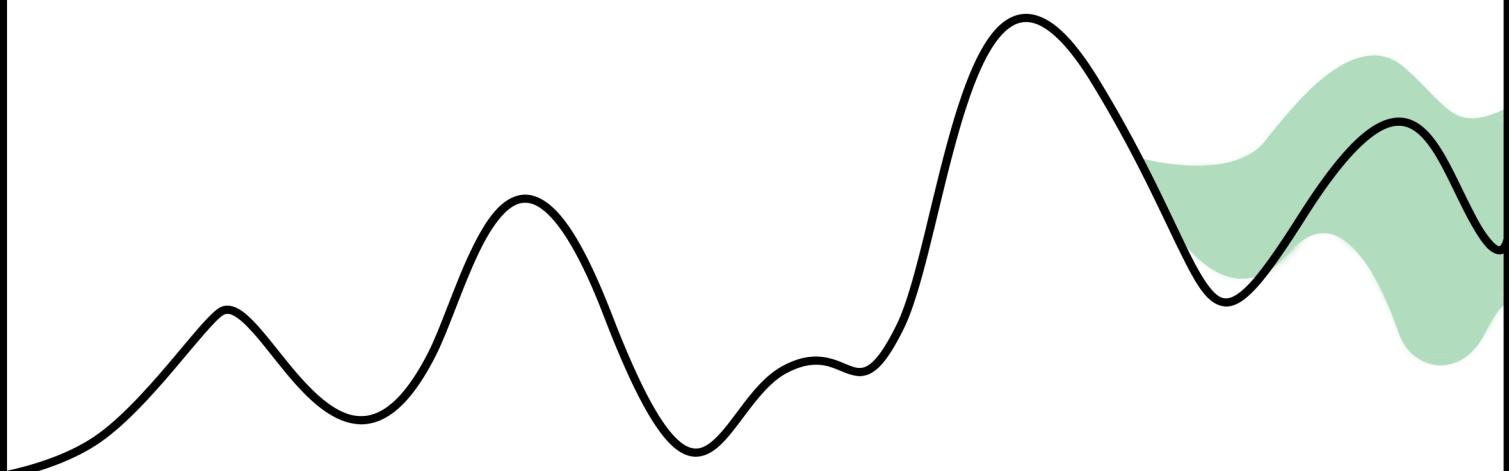


Comparación del desempeño de modelos estadísticos tradicionales, de aprendizaje automático y aprendizaje profundo para la predicción de series temporales

Tesina de grado



Alumno: Roncaglia Andrés Iván

Directora: Mag. Méndez Fernanda

Carrera: Licenciatura en Estadística



UNR Universidad
Nacional de Rosario

Agradecimientos

Este trabajo no solo es el cierre de mi paso por la carrera sino también de una de las mejores etapas de mi vida. Estuvo llena de crecimiento, esfuerzo y alegría. Decir que aprendí mucho académicamente es desmerecer todos estos años, porque también estuvieron acompañados de enseñanzas sobre la vida misma. Y como crecer no es algo que uno pueda lograr sin ayuda de nadie, está claro que debo agradecer a un gran puñado de personas.

En primer lugar a mis papás, que me dejaron venir desde lejos a pesar de que no les gustara separarse, que hicieron que no tenga que preocuparme por llegar a fin de mes, y que nunca me presionaron ni me exigieron nada.

A mis hermanos, al que tengo lejos por darme siempre mensajes de ánimo y preocuparse de que no me faltara nada, y al que tengo cerca por ser un gran compañero de piso a pesar de mis constantes quejas.

A mi novia le tengo que agradecer por estar a mi lado, en las buenas y en las malas, por levantarme el ánimo cuando me iba mal o celebrar juntos si me iba bien, por hacerme empujar siempre para adelante y obligarme a nunca bajar los brazos.

Uno de mis más grandes agradecimientos tiene que ir a todos los amigos que hice durante la carrera, tanto para con los que compartí los primeros años como para con los que compartí los últimos. Sin ellos probablemente me hubiera rendido mil y una veces, pero gracias a saber que no estaba solo y que siempre había alguien dispuesto a ayudar pude seguir adelante y llegar hasta donde estoy hoy.

También debo agradecer a todos los profesores que tuve, que sin ellos nada de esto sería posible. Fueron los que me abrieron las puertas al conocimiento y me llenaron de curiosidad, aprendí muchísimo de cada uno de ellos y muchos se convirtieron en mi rol a seguir como profesional.

Muchas gracias especialmente a Nanda, mi directora en este trabajo, que fue ella quien decidió apostar por mí para desarrollar este tema que me encantó investigar. Siempre estuvo disponible cuando la necesité y movió todos los hilos para que mi trabajo se desarrollara de la forma más suave y rápida posible.

Por último pero no menos importante, muchas gracias a la Universidad Nacional de Rosario, y en especial a la Facultad de Ciencias Económicas y Estadística, que brinda el lugar, el material y las oportunidades para que uno pueda crecer sin techo alguno.

Este trabajo no es solo mío, tiene una pizca de todos los que me acompañaron estos duros pero increíbles años, muchas gracias a cada uno de ustedes.

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 por hora 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*).

Los resultados muestran que no existe un modelo universalmente superior: mientras los modelos estadísticos ofrecen interpretabilidad y precisión, por otro lado son poco automatizables y requieren de un control manual que requiere tiempo y conocimiento. Los algoritmos de *boosting* presentan buena capacidad predictiva con bajo costo computacional, pero necesitan especial cuidado al seleccionar con qué características entrenar el modelo y no ofrecen intervalos probabilísticos de forma nativa. Las redes LSTM, aunque potentes, muestran sensibilidad al sobreajuste y son exigentes computacionalmente. Por su parte, los modelos fundacionales ofrecen una alternativa rápida para personas que no tienen profundos conocimientos en el pronóstico de series de tiempo, aunque son modelos poco personalizables 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	4
3.2.1 SARIMA	4
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 predicción en algoritmos de aprendizaje automático	9
3.4 Modelos de aprendizaje profundo	10
3.4.1 Introducción a redes neuronales	10
3.4.2 <i>Long Short Term Memory</i> (LSTM)	12
3.4.3 Modelos transformadores	14
3.4.4 Diferencias entre TimeGPT y Chronos	18
3.5 Métricas de evaluación	19
3.6 Selección de parámetros y validación del modelo	20
4. Aplicación	21
4.1 Series analizadas	21
4.2 Ajuste y evaluación de modelos	26
4.2.1 Modelización con ARIMA y SARIMAX	26
4.2.2 Modelos de aprendizaje automático	33
4.2.3 Redes neuronales (LSTM)	36
4.2.4 Modelos fundacionales (TimeGPT y Chronos)	38
4.3 Comparación de resultados y análisis final	41
5. Conclusiones	46
6. Mejoras y extensiones a la investigación	47
7. Bibliografía	48
8. Anexo	50
8.1 Salidas de modelos ARIMA	50

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, requieren de amplios conocimientos para encontrar un buen ajuste. Además, los modelos tradicionales como ARIMA son difíciles de automatizar, mientras que los algoritmos de aprendizaje automatizado que se utilizan actualmente pueden demandar largos tiempos de entrenamiento. 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 aprendizaje automático (*machine learning*) y modelos de aprendizaje profundo (*deep learning*). 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 aprendizaje automático y modelos de aprendizaje profundo, 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 para 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.

Otro concepto importante en series de tiempo es la estacionariedad. 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.

La falta de estacionariedad en la varianza se puede corregir mediante la familia de transformaciones de Box-Cox, definida como:

$$w_t = \begin{cases} \log(z_t) & \text{si } \lambda = 0 \\ \frac{z_t^\lambda - 1}{\lambda} & \text{si } \lambda \neq 0 \end{cases} \quad (1)$$

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

Los modelos *ARIMA* (*AutoRegressive Integrated Moving Average*) son unos de los modelos de pronóstico tradicionales mejor establecidos. Son una generalización de los modelos autorregresivos (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 períodos anteriores.

Formalmente un modelo *ARIMA*(p, d, q) se define de la siguiente manera:

$$\psi_p(B)(1 - B)^d z_t = \theta_0 + \theta_q(B)\alpha_t \quad (2)$$

Donde z_t es la observación t -ésima, $\psi_p(B)$ y $\theta_q(B)$ son funciones polinómicas de los rezagos (B), correspondientes a la parte autorregresiva y promedio móvil respectivamente, d es el grado de diferenciación, θ_0 es una constante y α_t es el error de la t -ésima observación. Se entiende por rezago a la distancia ordinal entre dos observaciones

Se deben tener en cuenta los siguientes 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.
- El proceso *ARMA* resultante haber aplicado las diferenciaciones necesarias al proceso *ARIMA* es estacionario e invertible si sus componentes *AR* y *MA* satisfacen, respectivamente, las condiciones de estacionariedad e invertibilidad.

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 que grafica la correlación 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:

$$\rho_k = \text{Corr}(z_t, z_{t+k}) = \frac{\text{Cov}(z_t, z_{t+k})}{\sqrt{\text{Var}(z_t) \cdot \text{Var}(z_{t+k})}} \quad (3)$$

y

$$\phi_{kk} = \text{Corr}(z_t, z_{t+k} | z_{t+1}, \dots, z_{t+k-1}) \quad (4)$$

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.

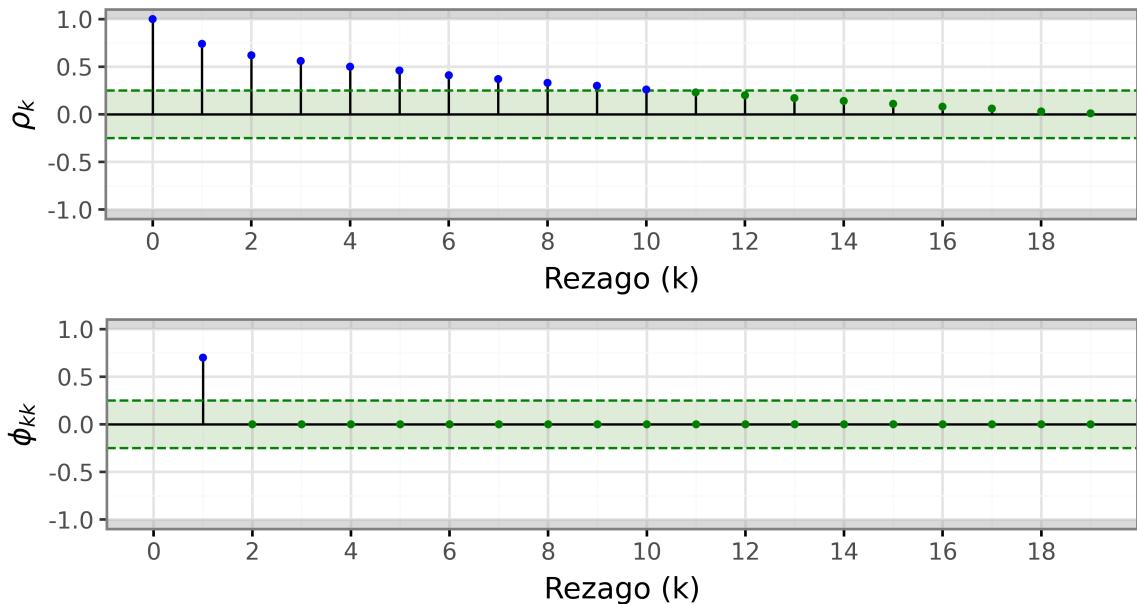


Figura 1: Ejemplo de las autocorrelaciones de un proceso *AR*(1).

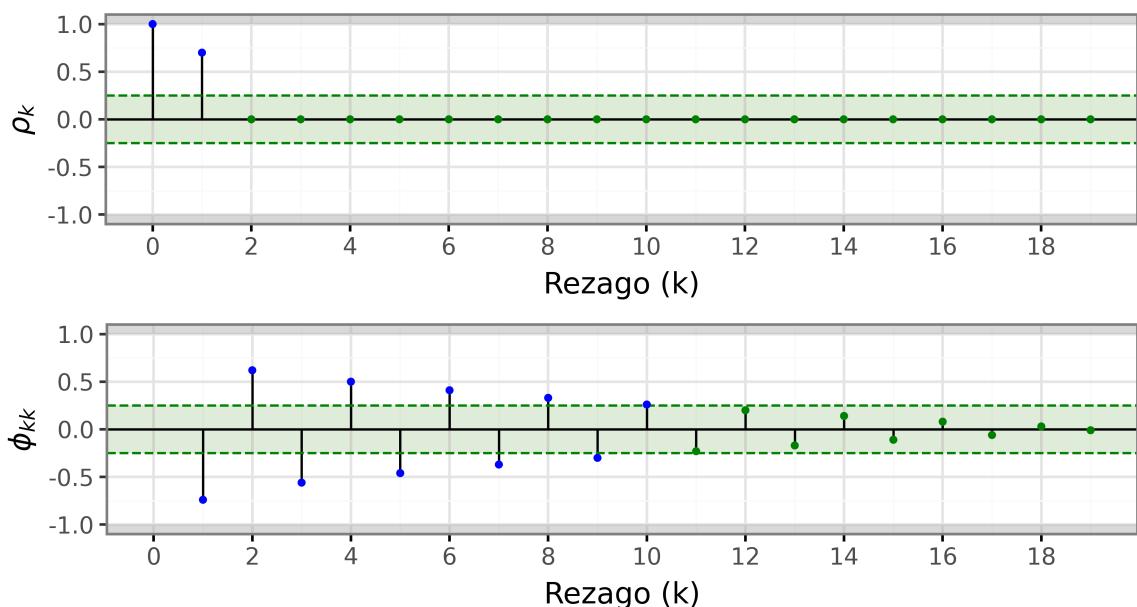


Figura 2: Ejemplo de las autocorrelaciones de un proceso *MA*(1).

Para poder identificar un modelo *SARIMA* a partir de las autocorrelaciones es necesario que la serie sea estacionaria. Entre los *tests* para probar la estacionariedad de una serie se encuentran el *test* aumentado de Dickey-Fuller y el de Kwiatkowski-Phillips-Schmidt-Shin. El *test* aumentado de Dickey-Fuller plantea como hipótesis nula que existe una raíz unitaria, indicando que la serie no es estacionaria y es necesaria una diferenciación, ante la alternativa de que no existe una raíz unitaria. Por otro lado, en el *test* Kwiatkowski-Phillips-Schmidt-Shin la hipótesis nula indica que la serie es estacionaria alrededor de una constante.

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 tiene pocos parámetros y son significativos.
- Es estable en los parámetros, que se cumple cuando las correlaciones entre los parámetros no son altas.

Existen múltiples opciones para probar la normalidad de una distribución. En este trabajo se utiliza el *test* de bondad de ajuste no paramétrico de Kolmogorov-Smirnov, cuya hipótesis nula sostiene que la muestra proviene de la distribución de referencia, que en este caso es la distribución normal. Para probar que los residuos no presentan correlación, se hace uso del *test* de Ljung-Box. La hipótesis nula de esta prueba indica que los datos no están correlacionados.

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 (*features*).

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 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 erróneamente.

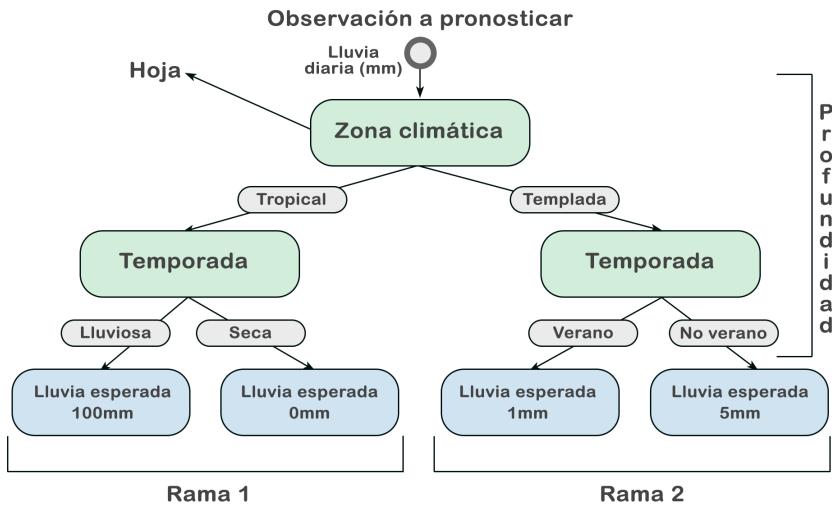


Figura 3: Ejemplo de árbol de decisión

Los métodos de ensamble buscan mejorar la robustez y precisión de las predicciones combinando los resultados de varios estimadores base, como los árboles de decisión. 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 ensamble (Bosques aleatorios, XGBoost, LightGBM, CatBoost, 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).

