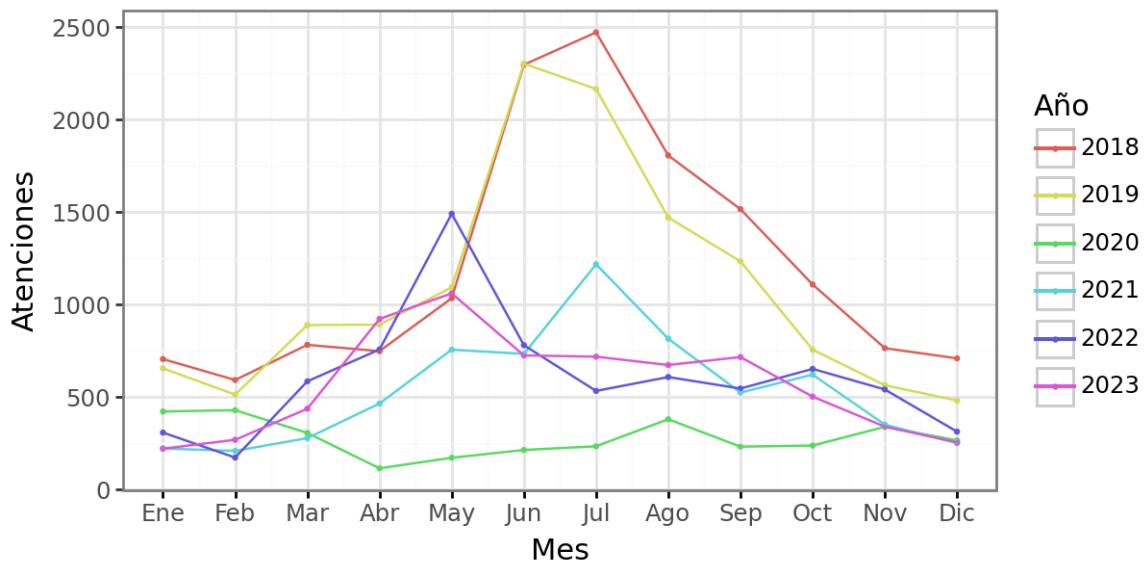


Figura 30: Atenciones por guardia mensuales por patologías respiratorias por año

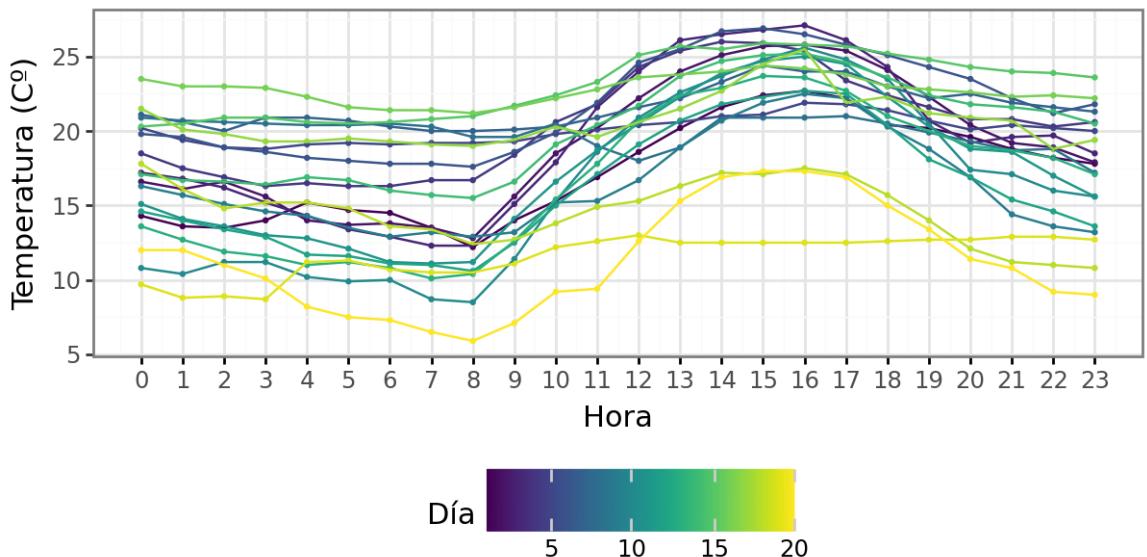


Figura 31: Atenciones por guardia mensuales por patologías respiratorias por año

8.2 Salidas de modelos arima

Atenciones modelo 1

```
SARIMAX Results
=====
Dep. Variable:                               y      No. Observations:    72
Model: SARIMAX(0, 1, 1)x(0, 1, [ ], 12)   Log Likelihood: -425.795
                                               

```

Date: Mon, 21 Jul 2025 AIC 857.589
 Time: 09:44:39 BIC 863.822
 Sample: 0 HQIC 860.022
 - 72
 Covariance Type: opg
 =====

	coef	std err	z	P> z	[0.025	0.975]
intercept	-0.3257	44.115	-0.007	0.994	-86.789	86.137
ma.L1	-0.0577	0.129	-0.448	0.654	-0.310	0.195
sigma2	1.086e+05	1.56e+04	6.973	0.000	7.81e+04	1.39e+05

Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB): 16.54
 Prob(Q): 0.97 Prob(JB): 0.00
 Heteroskedasticity (H): 0.90 Skew: -0.80
 Prob(H) (two-sided): 0.82 Kurtosis: 5.04
 =====

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Atenciones modelo 2

SARIMAX Results

Dep. Variable:	y	No. Observations:
Model:	SARIMAX(0, 1, 0)x(0, 1, [1], 12)	Log Likelihood -423.03
Date:	Mon, 21 Jul 2025	AIC 852.062
Time:	09:44:39	BIC 858.298
Sample:	0	HQIC 854.498
	- 72	
Covariance Type:	opg	

	coef	std err	z	P> z	[0.025	0.975]