Se denomina *Boosting* a 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.

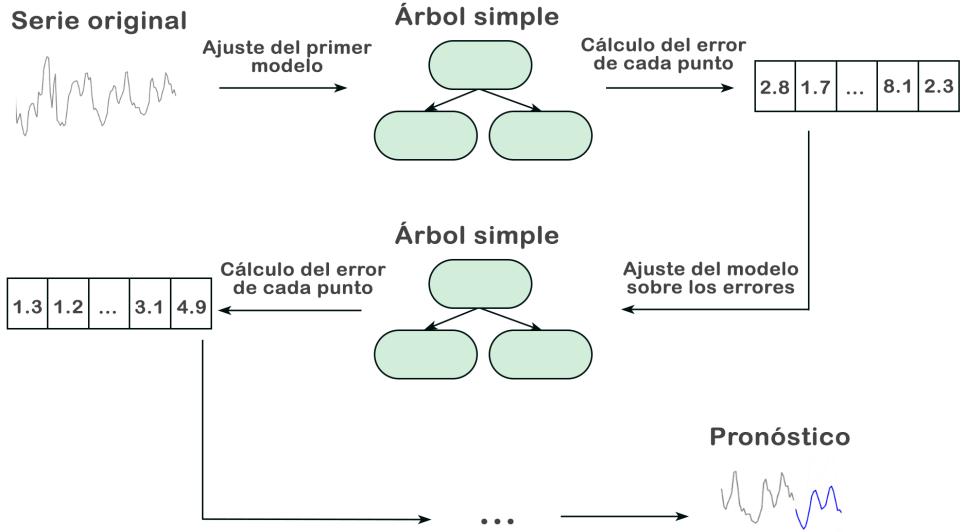


Figura 4: Proceso de ensamblado

La dirección de máximo crecimiento para una función está determinada por su gradiente, y por consiguiente, la dirección contraria es la dirección de máximo decrecimiento. El descenso del gradiente es un algoritmo iterativo en el que, con el objetivo de minimizar una función de pérdida, se calcula en cada paso la derivada de la función de costo con respecto a cada parámetro en su valor actual. Luego, se actualizan los valores de los parámetros desplazandolos una pequeña magnitud proporcional a η , llamada “tasa de aprendizaje”, de forma tal que la función de pérdida decrezca.

Sea $L(x)$ una función de pérdida y θ_0 el valor actual de un parámetro del modelo, la actualización de dicho parámetro se realiza de la siguiente manera:

$$\theta_1 = \theta_0 - \eta \cdot \nabla L(\theta_0) \quad (5)$$

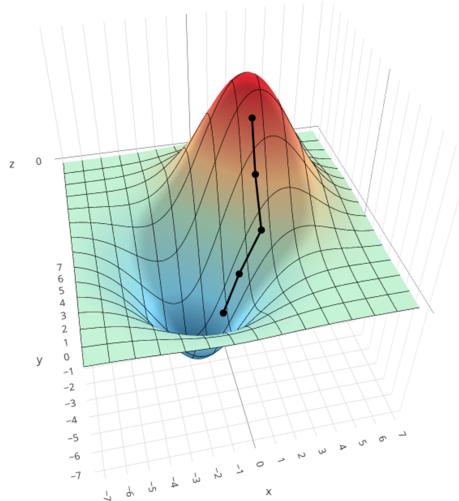


Figura 5: Ejemplo del descenso del gradiente en una función de pérdida

Este procedimiento iterativo se repite hasta lograr la convergencia o se llegue a un límite de iteraciones. Es por esto que la elección de η es sumamente importante para lograr el objetivo, ya que valores grandes podrían ocasionar divergencia, mientras que valores pequeños pueden llevar a que el algoritmo necesite demasiadas iteraciones para lograr la convergencia.

3.3.2 Diferencias entre XGBoost y LightGBM

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

XGBoost utiliza un método exacto en el que se calcula la ganancia de las particiones con cada característica, entendiendo por ganancia a la contribución relativa de una característica en el contexto de un árbol particular. Se elige la mejor partición y repite el proceso. Si bien este método es preciso, es lento computacionalmente. Por otro lado, 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. Que un punto tenga un gradiente bajo significaría que está “mejor pronosticado” que el resto, y por lo tanto no es necesario enfocarse tanto en mejorar dicho punto. Además, LightGBM utiliza un procedimiento que acelera el ajuste cuando se tienen muchas características correlacionadas entre las cuales elegir, denominado *Exclusive Feature Bundle* (EFB).

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 predicción 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 predicción.

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 predictions*: 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 predictions* 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.

$$IP_{z_{n+l};1-\alpha} = z_n(l) \pm Q_{1-\alpha}(e) \quad (6)$$

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

3.4 Modelos de aprendizaje profundo

El *deep learning* 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 la cantidad de pesos y neuronas que no formen parte de la capa de entrada dan el total de parámetros que tiene que ajustar el modelo.

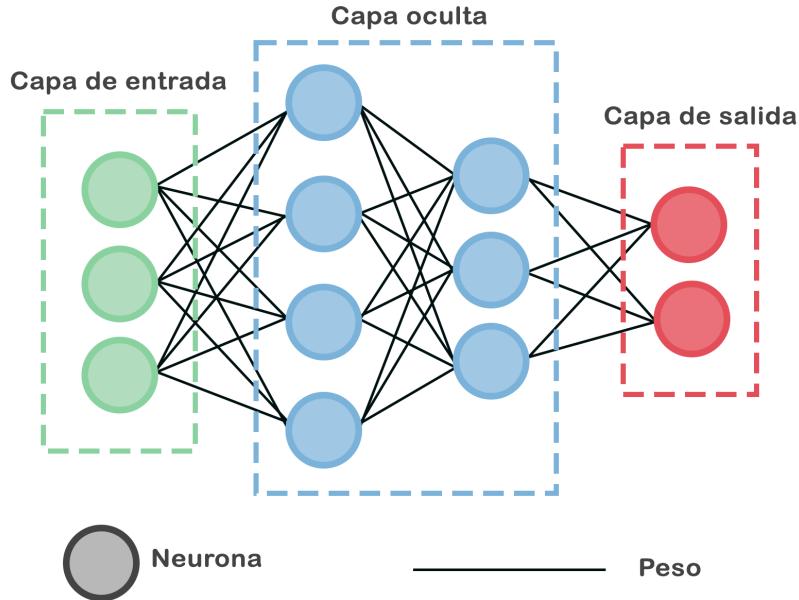


Figura 6: Ejemplo de red neuronal completamente conectada

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.

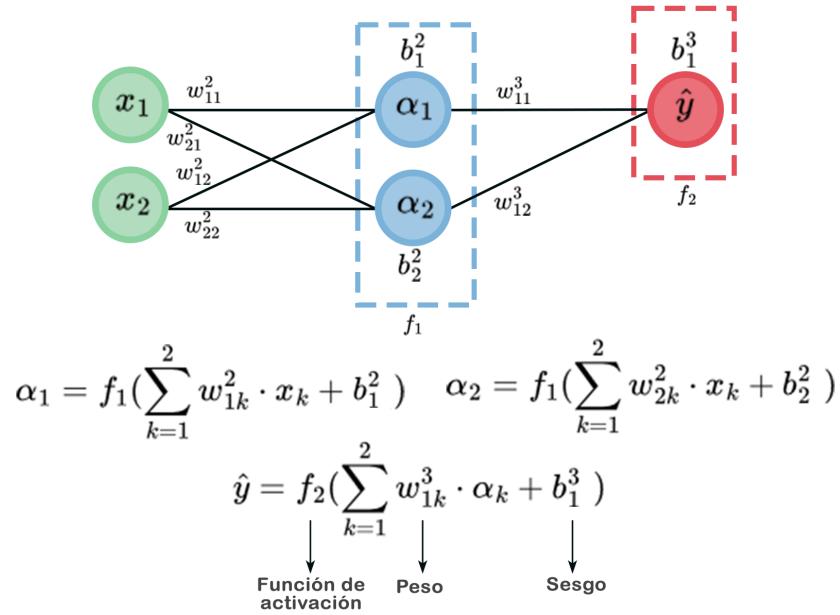


Figura 7: Obtención de una predicción en un red neuronal

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. Las *Feedforward Neural Networks* (FNN) son las redes neuronales más comunes y simples. Las redes neuronales recurrentes, del inglés *Recurrent Neural Networks* (RNN) utilizan bucles de retroalimentación que las hacen especialmente buenas en la predicción de datos secuenciales. Otro tipo de red neuronal son las *Convolutional Neural Networks* (CNN), las cuales son útiles para el reconocimiento de patrones en los datos.

3.4.2 Long Short Term Memory (LSTM)

Lo que caracteriza 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 FNN es independiente, en las redes neuronales recurrentes se relacionan entre ellas y se retroalimentan.

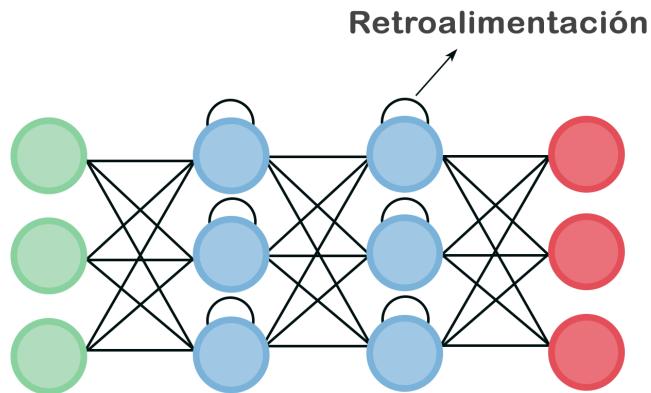


Figura 8: Ejemplo de RNN

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, provocando 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.

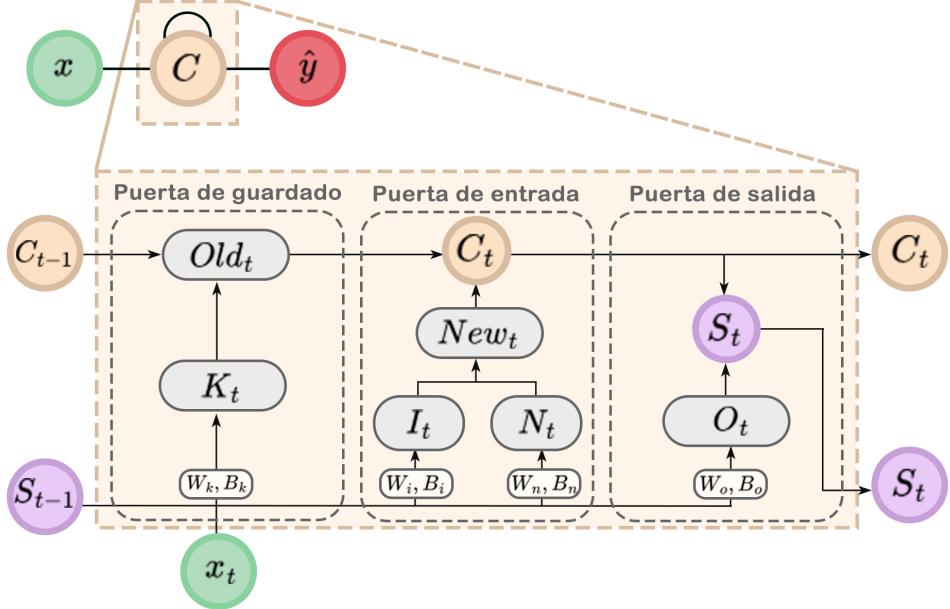


Figura 9: Estructura *Long Short Term Memory*

Puerta de guardado

La puerta de guardado se encarga de decidir que proporción de la información a largo plazo mantener en la neurona de memoria en cada iteración. Esta puerta recibe la entrada y el estado de la RNN, y las pasa como argumentos de una función sigmoide.

$$\tilde{\sigma}(x) = \frac{1}{1 + e^{-x}} \quad (7)$$

Sean S_{t-1} la información de la memoria a corto plazo actual de la red, x_t la entrada actual, y W_t y B_t los pesos y sesgos de la puerta de guardado respectivamente:

$$K_t = \tilde{\sigma}(W_k \times [S_{t-1}, x_t] + B_k) \quad (8)$$

$$Old_t = K_t \times C_{t-1} \quad (9)$$

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

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 a largo plazo debe ser descartada completamente.

Puerta de entrada

La puerta de entrada controla que información añadir a la neurona de memoria. Propone un nuevo valor para la información a largo plazo y decide que proporción de esta sumar al valor actual. Sea $\tanh(x)$ la función tangente hiperbólica y W_n y B_n pesos y sesgos respectivamente, el nuevo valor propuesto para la información a largo plazo es:

$$N_t = \tanh(W_n \times [S_{t-1}, x_t] + B_n) \quad (10)$$

Y la proporción de esta que se sumará al valor actual esta dada por I_t , cuyos pesos y sesgos son W_i y B_i .

$$I_t = \tilde{\sigma}(W_i \times [S_{t-1}, x_t] + B_i) \quad (11)$$

Por lo que el nuevo valor de la información a largo plazo de la red es:

$$C_t = Old_t + New_t \quad (12)$$

Donde $New_t = I_t \times N_t$

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:

$$O_t = \tilde{\sigma}(W_o \times [S_{t-1}, x_t] + B_o) \quad (13)$$

$$S_t = O_t \times \tanh(C_t) \quad (14)$$

Donde S_t es la nueva información a corto plazo en 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. Estos mecanismos capturan dependencias y relaciones en la secuencias de valores que se alimentan al modelo, logrando poner en contexto 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 importante de destacar en numerosas aplicaciones.

Sin embargo, estos valores siguen siendo n鷑eros 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 separen, $c_i < b_i < c_{i+1}$ para $i \in \{1, \dots, B - 1\}$. Las funciones de discretizaci髇 $q : \mathfrak{R} \rightarrow \{1, 2, \dots, B\}$, y de descuantizaci髇 $d : \{1, 2, \dots, B\} \rightarrow \mathfrak{R}$ se definen como:

$$q(x) = \begin{cases} 1 & \text{si } -\infty \leq x < b_1 \\ 2 & \text{si } b_1 \leq x < b_2 \\ \vdots & \\ B & \text{si } b_{B-1} \leq x < \infty \end{cases} \quad y \quad d(j) = c_j \quad (15)$$

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

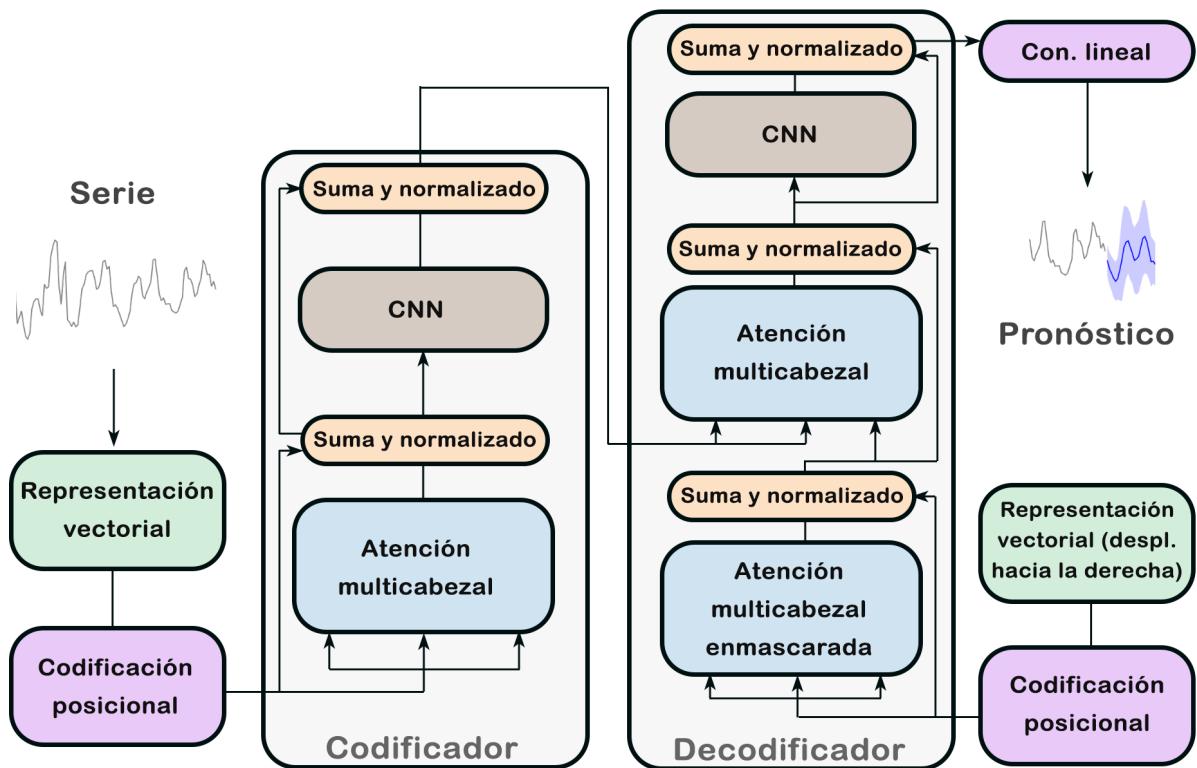


Figura 10: Diagrama de la estructura del modelo transformador de TimeGPT

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 (\vec{P}) 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 post-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 fila³ (\vec{S}). Valores altos indican que un token (de las filas) está siendo influenciado por el comportamiento de otro token (de las columnas).