intercept	0.6941	33.473	0.021	0.983	-64.911	66.299
ma.S.L12	-0.2919	0.107	-2.720	0.007	-0.502	-0.082
sigma2	9.8e+04	1.26e+04	7.789	0.000	7.33e+04	1.23e+05

Ljung-Box (L1) (Q): 0.13 Jarque-Bera (JB): 26.75
 Prob(Q): 0.72 Prob(JB): 0.00
 Heteroskedasticity (H): 0.79 Skew: -0.85
 Prob(H) (two-sided): 0.61 Kurtosis: 5.82
 =====

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Atenciones modelo selección automática

SARIMAX Results

```
=====
Dep. Variable:                      y      No. Observations:                 72
Model:                SARIMAX(0, 1, 0)x(1, 0, 0, 12)   Log Likelihood:            -504.749
Date:                  Mon, 21 Jul 2025     AIC:                         1013.497
Time:                      09:44:39       BIC:                         1018.023
Sample:                   0 - 72        HQIC:                        1015.297
Covariance Type:                opg
=====
```

	coef	std err	z	P> z	[0.025	0.975]
ar.S.L12	0.5009	0.067	7.487	0.000	0.370	0.632
sigma2	8.341e+04	1.11e+04	7.542	0.000	6.17e+04	1.05e+05

```
=====
Ljung-Box (L1) (Q):                  0.41    Jarque-Bera (JB):                  16.65
Prob(Q):                           0.52    Prob(JB):                            0.00
Heteroskedasticity (H):              0.65    Skew:                                0.60
Prob(H) (two-sided):                0.31    Kurtosis:                            5.05
=====
```