	UN	GRAN	BLANCO	AVIÓN	EN	EL	AMPLIO	CIELO
UN	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
GRAN	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
BLANCO	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
AVIÓN	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
EN	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
EL	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
AMPLIO	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
CIELO	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
	W_Q							
UN	-4.9	-23.2	-12.0	-5.4	-84.9	-48.7	-13.1	-52.9
GRAN	-40.2	+4.7	-63.1	-12.3	-2.8	-40.6	-18.9	-20.9
BLANCO	-13.9	-74.7	-0.02	-16.5	-85.1	-51.7	-23.6	+4.3
AVIÓN	-91.6	+82.3	+83.9	13.1	-62.7	-36.3	-48.9	-24.0
EN	-17.3	-87.4	-21.0	-40.0	-18.4	-34.3	-72.8	-13.2
EL	-97.9	-95.1	-73.4	-60.4	-84.0	-13.1	-34.3	-46.1
AMPLIO	-54.1	-23.0	-84.7	-33.0	-67.7	-91.8	-10.6	-84.6
CIELO	-21.9	-46.8	-67.0	-84.2	-75.5	-53.2	+87.2	-11.4

Figura 11: Verificación del contexto

Una vez conocido que *tokens* son relevantes para otros *tokens*, es necesario saber de qué forma son afectados. Una matriz de valores (W_V) es post-multiplicada por cada vector de entrada resultando en los vectores de valor (\vec{V}), los cuales son multiplicados a cada columna. La suma por filas devuelven el vector $\Delta\vec{E}$ que se suma al vector de entrada original de cada token.

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 fila se comporte como una distribución de probabilidad.

	\vec{E}_1 $\downarrow \vec{W}_V$ \vec{V}_1	\vec{E}_2 $\downarrow \vec{W}_V$ \vec{V}_2	\vec{E}_3 $\downarrow \vec{W}_V$ \vec{V}_3	\vec{E}_4 $\downarrow \vec{W}_V$ \vec{V}_4	\vec{E}_5 $\downarrow \vec{W}_V$ \vec{V}_5	\vec{E}_6 $\downarrow \vec{W}_V$ \vec{V}_6	\vec{E}_7 $\downarrow \vec{W}_V$ \vec{V}_7	\vec{E}_8 $\downarrow \vec{W}_V$ \vec{V}_8
$\text{UN} \rightarrow \text{Softmax}(\vec{Q}_1 \vec{K}^\dagger)$								
$\text{GRAN} \rightarrow \text{Softmax}(\vec{Q}_2 \vec{K}^\dagger)$								
$\text{BLANCO} \rightarrow \text{Softmax}(\vec{Q}_3 \vec{K}^\dagger)$								
$\text{AVIÓN} \rightarrow \text{Softmax}(\vec{Q}_4 \vec{K}^\dagger)$	0.00 \vec{V}_1	0.18 \vec{V}_2	0.82 \vec{V}_3	0.00 \vec{V}_4	0.00 \vec{V}_5	0.00 \vec{V}_6	0.00 \vec{V}_7	0.00 \vec{V}_8
$\text{EN} \rightarrow \text{Softmax}(\vec{Q}_5 \vec{K}^\dagger)$								
$\text{EL} \rightarrow \text{Softmax}(\vec{Q}_6 \vec{K}^\dagger)$								
$\text{AMPLIO} \rightarrow \text{Softmax}(\vec{Q}_7 \vec{K}^\dagger)$								
$\text{CIELO} \rightarrow \text{Softmax}(\vec{Q}_8 \vec{K}^\dagger)$								

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

$$\Delta \vec{E}_j = \sum_i S_j \cdot \vec{V}_i \quad (16)$$

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

$$\vec{E} \leftarrow \vec{E} + \sum_h \Delta \vec{E}_h \quad (17)$$

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

$$\text{Atencion}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (18)$$

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

- Suma y normalizado: Este paso hace referencia a la ecuación 17, donde, en lugar de simplemente pasar los datos por las capas y modificar directamente el vector, las conexiones residuales se añaden sobre el vector de entrada en la salida de cada capa.

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.

- Red neuronal convolucional (CNN): Se utiliza una red neuronal para capturar patrones en los datos.

Decodificador

1. Representación vectorial (desplazado hacia la derecha): La entrada del decodificador son los *tokens* desplazados hacia la derecha, es decir, el primer valor del vector de entrada del codificador pasará a ser el segundo valor en el vector de entrada del decodificador, y así sucesivamente para el resto de elementos del vector. El primer valor de \vec{E} en el decodificador es un token con el valor **BOS**, comúnmente utilizado en modelos de aprendizaje profundo para indicar el inicio de una secuencia.
2. Codificación posicional
3. *Masked multi-head attention* (Atención multicabezal enmascarada): Las predicciones deben realizarse únicamente con los valores previos a cada token. Como consecuencia, antes de aplicar la transformación softmax, se debe reemplazar todos los valores debajo de la diagonal principal de la matriz QK por $-\infty$. Esto evita que los *tokens* sean influenciados por *tokens* posteriores.
4. *Multi-head attention*: 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 transformador preentrenado (de aquí las siglas GPT, *generative pre-trained transformer*) para el pronóstico de series de tiempo. Puede producir predicciones en diversas áreas y aplicaciones con gran precisión, sin necesidad de 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, sin embargo es de código cerrado.

Chronos es una familia de modelos transformadores preentrenados para series de tiempo basados en arquitecturas de modelos de lenguaje. Fue publicado por Amazon en marzo de 2024 y su código es de libre acceso.

Nixtla desarrolló para TimeGPT un modelo con arquitectura *transformer* que puede trabajar directamente con series de tiempo. Por otro lado, Chronos transforma los datos de las series para poder alimentarlos a los modelos de lenguaje existentes. Dado que al transformar los valores los datos se vuelven discretos, Chronos busca minimizar la entropía cruzada entre las distribuciones de las categorías reales contra las predichas.

La función de pérdida utilizada por Chronos está dada por:

$$\ell(\theta) = \sum_{l=1}^{h+1} \sum_{i=1}^{|\nu_{ts}|} 1_{z_{n+l+1}=i} \log p_\theta(z_{n+l+1} = i | z_{1:n+l}) \quad (19)$$

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*, cuyas primeras n observaciones se utilizan 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, 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 $\hat{e}_l = z_{n+l} - \hat{z}_n(l)$ el error de la l -ésima predicción, donde $\hat{z}_n(l)$ representa el pronóstico l pasos hacia adelante, el porcentaje del error absoluto medio (*Mean Absolute Percentage Error*, MAPE) para pronósticos h pasos hacia adelante se define como:

$$MAPE = \left(\frac{1}{h} \sum_{l=1}^h \left| \frac{\hat{e}_l}{z_{n+l}} \right| \right) \cdot 100\% \quad (20)$$

El problema del MAPE es que solo tiene 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*. Para un intervalo de predicción con una cobertura de $1 - \alpha$ el *Interval Score* es:

$$S = \frac{1}{h} \sum_{l=1}^h (W_l + O_l + U_l) \quad (21)$$

Donde:

$$W_l = IS_l - II_l \quad (22)$$

$$O_l = \begin{cases} \frac{2}{\alpha} (z_n(l) - z_{n+l}) & \text{si } z_n(l) > z_{n+l} \\ 0 & \text{en otro caso} \end{cases} \quad U_l = \begin{cases} \frac{2}{\alpha} (z_{n+l} - z_n(l)) & \text{si } z_n(l) < z_{n+l} \\ 0 & \text{en otro caso} \end{cases} \quad (23)$$

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.

Aquel modelo que minimice el MAPE y el *Interval Score* será considerado el mejor.

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 considera que un modelo presenta sobreajuste cuando se ajusta en exceso a los datos de entrenamiento, comprometiendo su capacidad de generalización frente a datos no observados. 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. Se ajustan las distintas configuraciones de parámetros sobre el resto de los datos de entrenamiento, y se prueban las métricas de evaluación sobre el conjunto reservado para validar. Luego, se ajusta el modelo con la mejor combinación de parámetros utilizando los datos de entrenamiento y de validación. Debido a la ordinalidad de los datos, las observaciones del conjunto de validación deben ser posteriores a las del conjunto de entrenamiento.

$$\text{Conjunto de entrenamiento total : } \left\{ \underbrace{z_1, \dots, z_c}_{\text{Entrenamiento}}, \underbrace{z_{c+1}, \dots, z_n}_{\text{Validación}} \right\} \quad (24)$$

En series que presentan estacionalidad, se prioriza que el conjunto de validación cubra al menos un ciclo. Esto tiene el objetivo de poder evaluar el ajuste del modelo en todo el ciclo estacional.

Exclusivamente para los modelos de la familia ARIMA, se utiliza el método de Box-Jenkins. El método consiste en comparar aquellos modelos que cumplan los supuestos, y elegir como mejor combinación de parámetros aquella que minimize el AIC (*Akaike Information Criterion*), medida de ajuste que penaliza por la cantidad de parámetros.

4. Aplicación

La aplicación empírica de esta tesina tiene como objetivo implementar, ajustar y comparar los modelos presentados en la sección 3, empleando un conjunto de series temporales seleccionadas. Para este fin, se utilizó el lenguaje de programación Python, junto con librerías de código abierto ampliamente reconocidas.

Se trabajó con series temporales reales, obtenidas de fuentes públicas y confiables. Cada serie fue analizada desde tres enfoques metodológicos: modelos estadísticos tradicionales, algoritmos de aprendizaje automático y modelos de aprendizaje profundo. En cada caso se exploraron distintas configuraciones de parámetros, explicando su significado y función en el ajuste.

- Los modelos estadísticos clásicos (ARIMA y SARIMA) se ajustaron mediante la librería `pmdarima`. La selección de parámetros se efectuó tanto de manera manual como automática, empleando 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 generaron pronósticos puntuales y probabilísticos, con una probabilidad de cobertura del 80%. Su desempeño se evaluó apartir de dos métricas: el *Mean Absolute Percentage Error* (MAPE) y el *Interval Score*. La elección de la mejor combinación de hiperparámetros se realizó en función del MAPE, con excepción de los modelos ARIMA. Los resultados fueron sintetizados en tablas comparativas y visualizaciones gráficas, acompañadas de un análisis crítico.

4.1 Series analizadas

Con el propósito de evaluar el desempeño de los distintos modelos bajo condiciones diversas, se seleccionaron tres series temporales con características heterogéneas:

- Número mensual de atenciones en guardia por patologías respiratorias en un hospital de la ciudad de Rosario
- Número mensual de trabajadores asalariados registrados en el sector educativo privado de Argentina
- Temperatura por hora en la ciudad de Rosario.

Serie 1. Atenciones en guardia por patologías respiratorias – Hospital de Niños Víctor J. Vilela (HNVV)

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

de Estadística de la Municipalidad de Rosario⁴. La serie presenta una marcada estacionalidad, con picos en los meses de invierno y una disminución significativa en el año 2020, atribuida a las medidas sanitarias implementadas en el contexto de la pandemia de COVID-19. Este patrón estacional se aprecia con mayor claridad la Figura 14.

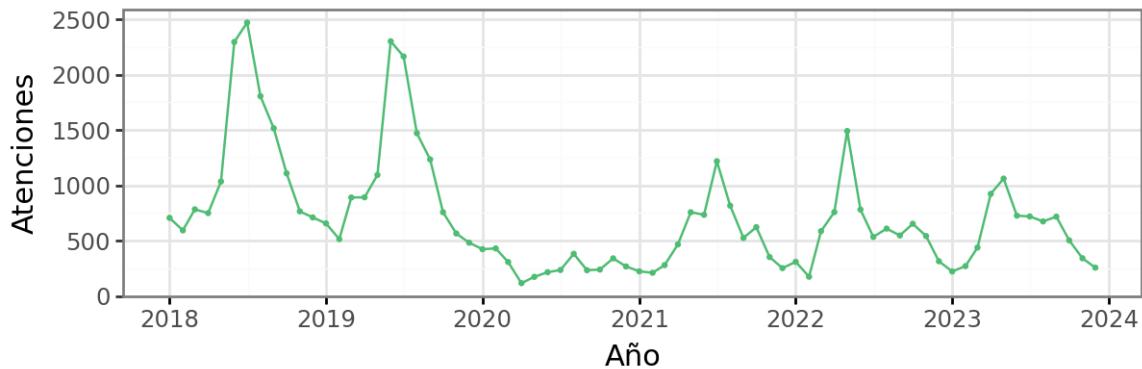


Figura 13: Atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario. Período 2018/01-2023/12.

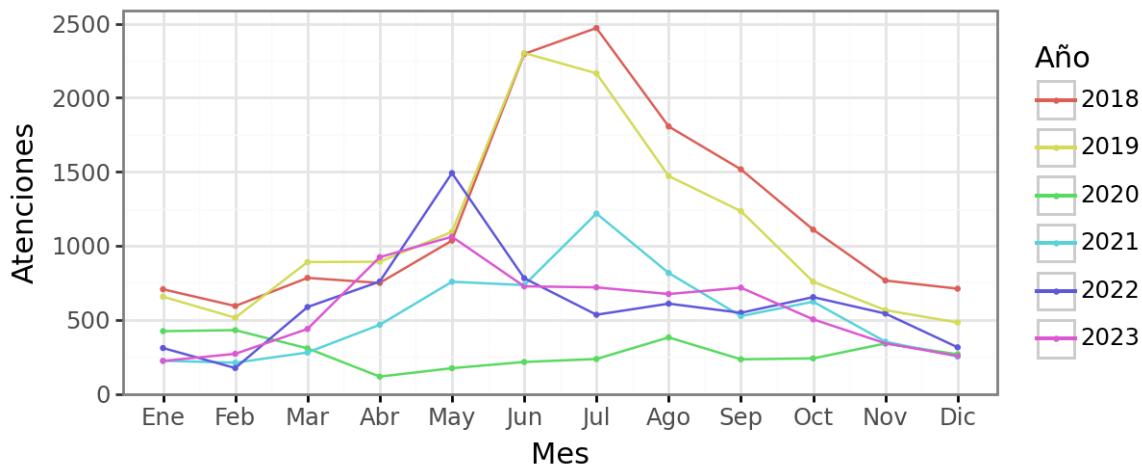


Figura 14: Atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario por año.