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Trabajadores modelo 1

SARIMAX Results

```
=====
Dep. Variable:                      y      No. Observations:                 96
Model:                SARIMAX(0, 1, 1)x(1, 1, [], 12)   Log Likelihood:            -175.940
Date:                  Mon, 21 Jul 2025     AIC:                         359.880
Time:                      09:44:39       BIC:                         369.556
Sample:                   0 - 96        HQIC:                        363.767
Covariance Type:                opg
=====
```

	coef	std err	z	P> z	[0.025	0.975]
intercept	-0.0347	0.314	-0.111	0.912	-0.649	0.580
ma.L1	0.3242	0.110	2.958	0.003	0.109	0.539
ar.S.L12	-0.3532	0.081	-4.337	0.000	-0.513	-0.194
sigma2	3.9792	0.467	8.514	0.000	3.063	4.895

```
=====
Ljung-Box (L1) (Q):                  0.12    Jarque-Bera (JB):                  16.66
Prob(Q):                           0.73    Prob(JB):                            0.00
Heteroskedasticity (H):              1.28    Skew:                                -0.23
Prob(H) (two-sided):                0.52    Kurtosis:                            5.15
=====
```

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Trabajadores modelo selección automática

```
SARIMAX Results
=====
Dep. Variable:                      y      No. Observations:                 96
Model:                SARIMAX(2, 0, 0)x(2, 1, 0, 12)   Log Likelihood:            -173.349
Date:                  Mon, 21 Jul 2025     AIC:                         356.697
Time:                      09:44:39         BIC:                         368.851
Sample:                   0 - 96          HQIC:                        361.583
Covariance Type:                opg
=====
              coef    std err       z   P>|z|      [0.025      0.975]
-----
ar.L1        1.3724    0.090    15.176      0.000      1.195      1.550
ar.L2       -0.3937    0.081    -4.890      0.000     -0.552     -0.236
ar.S.L12     -0.5199    0.097    -5.373      0.000     -0.710     -0.330
ar.S.L24     -0.3363    0.119    -2.827      0.005     -0.570     -0.103
sigma2       3.3072    0.426     7.767      0.000      2.473      4.142
=====
Ljung-Box (L1) (Q):                  0.14  Jarque-Bera (JB):             13.80
Prob(Q):                           0.71  Prob(JB):                     0.00
Heteroskedasticity (H):               1.04  Skew:                       -0.32
Prob(H) (two-sided):                 0.92  Kurtosis:                    4.88
=====
```

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Temperatura modelo selección automática

```
SARIMAX Results
=====
Dep. Variable:                      y      No. Observations:                 480
Model:                SARIMAX(1, 1, 1)x(2, 0, 1, 24)   Log Likelihood:            -524.526
Date:                  Mon, 21 Jul 2025     AIC:                         1067.051
Time:                      09:44:39         BIC:                         1104.597
Sample:                   0 - 480          HQIC:                        1081.811
Covariance Type:                opg
=====
              coef    std err       z   P>|z|      [0.025      0.975]
-----
intercept   -0.0009    0.006    -0.135      0.892     -0.013      0.012
HUM        -0.0076    0.001    -5.217      0.000     -0.010     -0.005
PNM        -0.1338    0.061    -2.210      0.027     -0.253     -0.015
ar.L1        0.2251    0.097     2.324      0.020      0.035      0.415
ma.L1        0.1805    0.097     1.857      0.063     -0.010      0.371
ar.S.L24     0.9792    0.087    11.194      0.000      0.808      1.151
ar.S.L48     -0.0339    0.069    -0.490      0.624     -0.170      0.102
=====
```

ma.S.L24	-0.7494	0.074	-10.152	0.000	-0.894	-0.605
sigma2	0.4960	0.025	19.884	0.000	0.447	0.545
<hr/>						
Ljung-Box (L1) (Q):	0.03	Jarque-Bera (JB):	102.24			
Prob(Q):	0.87	Prob(JB):	0.00			
Heteroskedasticity (H):	1.09	Skew:	-0.08			
Prob(H) (two-sided):	0.59	Kurtosis:	5.26			
<hr/>						

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Temperatura modelo 1

SARIMAX Results

Dep. Variable:	y	No. Observations:	480			
Model:	SARIMAX(1, 1, 1)x(1, 0, 1, 24)	Log Likelihood	-515.068			
Date:	Mon, 21 Jul 2025	AIC	1046.136			
Time:	09:44:39	BIC	1079.509			
Sample:	0 - 480	HQIC	1059.255			
Covariance Type:	opg					
<hr/>						
	coef	std err	z	P> z	[0.025	0.975]
intercept	-4.422e-05	0.001	-0.037	0.971	-0.002	0.002
HUM	-0.0072	0.001	-5.035	0.000	-0.010	-0.004
PNM	-0.2318	0.062	-3.755	0.000	-0.353	-0.111
ar.L1	0.5224	0.084	6.246	0.000	0.358	0.686
ma.L1	-0.1066	0.089	-1.193	0.233	-0.282	0.068
ar.S.L24	0.9869	0.011	90.703	0.000	0.966	1.008
ma.S.L24	-0.8863	0.049	-17.915	0.000	-0.983	-0.789
sigma2	0.4682	0.024	19.347	0.000	0.421	0.516
<hr/>						
Ljung-Box (L1) (Q):	0.02	Jarque-Bera (JB):	118.46			
Prob(Q):	0.88	Prob(JB):	0.00			
Heteroskedasticity (H):	1.08	Skew:	-0.06			
Prob(H) (two-sided):	0.64	Kurtosis:	5.43			
<hr/>						

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Temperatura modelo 2

SARIMAX Results

Dep. Variable:	y	No. Observations:	480
Model:	SARIMAX(1, 1, 0)x(2, 0, [1], 24)	Log Likelihood	-517.974

Date: Mon, 21 Jul 2025 AIC 1051.94
 Time: 09:44:39 BIC 1085.32
 Sample: 0 HQIC 1065.06
 - 480

Covariance Type: opg

	coef	std err	z	P> z	[0.025	0.975]
intercept	0.0003	0.002	0.150	0.880	-0.004	0.004
HUM	-0.0077	0.001	-5.315	0.000	-0.011	-0.005
PNM	-0.1264	0.064	-1.980	0.048	-0.251	-0.001
ar.L1	0.4565	0.035	13.071	0.000	0.388	0.525
ar.S.L24	1.0639	0.093	11.453	0.000	0.882	1.246
ar.S.L48	-0.0892	0.070	-1.267	0.205	-0.227	0.049
ma.S.L24	-0.8827	0.083	-10.609	0.000	-1.046	-0.720
sigma2	0.5096	0.028	18.016	0.000	0.454	0.565

Ljung-Box (L1) (Q): 0.44 Jarque-Bera (JB): 126.11
 Prob(Q): 0.51 Prob(JB): 0.00
 Heteroskedasticity (H): 1.03 Skew: -0.00
 Prob(H) (two-sided): 0.86 Kurtosis: 5.51

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Temperatura modelo 3

SARIMAX Results

Dep. Variable:	y	No. Observations:	480
Model:	SARIMAX(1, 1, 0)x(1, 1, 0, 24)	Log Likelihood	-545.785
Date:	Mon, 21 Jul 2025	AIC	1103.570
Time:	09:44:39	BIC	1128.292
Sample:	0	HQIC	1113.309
	- 480		
Covariance Type:	opg		

	coef	std err	z	P> z	[0.025	0.975]
intercept	-0.0075	0.039	-0.195	0.845	-0.083	0.068
HUM	-0.0071	0.002	-4.181	0.000	-0.010	-0.004
PNM	-0.0905	0.070	-1.290	0.197	-0.228	0.047
ar.L1	0.3353	0.035	9.558	0.000	0.267	0.404
ar.S.L24	-0.4432	0.034	-12.878	0.000	-0.511	-0.376
sigma2	0.6372	0.032	20.082	0.000	0.575	0.699

Ljung-Box (L1) (Q): 0.04 Jarque-Bera (JB): 70.82
 Prob(Q): 0.84 Prob(JB): 0.00
 Heteroskedasticity (H): 1.29 Skew: -0.18
 Prob(H) (two-sided): 0.12 Kurtosis: 4.90