El diagrama de cajas (Figura 15) muestra grandes diferencias en la variabilidad de atenciones por año y una tendencia decreciente en la media. Estos patrones indican que la serie no es estacionaria, y una transformación es necesaria para lograr estacionariedad en varianza. La función `boxcox` de la librería `stats` propone una valor de λ igual a cero para la transformación de Box-Cox (ecuación 1). En consecuencia, se trabajó con la serie del logaritmo del número de atenciones en guardia por enfermedades respiratorias, presentada en la Figura 16, y los pronósticos se transformaron posteriormente a la escala original.

⁴Dirección General de Estadística. (s.f.) *Situación epidemiológica de problemas de salud priorizados*. Municipalidad de Rosario.

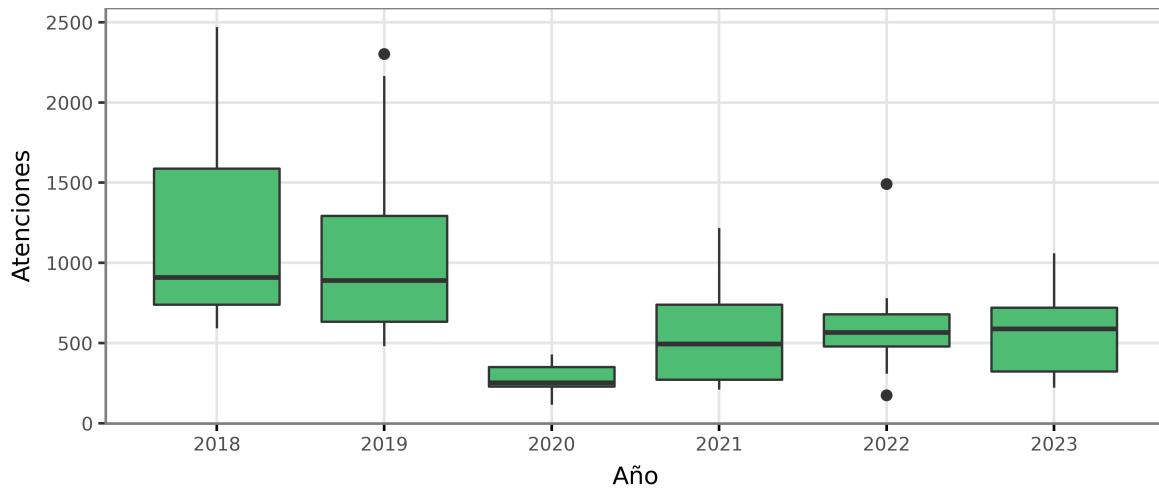


Figura 15: Distribución por año del número de atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

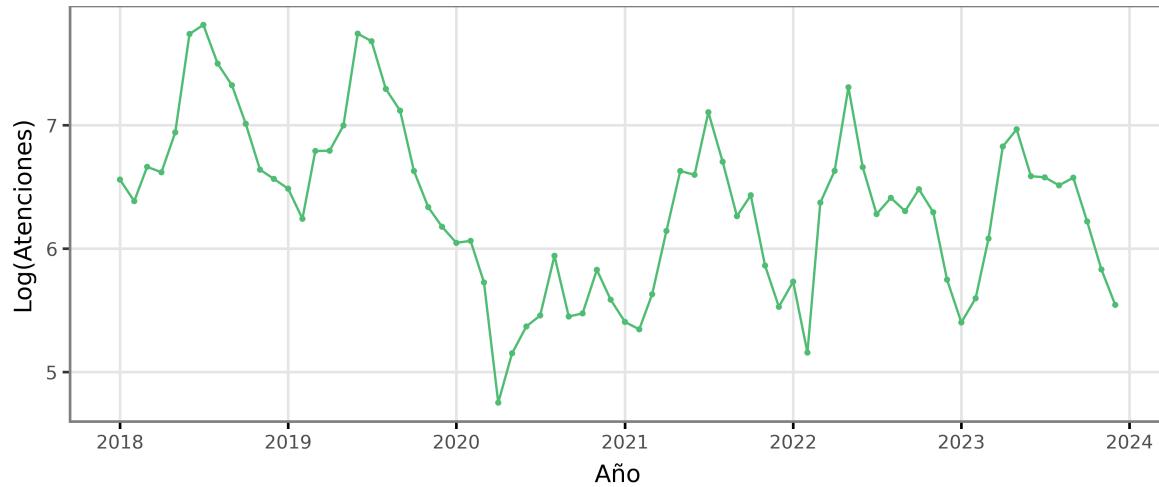


Figura 16: Logaritmo del número de atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

Serie 2. Trabajadores asalariados del sector educativo privado

La segunda serie corresponde al número mensual de trabajadores asalariados en el área de la enseñanza privada en Argentina. La serie presenta una tendencia creciente a lo largo del tiempo, con descensos pronunciados en diciembre y enero, los cuales se visualizan en la Figura 18. Asimismo, se observan impactos atribuibles a la pandemia de COVID-19. Los datos provienen del informe “Situación y evolución del Trabajo Registrado” de la Secretaría de Trabajo, Empleo y Seguridad Social⁵.

⁵Secretaría de Trabajo, Empleo y Seguridad Social. (2025). *Situación y evolución del Trabajo Registrado*. <https://www.argentina.gob.ar/trabajo/estadisticas/situacion-y-evolucion-del-trabajo-registrado>

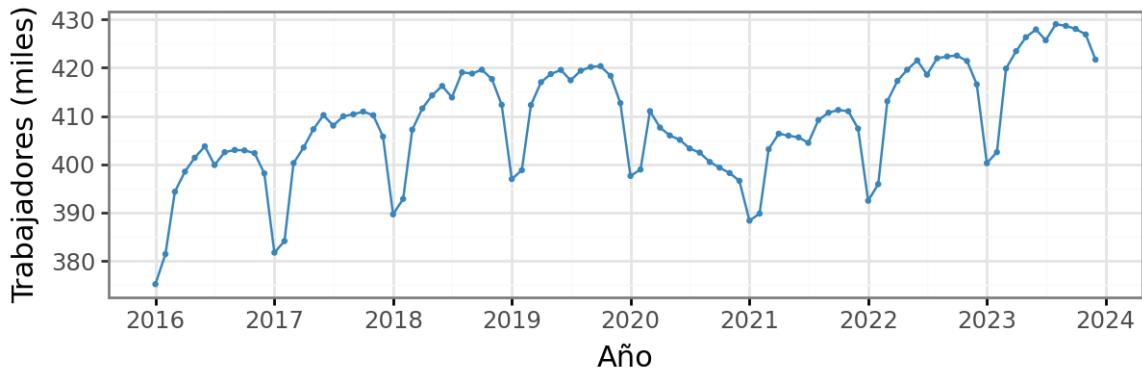


Figura 17: Trabajadores asalariados en el rubro de la enseñanza privada en Argentina. Período 2016/01-2023/12.

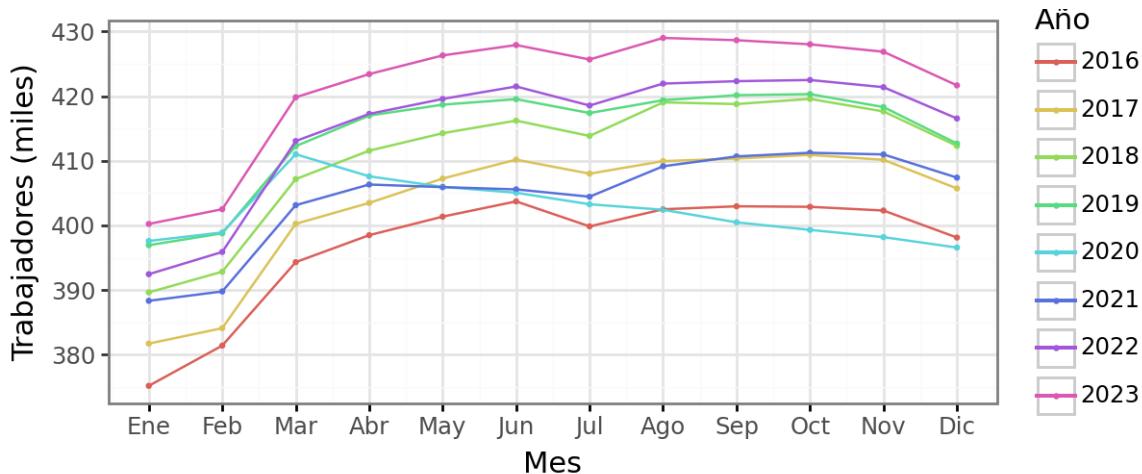


Figura 18: Número de trabajadores asalariados en el rubro de la enseñanza privada en Argentina por año.

Serie 3. Temperatura por hora en la ciudad de Rosario

Por último, se analizaron las temperaturas por hora en Rosario correspondientes a los primeros días de marzo de 2025, considerando su relación con la humedad relativa y la presión atmosférica estándar. El patrón estacional diario se aprecia claramente en la Figura 20: la temperatura se mantiene relativamente estable entre la noche y la mañana, y aumenta de forma pronunciada hacia la tarde. Los datos fueron relevados a partir de la página del Servicio Meteorológico Nacional⁶.

⁶Servicio Meteorológico Nacional. (s.f.). *Datos horarios*. <https://www.smn.gob.ar/descarga-de-datos>

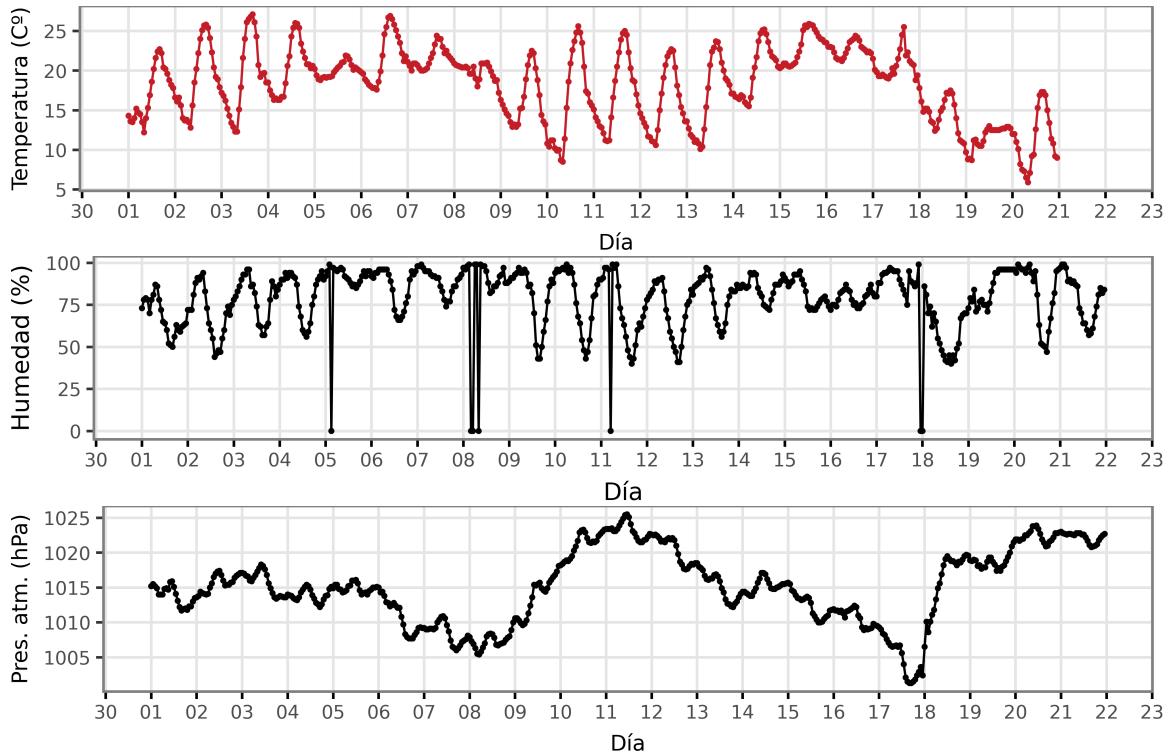


Figura 19: Temperatura (C°), humedad (%) y presión atmosférica (hPa) por hora en la ciudad de Rosario. Período 2025/03/01-2025/03/20.

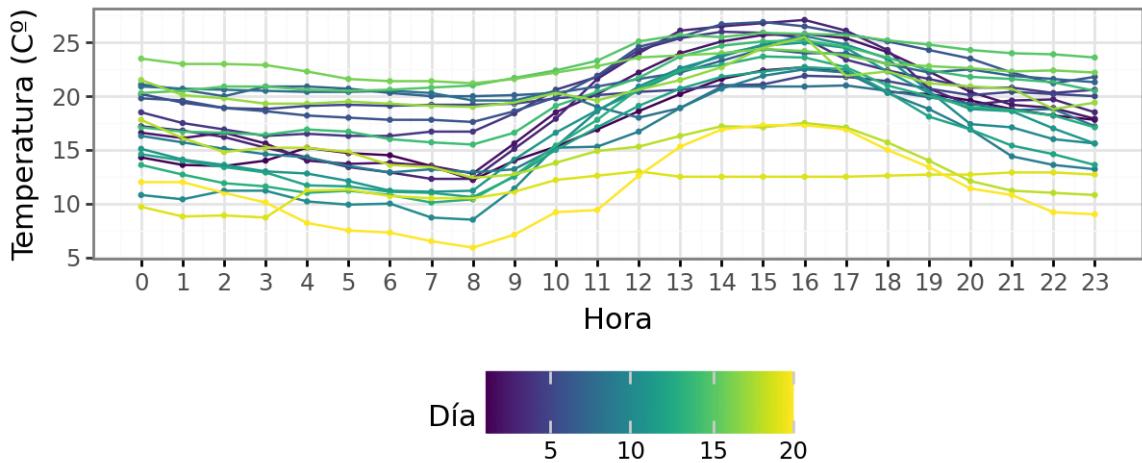


Figura 20: Temperatura por hora y día en la ciudad de Rosario.

Es importante señalar que, en este análisis, tanto la humedad como la presión atmosférica se suponen conocidas en las horas a pronosticar. Si bien esta condición no refleja una situación realista, en algunos contextos ciertas variables exógenas pueden disponerse con antelación. La inclusión de esta serie tuvo como finalidad evaluar la capacidad de los modelos para explotar información adicional. Cabe destacar que algunos de los modelos analizados permiten incorporar variables exógenas sin requerir el conocimiento de sus valores futuros. No obstante, para asegurar la comparabilidad de los resultados, en este trabajo se asumió que tanto la humedad como la presión atmosférica eran conocidas a futuro en todos los casos.

Otro aspecto a tener en cuenta es que no es adecuado realizar pronósticos de largo plazo para

series de temperaturas intradiarias, dado que la estacionalidad asociada al calendario introduce cambios estructurales en la dinámica del proceso.

4.2 Ajuste y evaluación de modelos

4.2.1 Modelización con ARIMA y SARIMAX

Serie 1. Atenciones en guardia por patologías respiratorias – Hospital de Niños Víctor J. Vilela (HNVV)

El logaritmo del número de atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario (Figura 16) presenta una tendencia decreciente en la media y un patrón estacional marcado que produce picos estacionales en los meses de invierno. A su vez, el correlograma simple de la serie, presentado en la Figura 21, muestra un descenso gradual en los primeros rezagos, lo que indica una posible falta de estacionariedad, la cual se confirma con el *test* aumentado de raíz unitaria de Dickey-Fuller, cuyo *p-value* es de 0.3837. Se concluye que una diferenciación regular y otra estacional son necesarias para conseguir que la serie sea estacionaria (Figura 22).