=====
Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

8.3 Comprobación de supuestos de modelos arima

Modelo atenciones selección automática

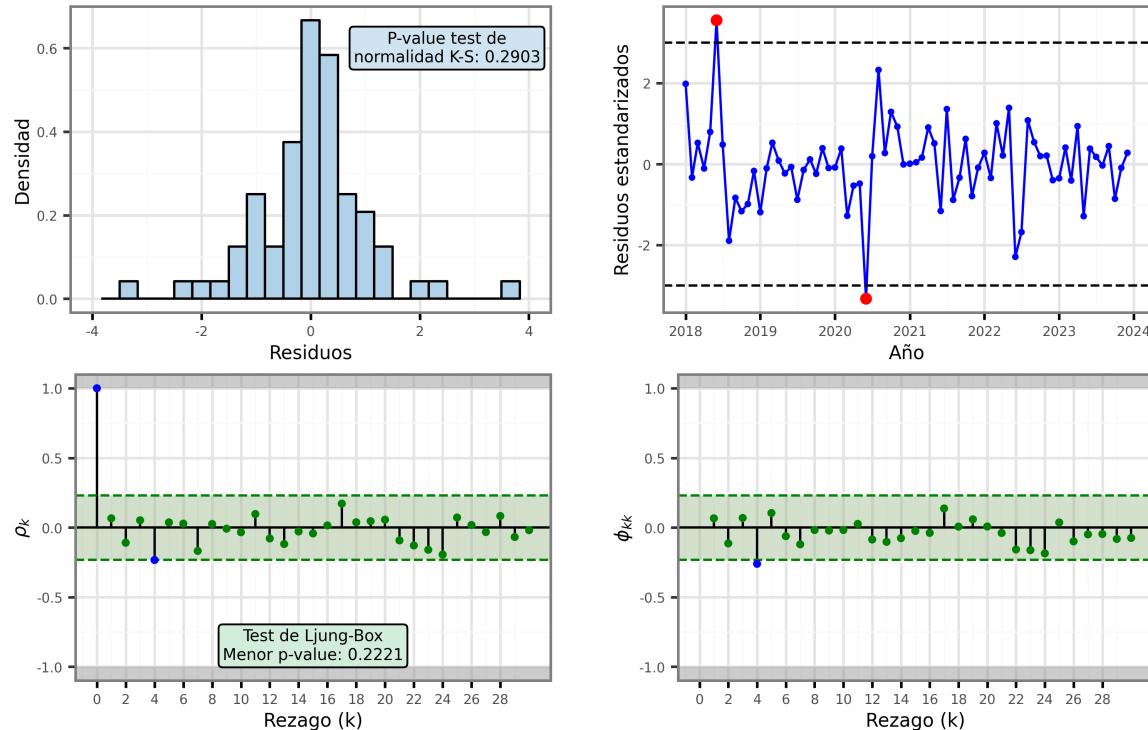


Figura 32: Comprobación de supuestos del modelo arima automático para la serie de atenciones

Modelo trabajadores 1

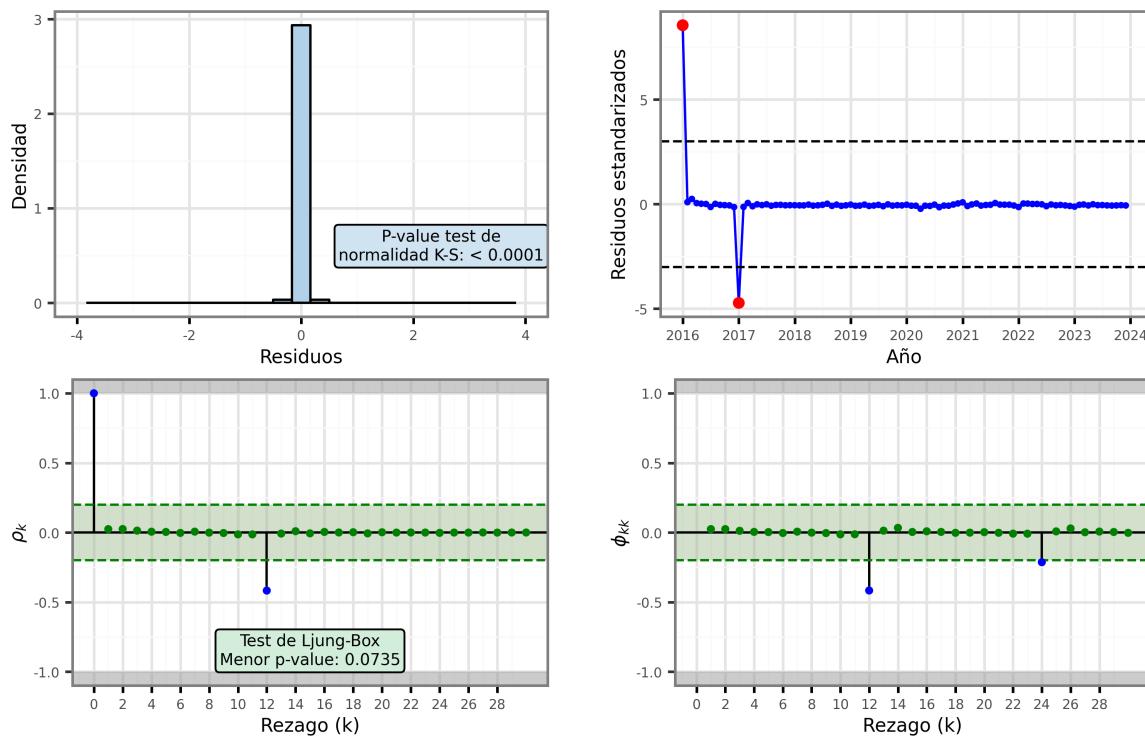


Figura 33: Comprobación de supuestos del modelo arima manual para la serie de trabajadores

Modelo trabajadores selección automática

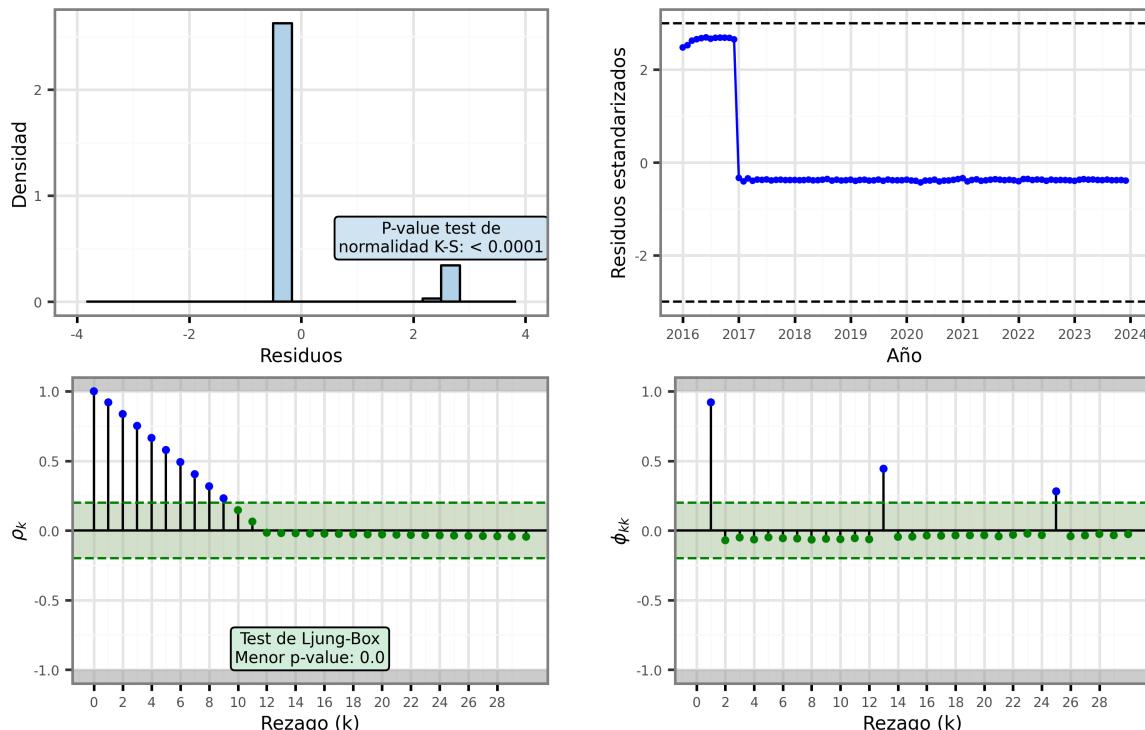


Figura 34: Comprobación de supuestos del modelo arima automático para la serie de trabajadores