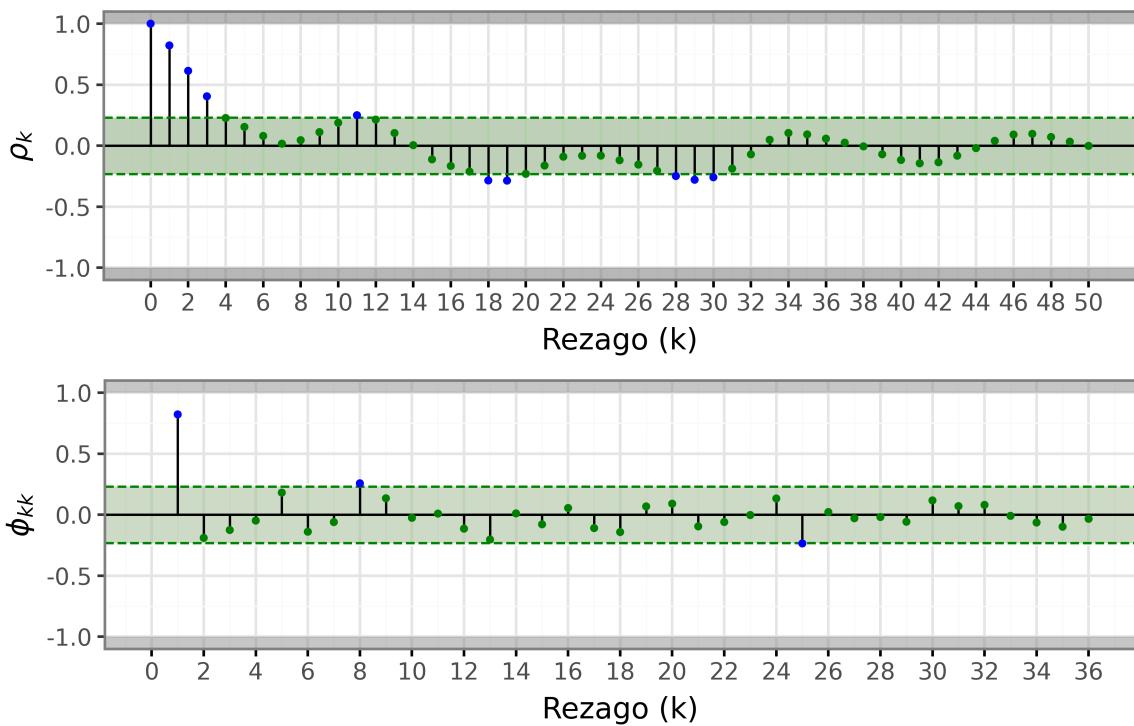


Figura 21: Correlograma simple y parcial para el logaritmo del número de atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

La Figura 23 exhibe el correlograma simple de la serie del logaritmo de atenciones en guardia, donde se destaca el rezago número 12, rezago que también es significativo en el correlograma parcial, indicando la presencia de una componente MA de orden 1 en la parte estacional. Por esta razón, se propone el modelo $SARIMA(0, 1, 0)(0, 1, 1)_{12}$ para pronosticar el número de atenciones en guardia por patologías respiratorias en Hospital de Niños Víctor J. Vilela (HNVV).

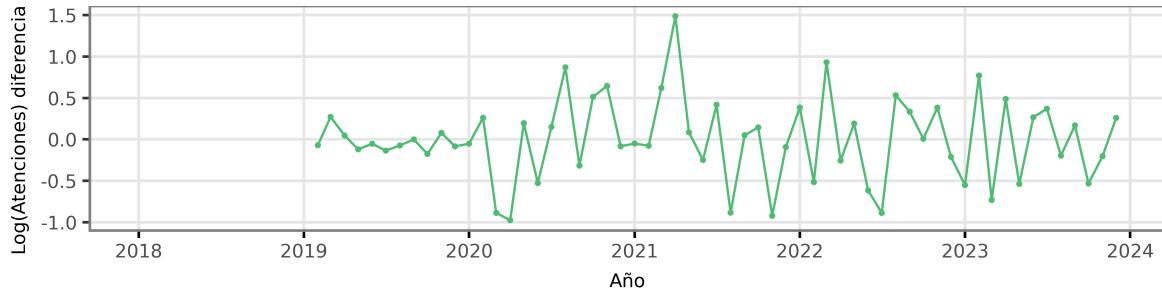


Figura 22: Logaritmo del número de atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario con una diferenciación regular y otra estacional.

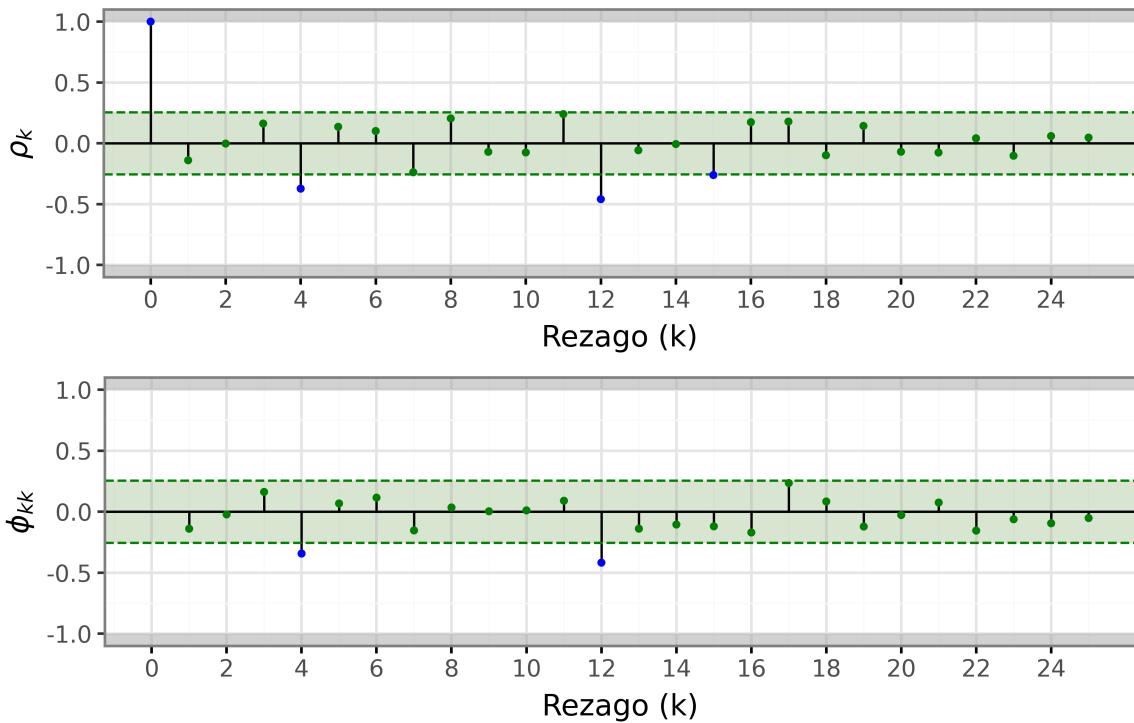


Figura 23: Correlograma simple y parcial del logaritmo del número de atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario con una diferenciación regular y otra estacional.

La Figura 24 muestra la comprobación de supuestos del modelo. En el gráfico superior izquierdo se presenta la distribución de los residuos estandarizados, acompañada con el *test* de Kolmogorov-Smirnov, que con un *p-value* de 0.0011 se concluye que no existe suficiente evidencia muestral para aceptar la normalidad, sin embargo esto puede deberse a ciertos valores atípicos. El gráfico superior derecho muestra los residuos estandarizados a través del tiempo, con el objetivo de comprobar la ausencia de patrones sistemáticos, esto se cumple para el modelo elegido y se destacan únicamente 2 *outliers*. En los gráficos inferiores, de izquierda a derecha, se visualizan las funciones de autocorrelación y autocorrelación parcial de los residuos estandarizados, junto al *test* de Ljung-Box. El valor que se muestra en el gráfico es el menor *p-value* registrado, producto de haber realizado el *test* para los 30 primeros rezagos. Con un *p-value* de 0.2562 se concluye que los residuos están incorrelacionados.

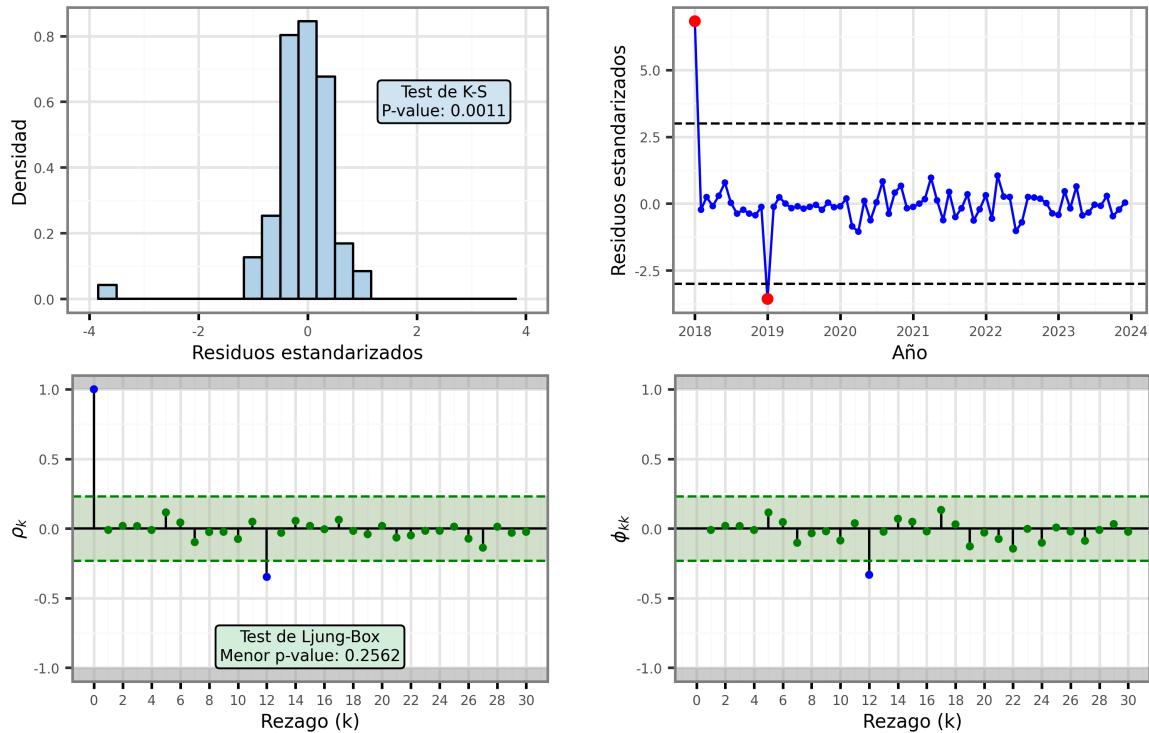


Figura 24: Comprobación de supuestos del modelo $SARIMA(0, 1, 0)(0, 1, 1)_{12}$ para el número de atenciones en guardia.

El modelo $SARIMA(0, 1, 0)(0, 1, 1)_{12}$ pronostica correctamente las temporadas bajas. Por otro lado, en los meses de invierno los pronósticos puntuales subestiman por un amplio margen los valores reales, aún así, estos son incluidos en los intervalos de predicción del 80% (Figura 25).

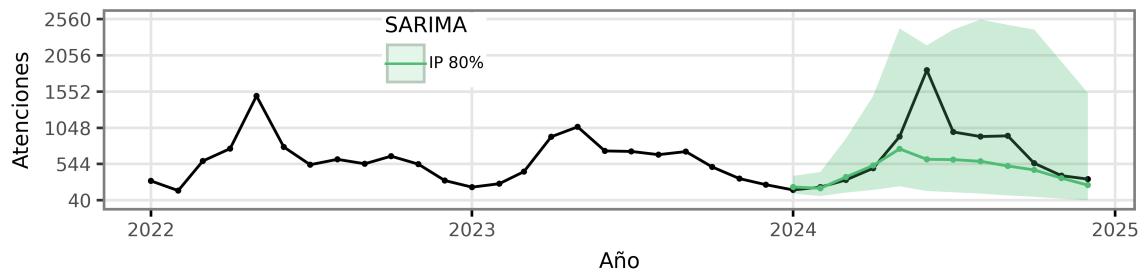


Figura 25: Pronósticos con el modelo $SARIMA(0, 1, 0)(0, 1, 1)_{12}$ sobre el número de atenciones en guardia por patologías respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

Tabla 1: Métricas de evaluación del modelo $SARIMA(0, 1, 0)(0, 1, 1)_{12}$ sobre la serie del número de atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

Modelo	Horizonte	MAPE	<i>Interval Score</i>
	3	0.1413	443.7176
$SARIMA(0, 1, 0)(0, 1, 1)_{12}$	6	0.2266	1143.1139
	12	0.2566	1636.6127

Serie 2. Trabajadores asalariados del sector educativo privado

La Figura 26 muestra la distribución por año del número de trabajadores asalariados en el rubro de la enseñanza privada en Argentina. Esta muestra que la variabilidad anual no cambia a través del tiempo, sin embargo, si se visualiza una tendencia positiva clara en la media de la serie. Esta falta de estacionariedad provoca un descenso gradual en las autocorrelaciones muestrales, graficadas en la Figura 27. Al mismo tiempo, el *test* de Kwiatkowski-Phillips-Schmidt-Shin, con un *p-value* igual a 0.0147, rechaza que la media de la serie sea estacionaria alrededor de una constante, lo cual confirma la necesidad de una diferenciación en la parte regular para lograr que la serie sea estacionaria.

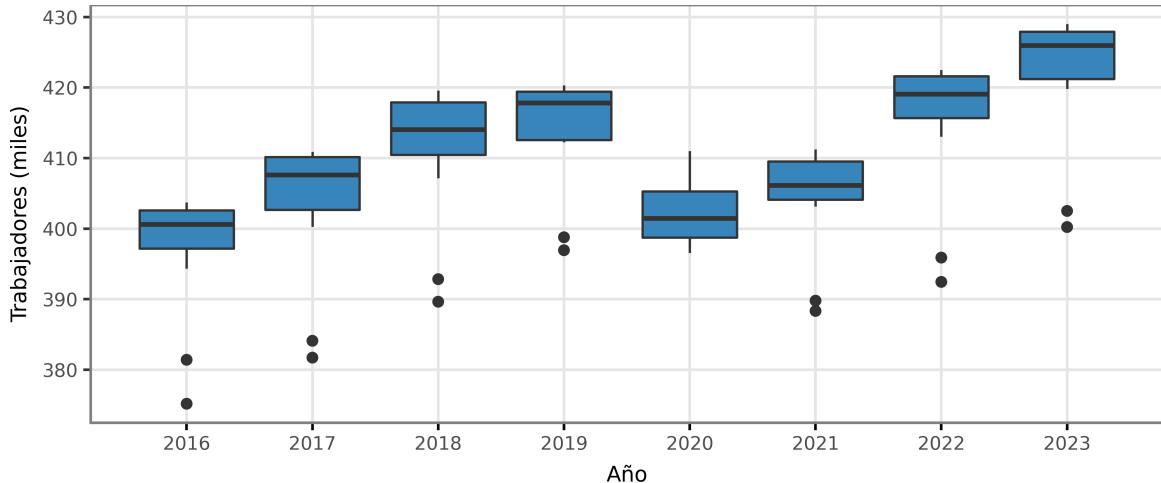


Figura 26: Distribución del número de trabajadores asalariados en el rubro de la enseñanza privada en Argentina por año. Período 2016/01-2023/12.

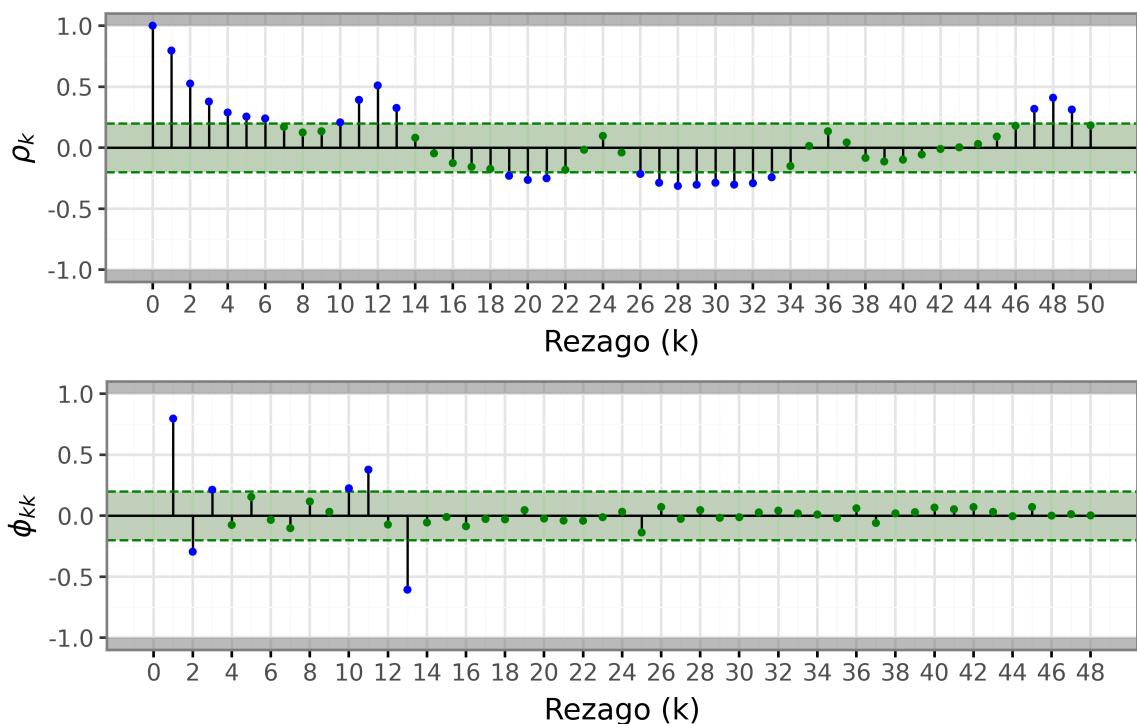


Figura 27: Correlograma simple y parcial para el número de trabajadores asalariados en el rubro de la enseñanza privada en Argentina.

La serie del número de trabajadores asalariados del sector educativo privado en Argentina con una diferenciación de orden uno no exhibe tendencia pero sí un comportamiento estacional marcado (Figura 28). El correlograma simple se destaca por tener valores significativos en los rezagos 1, 2, 12, 24, 36 y 48; mientras que en el correlograma parcial sobresalen los rezagos 1, 2, 11, 12 y 13 (Figura 29). Esta información lleva a proponer el modelo $SARIMA(1,1,0)(1,0,1)_{12}$.

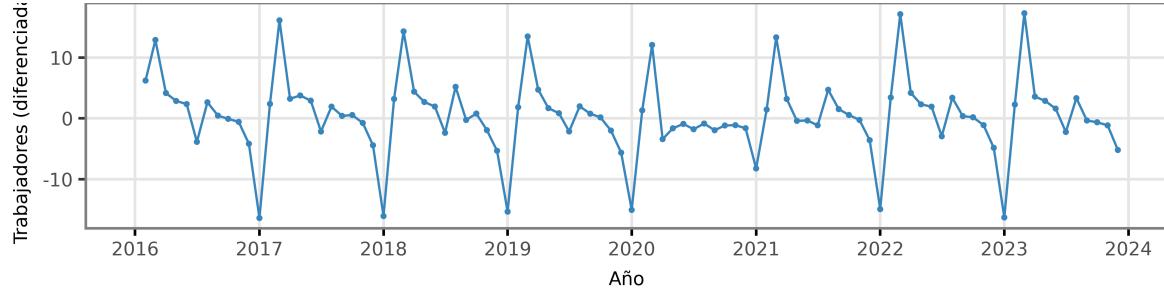


Figura 28: Número de trabajadores asalariados en el rubro de la enseñanza privada en Argentina con una diferenciación regular. Período 2016/01-2023/12.

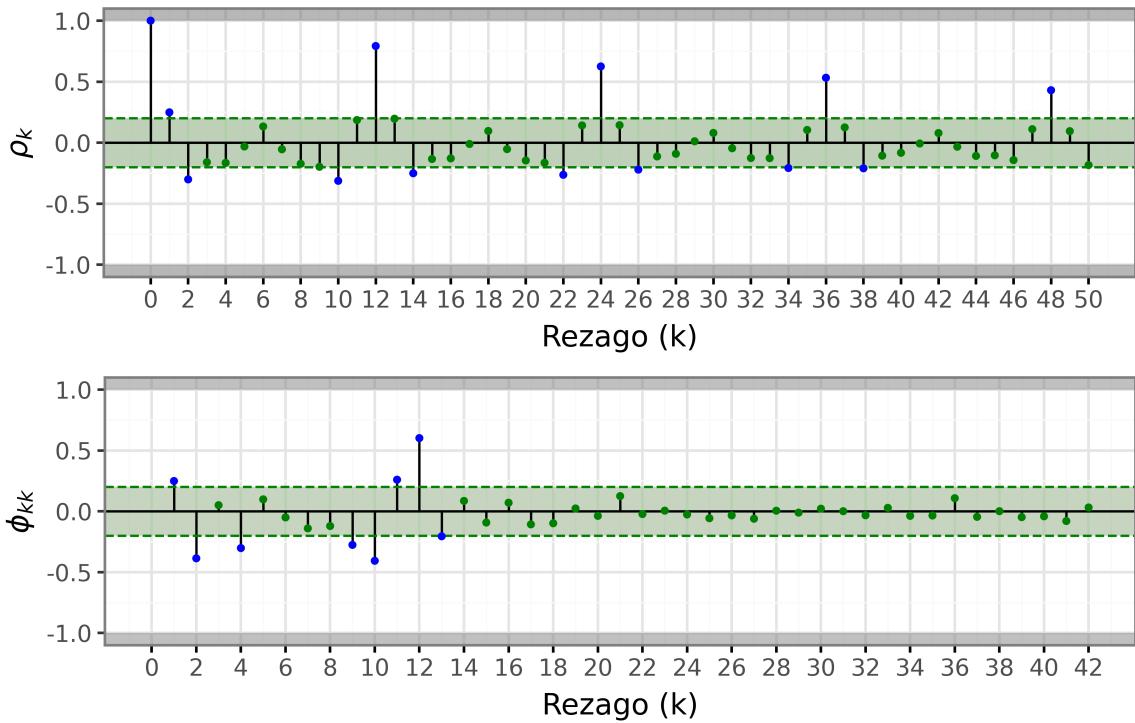


Figura 29: Correlograma simple y parcial del número de trabajadores asalariados en el rubro de la enseñanza privada en Argentina con una diferenciación regular.

A partir de la 10 del anexo 8.1, se puede apreciar que el modelo seleccionado goza de estacionariedad e invertibilidad, tanto en la parte regular como en la estacional, alcanza un AIC de 423.85 y todos sus parámetros son significativos.

Además, los residuos estandarizados del modelo cumplen los supuestos distribucionales, esto se comprueba a partir de la Figura 30, en la que se observa que los residuos no siguen ningún patrón particular ni están correlacionados. Si bien se rechaza la normalidad, esto puede deberse a un valor atípico.

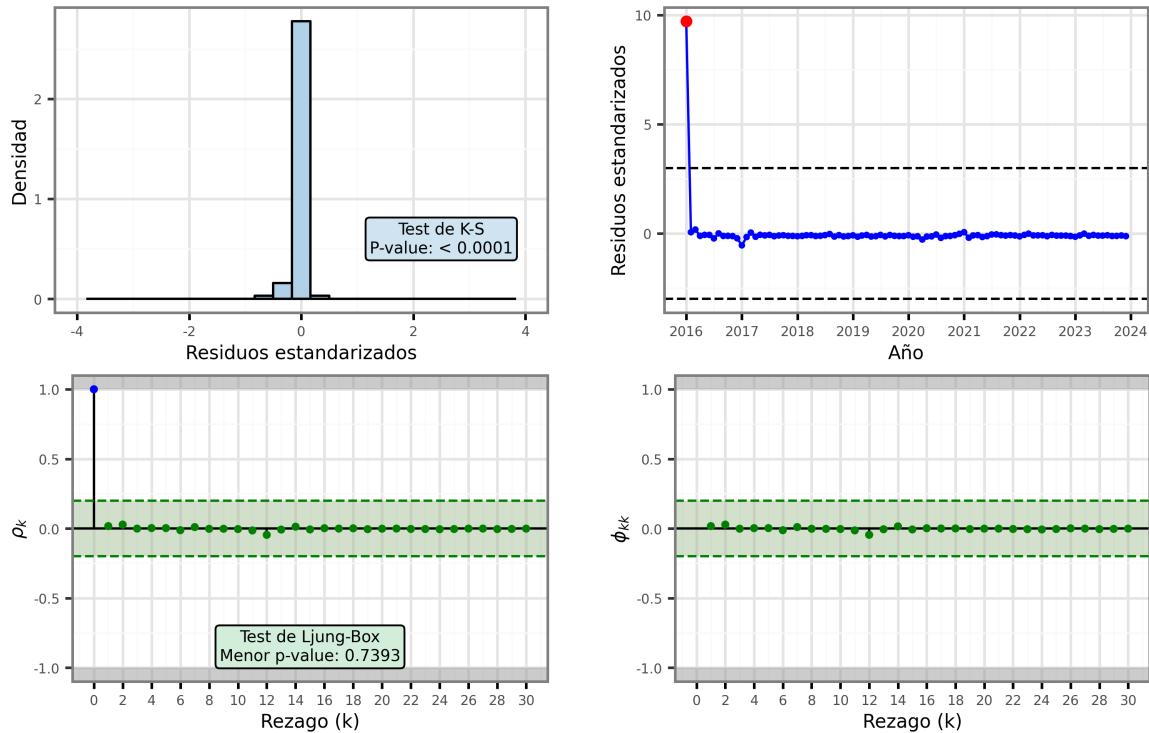


Figura 30: Comprobación de supuestos del modelo $SARIMA(1,1,0)(1,0,1)_{12}$ sobre los residuos estandarizados de la serie de trabajadores asalariados en el rubro de la enseñanza privada en Argentina.

Los pronósticos puntuales del modelo seleccionado sobre el número de trabajadores asalariados en el rubro de la enseñanza privada, presentes en la Figura 31, son cercanos a los valores reales de la serie. El modelo consigue incorporar correctamente el patrón de los datos, logrando un MAPE del 0.0085 en el pronóstico a largo plazo. Esto significa que la diferencia media absoluta entre los valores reales y los pronosticados es únicamente del 0.8513% (Tabla 2).

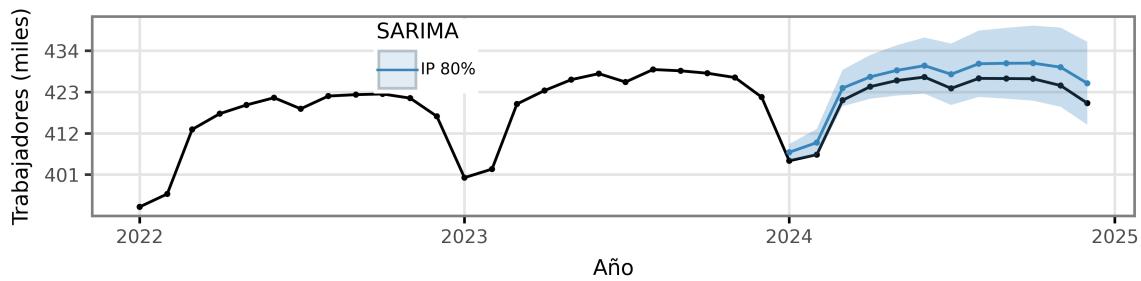


Figura 31: Pronósticos con el modelo $SARIMA(1,1,0)(1,0,1)_{12}$ sobre el número de trabajadores asalariados en el rubro de la enseñanza privada en Argentina.

Tabla 2: Métricas de evaluación del modelo $SARIMA(1,1,0)(1,0,1)_{12}$ sobre la serie del número de trabajadores asalariados en el rubro de la enseñanza privada en Argentina.

Modelo	Horizonte	MAPE	Interval Score
	3	0.0071	29.1066
$SARIMA(1,1,0)(1,0,1)_{12}$	6	0.0068	10.3768
	12	0.0085	14.8479

Serie 3. Temperatura por hora en la ciudad de Rosario

Debido a la complejidad que implica modelar series de alta frecuencia con variables exógenas con la metodología ARIMA, el modelo para la serie de temperaturas fue obtenido mediante selección automática usando la función `auto_arima` de la librería `pmdarima`. El modelo propuesto por este método es el $SARIMAX(1, 1, 1)(2, 0, 1)_{24}$, sin embargo, habiendo analizado los coeficientes presentes en la 11 del anexo 8.1, se concluye que el modelo no goza de estacionariedad en parte estacional. Como corrección, se propone el modelo $SARIMAX(1, 1, 0)(1, 0, 1)_{24}$, que sí goza de las propiedades mencionadas.

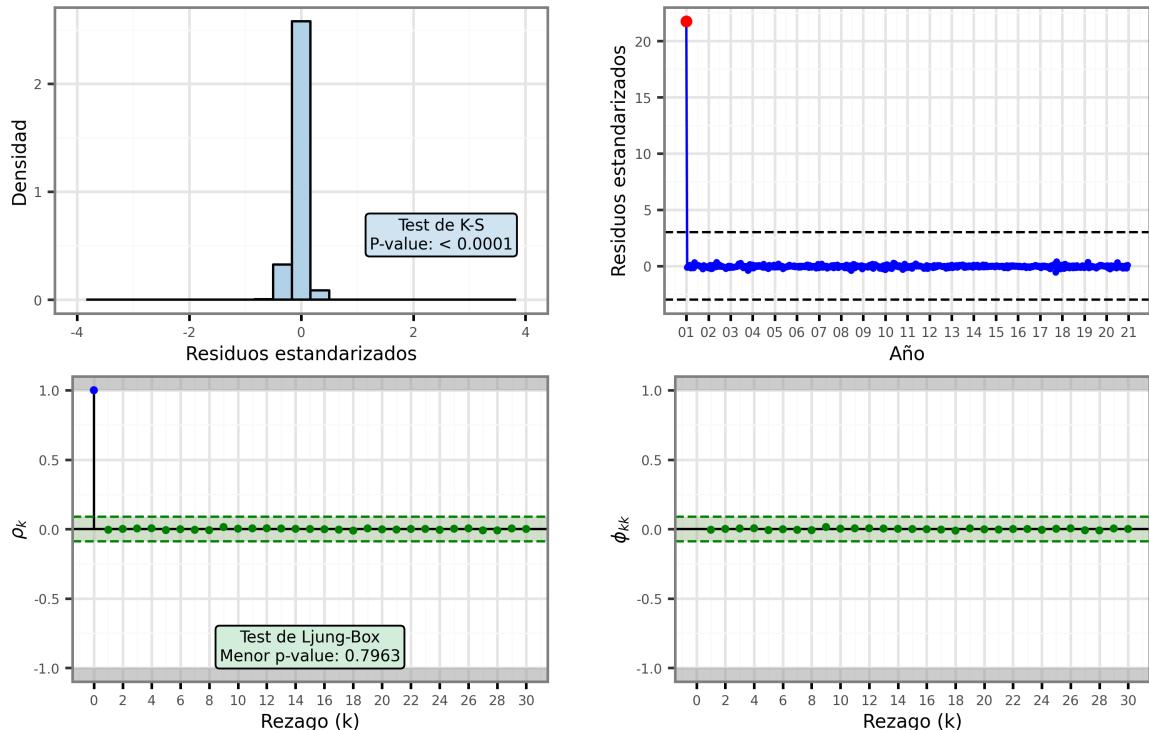


Figura 32: Comprobación de supuestos del modelo $SARIMAX(1, 1, 0)(1, 0, 1)_{24}$ para la serie de temperaturas en la ciudad de Rosario.

La Figura 32 muestra ausencia de patrones sistemáticos e incorrelación en los residuos estandarizados del modelo $SARIMAX(1, 1, 0)(1, 0, 1)_{24}$. Si bien la normalidad no se cumple, esto afecta únicamente a los pronósticos probabilísticos.

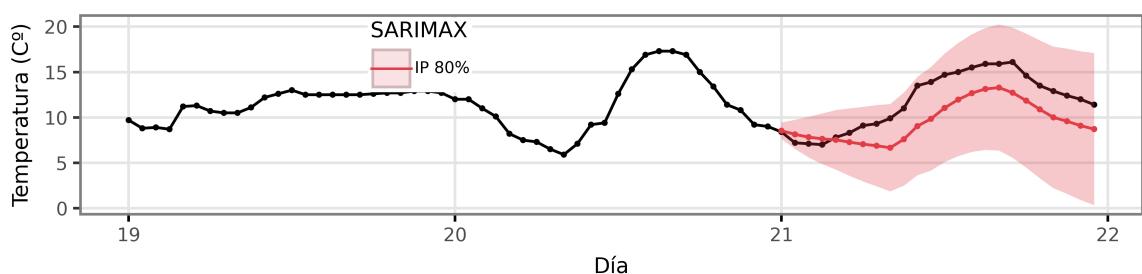


Figura 33: Pronósticos con el modelo $SARIMAX(1, 1, 0)(1, 0, 1)_{24}$ sobre la temperatura por hora en la ciudad de Rosario.