Modelo Temperatura 3

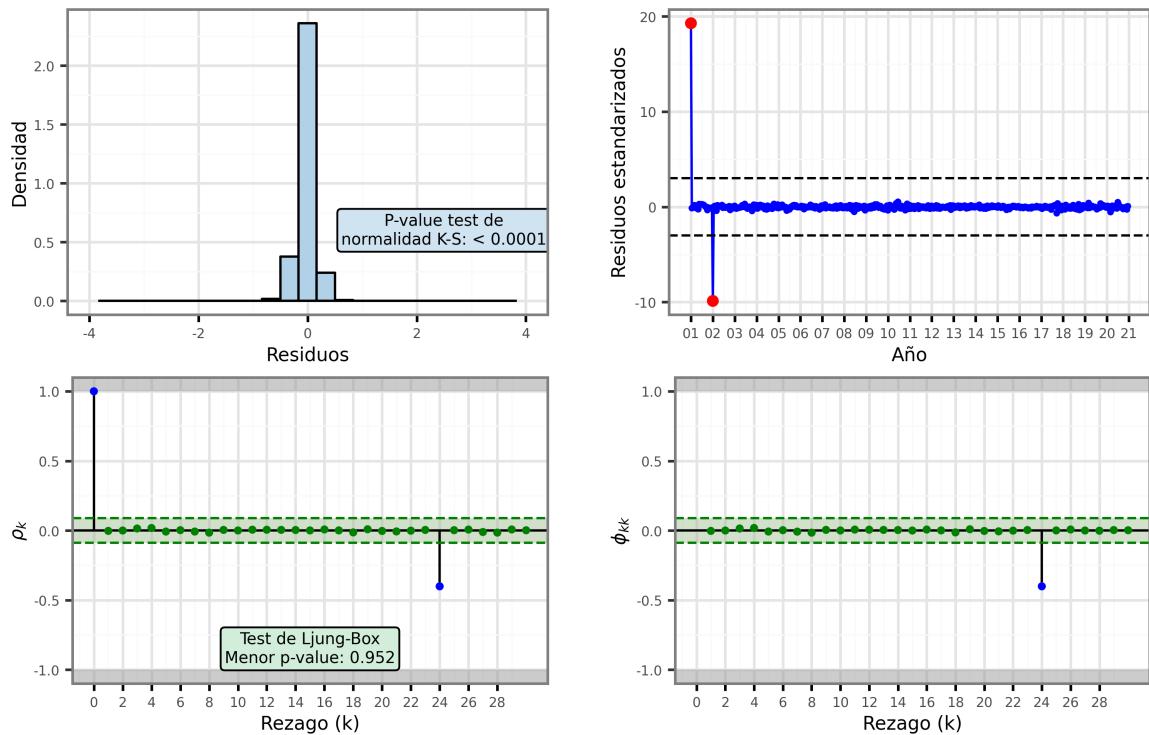


Figura 35: Comprobación de supuestos del tercer modelo arima manual para la serie de temperatura

NOTAS

- TO DO Redactar conclusiones de tablas controlar seleccion de hiperparametros Redactar Revisar Arima contar como funcionan las autocorrelaciones y parciales para la seleccion de modelos arima revisar transformers revisar medidas de evaluacion Eliminar la instalacion de neuralforecast Volver a correr toda la aplicacion y verificar resultados verificar luego de la aplicacion cuales son las librerias que termino usando Verificar, actualizar y completar las descripciones de las funciones Actualizar los requirements
- Guardar las versiones de librerias, software y hardware en la que se corre el codigo
- Hacer equivalencia del documento en GitBook

nomenclatura: - conjunto de observaciones $z_1, \dots, z_t, \dots, z_n$ - pronostico $z_{n+1}, \dots, z_{n+l}, \dots, z_{n+h}$