Tabla 3: Métricas de evaluación del modelo $SARIMAX(1, 1, 0)(1, 0, 1)_{24}$ sobre la serie de temperatura por hora en la ciudad de Rosario.

Modelo	Horizonte	MAPE	<i>Interval Score</i>
$SARIMA(1, 1, 0)(1, 1, 0)_{24}$	6	0.0829	4.8344
	12	0.1869	7.3556
	24	0.1974	10.915

La diferencia media absoluta entre los valores reales y los pronosticados es apenas del 8.2868% en el pronóstico del modelo $SARIMAX(1, 1, 0)(1, 0, 1)_{24}$ sobre las temperaturas en las primeras 6 horas. Sin embargo, esta diferencia aumenta en gran medida en los demás horizontes, al igual que la amplitud del intervalo de predicción, generando *Interval Scores* mayores a medida que se aumenta el horizonte.

4.2.2 Modelos de aprendizaje automático

Los modelos de pronóstico basados en aprendizaje automático no imponen supuestos estadísticos estrictos, lo que simplifica su implementación y facilita la automatización. No obstante, su desempeño depende en gran medida de la selección adecuada de *features* y de la configuración de hiperparámetros del modelo.

En este trabajo se implementaron los algoritmos XGBoost y LightGBM, ambos basados en técnicas de *gradient boosting* sobre árboles de decisión. Las características utilizadas en todas las series fueron:

- 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

XGBoost y LightGBM permiten parametrizar los modelos de varias formas. Con el objetivo de determinar la mejor configuración, se construyó una grilla de búsqueda de hiperparámetros, donde cada combinación se ajustó sobre el conjunto de entrenamiento y se evaluó sobre un conjunto de validación utilizando el MAPE como criterio principal. Los parámetros que se decidieron probar en este trabajo son los siguientes:

- Número de árboles que se contruyen 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 la Tabla 4 y la Tabla 5 se muestra que combinación de parámetros, de las 384 posibles, fue la que menor MAPE produjo sobre el conjunto de validación para cada serie, 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 4: Modelos XGBoost seleccionados y métricas de evaluación.

Serie	Hor.	A	P	H	η	C	MAPE	Interval Score
Atenciones	3	20	2	2	0.1	0.7	1.1992	909.3306
	6	20	2	2	0.001	0.7	1.6315	1895.0069
	12	20	5	2	0.2	1.0	0.5257	1553.1046
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

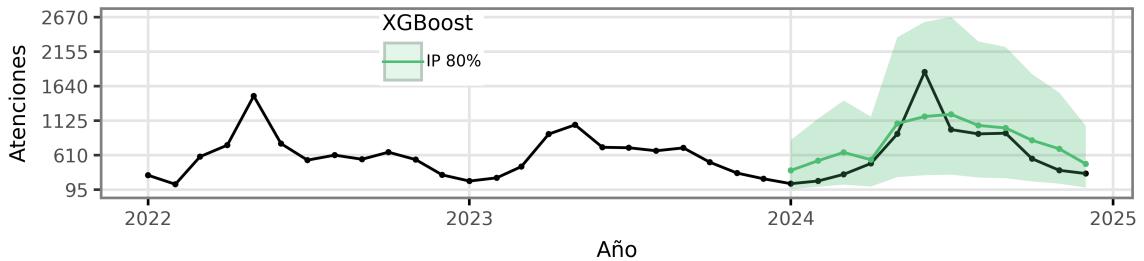


Figura 34: Pronóstico con XGBoost sobre el número de atenciones en guardia por patologías respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

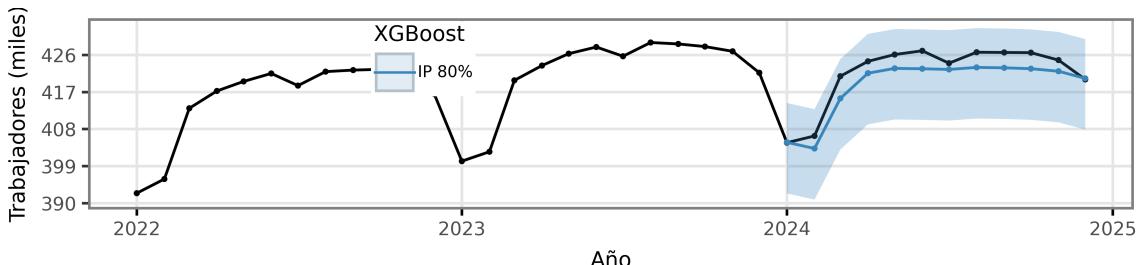


Figura 35: Pronóstico con XGBoost sobre el número de trabajadores asalariados (miles) en el rubro de la enseñanza privada en Argentina.

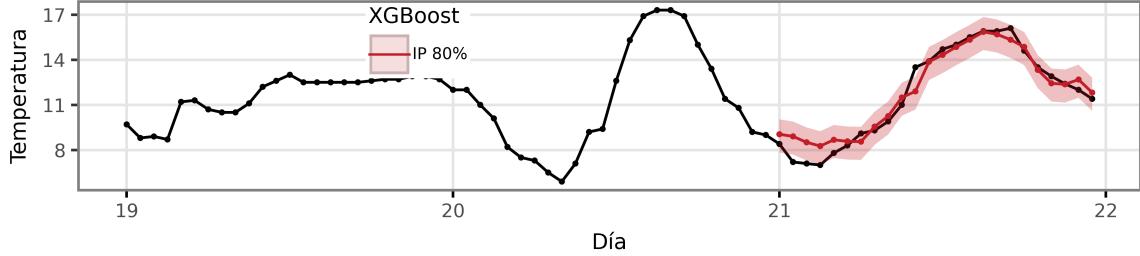


Figura 36: Pronóstico con XGBoost sobre las temperaturas por hora en la ciudad de Rosario.

Los pronósticos con XGBoost fueron satisfactorios sobre las tres series, logrando captar correctamente los patrones estacionales. Los intervalos de predicción del 80% conseguidos gracias a *Ensemble Batch Prediction Intervals* contienen en la mayoría de los casos a los valores reales. Es interesante notar la baja amplitud en los intervalos para la serie de temperaturas, lo que indica una mejora en la incertidumbre del modelo con respecto a los resultados vistos con el modelo SARIMAX.

Es interesante notar que todos los modelos elegidos fueron construidos con árboles sencillos, de una única división ($H = 2$). Esto puede deberse a la existencia de alguna característica que por sí sola explica gran parte de la aleatoriedad de las variables. Otro aspecto a notar es que los modelos que pronosticaron la serie de atenciones utilizaron pocos árboles en las estimaciones ($A = 20$), mientras que para la serie de trabajadores se usaron el máximo establecido ($A = 150$). Esto se explica con la variabilidad de las series, dado que el número de atenciones en guardia no tiene patrones muy marcados, modelos más complejos podrían llevar a sobreajustes, mientras que la poca aleatoriedad en el comportamiento de la serie de trabajadores beneficia a los modelos más complejos.

Tabla 5: Modelos LightGBM seleccionados y métricas de evaluación.

Serie	Hor.	A	P	H	η	C	MAPE	Interval Score
Atenciones	3	50	2	4	0.1	0.7	1.0234	826.8506
	6	20	2	2	0.001	0.7	1.633	1893.8128
	12	100	2	4	0.2	1.0	0.4509	1287.0798
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

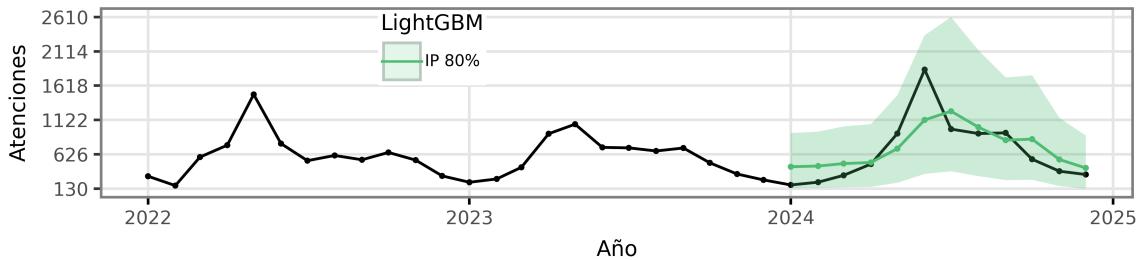


Figura 37: Pronóstico con LightGBM sobre el número de atenciones en guardia por patologías respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

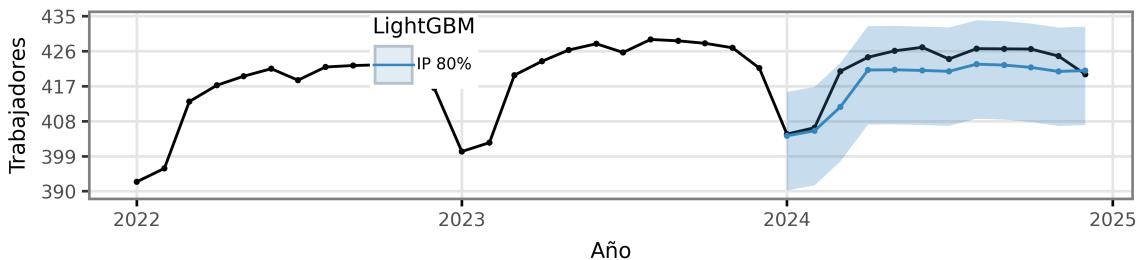


Figura 38: Pronóstico con LightGBM sobre el número de trabajadores asalariados (miles) en el rubro de la enseñanza privada en Argentina.

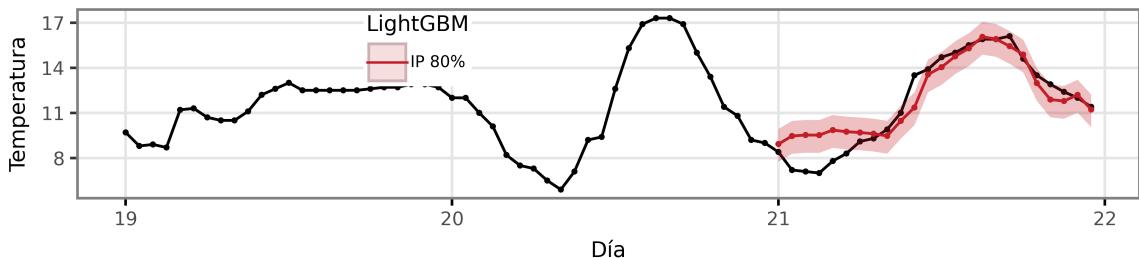


Figura 39: Pronóstico con LightGBM sobre las temperaturas por hora en la ciudad de Rosario.

En el caso de LightGBM, los resultados fueron ligeramente inferiores a los de XGBoost en las series de trabajadores y temperaturas. En esta última la diferencia es especialmente importante, ya que en el pronóstico de las temperaturas a 24 horas los intervalos de predicción en las primeras 6 horas no suelen cubrir a las observaciones, lo que podría indicar una subestimación en la variabilidad de los pronósticos.

Resulta interesante destacar en la Tabla 4 y Tabla 5 como XGBoost tendió a elegir estructuras más sencillas, con menos árboles y menor profundidad, mientras que LightGBM optó por estructuras más complejas. En general, ambos modelos privilegiaron una tasa de aprendizaje alta ($\eta = 0.2$) y una amplia proporción de características ($C = 1$), configuraciones que podrían aumentar el riesgo de sobreajuste, aunque este no se evidenció en los resultados obtenidos.

4.2.3 Redes neuronales (LSTM)

Los modelos basados en aprendizaje profundo (*deep learning*) tienen la capacidad de extraer automáticamente las características más relevantes y de ajustar los parámetros internos del

modelo durante el entrenamiento. En el caso de las redes neuronales recurrentes, como las LSTM (*Long Short-Term Memory*), el principal desafío radica en definir la estructura o arquitectura del modelo.

Las redes neuronales tienen tendencia a sobreajustar, para evitar esto existen numerosas alternativas. En primer lugar se puede optar por detener 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* o detención temprana en español. Otra técnica es el *dropout*, que consiste en “apagar” aleatoriamente una proporción de neuronas en cada iteración, lo que equivale a entrenar un conjunto de redes más pequeñas y reduce la dependencia entre ellas.

Para cada serie se entrenaron modelos LSTM con 300 iteraciones de entrenamiento y una tasa de aprendizaje de 0.001. Se adoptó una paciencia para el *early stopping* de 10, lo cual quiere decir que si la función de pérdida no muestra mejoras significativas en 10 iteraciones seguidas se detiene el entrenamiento. Además, se probaron las siguientes características para la estructura del 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.
- Proporción de *dropout* en cada capa (D). Opciones: 0.1, 0.3.
- Función de activación (A). Opciones: ReLu, tangente hiperbólica (tanh).

Tabla 6: Modelos LSTM seleccionados y métricas de evaluación.

Serie	Hor.	N	R	D	A	MAPE	<i>Interval Score</i>
Atenciones	3	[24, 42]	1	0.3	tanh	0.699	886.5567
	6	[32]	24	0.3	tanh	0.3871	1203.9912
	12	[12, 24]	24	0.3	relu	0.8461	1469.5551
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

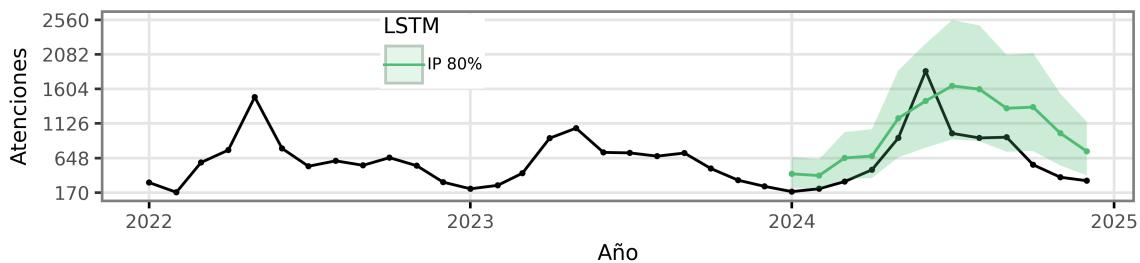


Figura 40: Pronóstico con LSTM sobre el número de atenciones en guardia por patologías respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

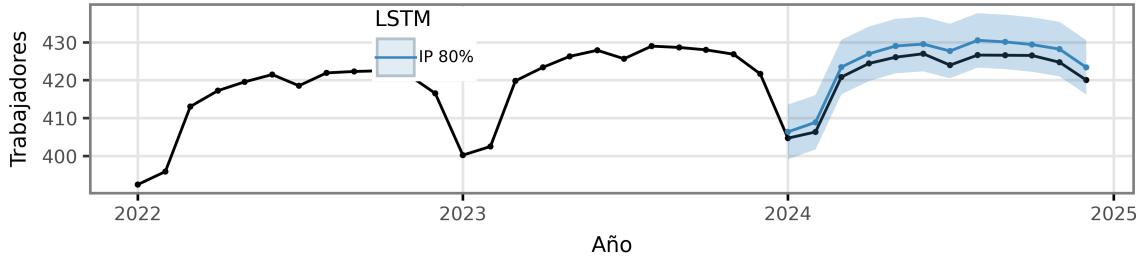


Figura 41: Pronóstico con LSTM sobre el número de trabajadores asalariados (miles) en el rubro de la enseñanza privada en Argentina.

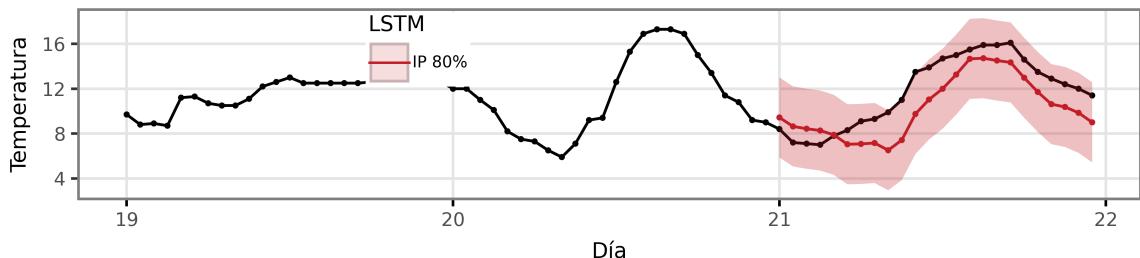


Figura 42: Pronóstico con LSTM sobre las temperaturas por hora en la ciudad de Rosario.

Al igual que en los métodos de aprendizaje automatizado, los intervalos de predicción se obtuvieron usando *conformal predictions*, pero se realizaron con funciones ya integradas en la librería `scalecast`.

Los pronósticos sobre todas las series fueron satisfactorios, sin embargo, se destaca la sobreestimación de las atenciones luego del pico invernal. La serie de trabajadores se destacó al usar una proporción de *dropout* baja ($D = 0.1$), al contrario de la serie de atenciones, llevando a pensar que una mayor proporción de *dropout* parece funcionar mejor en series con patrones menos marcados, similar a lo ocurrido con el número de estimadores en XGBoost.

4.2.4 Modelos fundacionales (TimeGPT y Chronos)

Los modelos fundacionales constituyen una nueva generación de herramientas para el pronóstico de series temporales, caracterizados por estar preentrenados en grandes volúmenes de datos. A diferencia de las redes neuronales tradicionales, no requieren la definición manual de características, la selección de hiperparámetros ni el ajuste de la arquitectura del modelo. Este tipo de predicción automática, que no necesita entrenamiento adicional sobre los datos específicos de la serie, se denomina *zero-shot forecasting*.

En caso de buscar una mayor adaptación del modelo a una serie específica, puede aplicarse la técnica conocida como *fine-tuning* o ajuste fino. Esta consiste en continuar el entrenamiento del modelo preentrenado mediante iteraciones adicionales que minimizan la función de pérdida sobre el nuevo conjunto de datos. Sin embargo, las pruebas realizadas en este trabajo no evidenciaron mejoras significativas en las métricas de pronóstico respecto de las configuraciones base, logrando en algunos casos resultados inferiores y aumentos considerables en los tiempos de procesamiento. Por esta razón, el ajuste fino no se emplea en los resultados finales, aunque se lo considera de interés para futuras investigaciones.

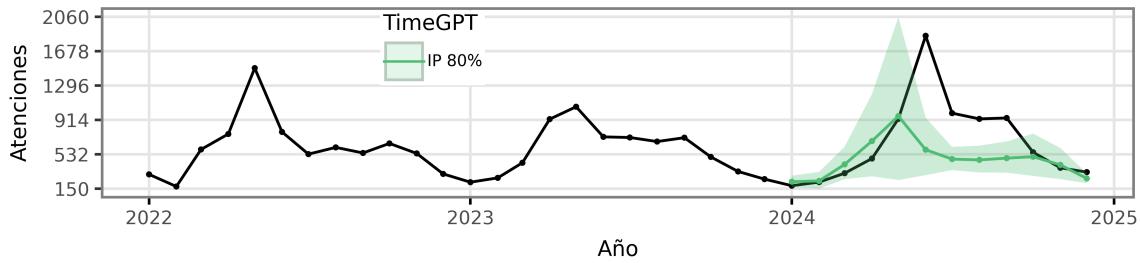


Figura 43: Pronóstico con TimeGPT sobre el número de atenciones en guardia por patologías respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

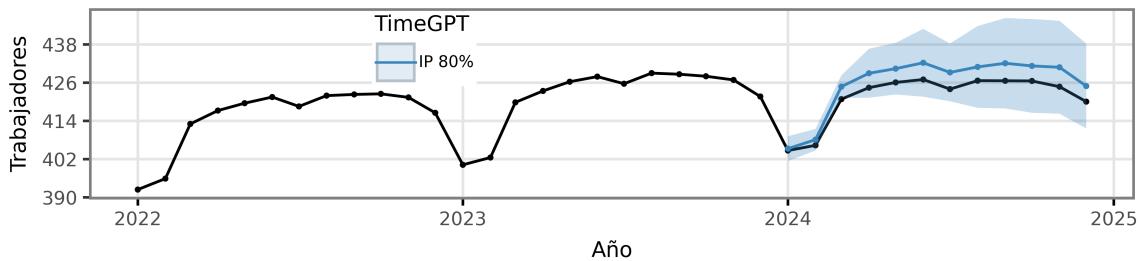


Figura 44: Pronóstico con TimeGPT sobre el número de trabajadores asalariados (miles) en el rubro de la enseñanza privada en Argentina.

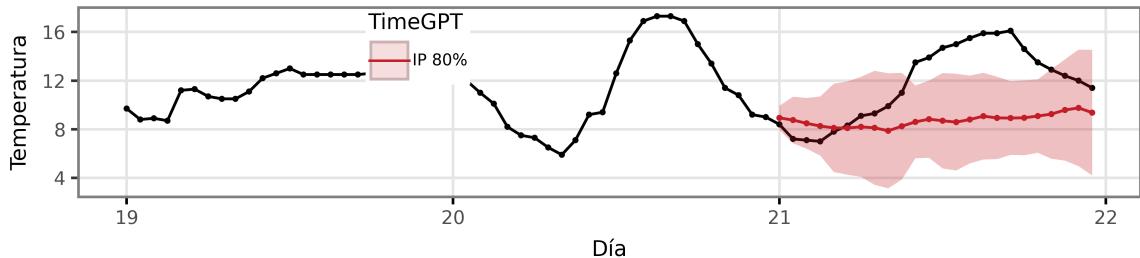


Figura 45: Pronóstico con TimeGPT sobre las temperaturas por hora en la ciudad de Rosario.

Tabla 7: Métricas de evaluación para los ajustes con TimeGPT.

Serie	Hor.	MAPE	<i>Interval Score</i>
Atenciones	3	1.7431	2774.7223
	6	2.1404	5520.5854
	12	0.2998	2037.713
Trabajadores	3	0.0015	2.6232
	6	0.0025	11.1076
	12	0.0101	19.6513
Temperatura	6	0.238	9.212
	12	0.1219	5.1091
	24	0.2625	18.0171

Los pronósticos obtenidos con TimeGPT mostraron un desempeño moderado en las series de

temperatura y atenciones, particularmente en los horizontes largos, donde el modelo no logró capturar adecuadamente los patrones estacionales. En cambio, en la serie de trabajadores el modelo reprodujo correctamente la dinámica temporal y alcanzó una alta precisión en los pronósticos. En la Figura 43 se observa un aumento significativo en la amplitud del intervalo de predicción para el pronóstico de la serie de atenciones, indicando una gran incertidumbre en el período de invierno.

Chronos es una familia de modelos fundacionales basada en arquitecturas de lenguaje adaptadas a series temporales. Dispone de múltiples variantes de distinto tamaño y complejidad para realizar pronósticos en modo *zero-shot*. En este trabajo se utilizó la versión **Chronos-bolt-small**, que cuenta con aproximadamente 48 millones de parámetros.

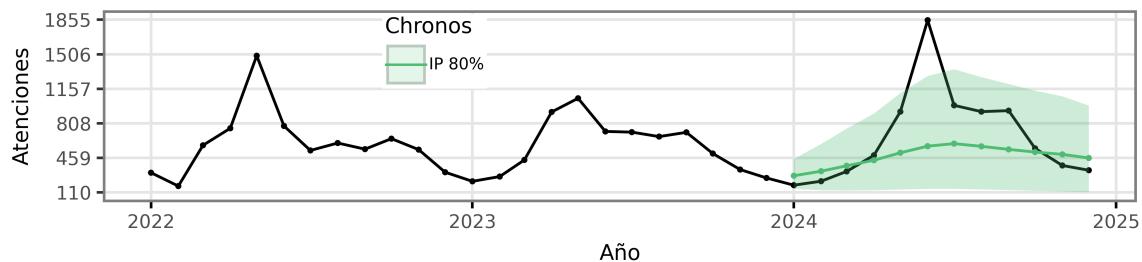


Figura 46: Pronóstico con Chronos sobre el número de atenciones en guardia por patologías respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

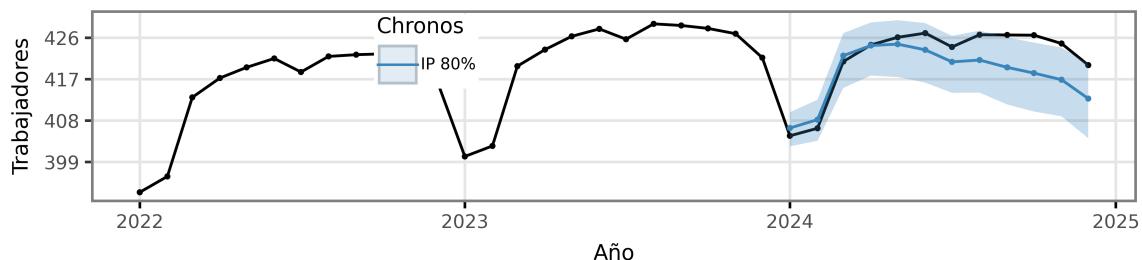


Figura 47: Pronóstico con Chronos sobre el número de trabajadores asalariados (miles) en el rubro de la enseñanza privada en Argentina.

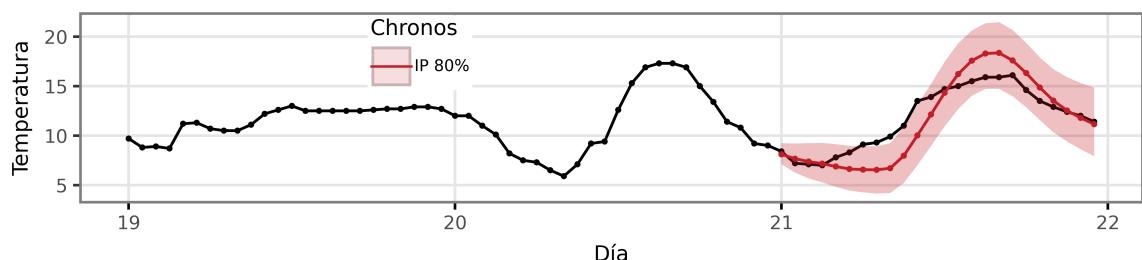


Figura 48: Pronóstico con Chronos sobre las temperaturas por hora en la ciudad de Rosario.

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

Serie	Hor.	MAPE	<i>Interval Score</i>
Atenciones	3	0.3838	463.7966
	6	0.3986	1654.112
	12	0.3592	1348.8879
Trabajadores	3	0.0318	117.8766
	6	0.0212	67.4687
	12	0.0097	15.1275
Temperatura	6	0.3489	26.5617
	12	0.4502	49.5046
	24	0.125	6.2749

Chronos logró capturar adecuadamente las variaciones diarias de temperatura, mostrando un buen ajuste en todos los horizontes probados. Sin embargo, su rendimiento fue inferior en las series de atenciones y trabajadores, donde los pronósticos puntuales se alejaron significativamente de los reales en comparación con los resultados obtenidos por otros modelos.

4.3 Comparación de resultados y análisis final

Con el propósito de evaluar el desempeño relativo de los distintos modelos de pronóstico, se realizó una comparación sistemática entre sus métricas de error y de calidad de intervalos para cada una de las tres series analizadas. En todos los casos, se consideraron horizontes de predicción de corto, mediano y largo plazo, con intervalos de predicción con una probabilidad de cobertura del 80%. Como se mencionó previamente, las métricas empleadas fueron el *Mean Absolute Percentage Error* (MAPE), que mide la precisión media del pronóstico, y el *Interval Score*, que evalúa la amplitud y la cobertura de los intervalos de predicción.

Serie 1. Atenciones en guardia por patologías respiratorias – Hospital de Niños Víctor J. Vilela (HNVV)

La Figura 49 muestra la comparación de métricas de evaluación entre los distintos modelos y horizontes de pronóstico para la serie del número de atenciones en guardia por patologías respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

Modelo	Horizonte 3		Horizonte 6		Horizonte 12	
	MAPE	Interval Score	MAPE	Interval Score	MAPE	Interval Score
ARIMA	0.1413	443.7176	0.2266	1143.1139	0.2566	1636.6127
XGBoost	1.1992	909.3306	1.6315	1895.0069	0.5257	1553.1046
LightGBM	1.0234	826.8506	1.6330	1893.8128	0.4509	1287.0798
LSTM	0.6990	886.5567	0.3871	1203.9912	0.8461	1469.5551
TimeGPT	1.7431	2774.7223	2.1404	5520.5854	0.2998	2037.7130
Chronos	0.3838	463.7966	0.3986	1654.1120	0.3592	1348.8879

Figura 49: Comparación de métricas de evaluación entre distintos modelos y horizontes para la serie de atenciones.

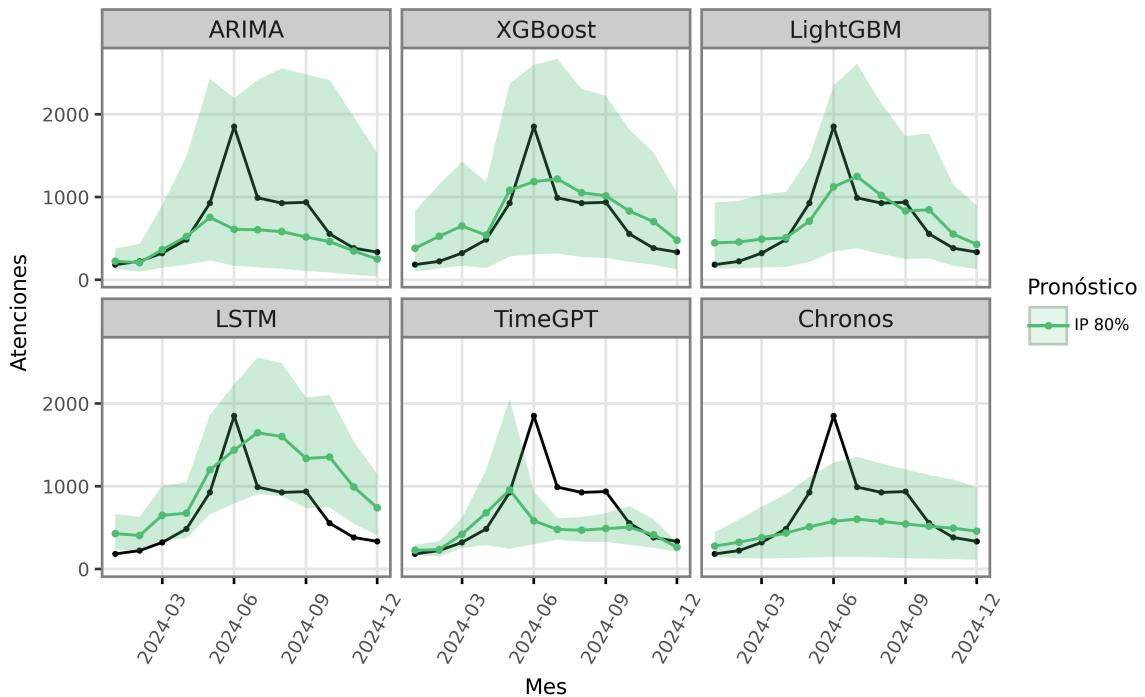


Figura 50: Comparación de pronósticos un año hacia adelante entre distintos modelos para la serie de atenciones.

El modelo ARIMA superó a los modelos más recientes en todos los horizontes probados. En el pronóstico a corto plazo la diferencia entre el modelo estadístico tradicional y los demás fue contundente, mientras que en el mediano plazo tanto LSTM como Chronos obtuvieron métricas similares. En el pronóstico un año hacia adelante, LightGBM supera al modelo ARIMA en términos de *Interval Score*, y TimeGPT consigue un MAPE próximo al conseguido por ARIMA.

En la Figura 50 se observa que, aunque el modelo ARIMA presenta un mejor desempeño según las métricas evaluadas, los modelos basados en árboles de decisión reproducen de manera más adecuada el patrón estacional, ofreciendo una representación más fiel de los valores observados. Esto refleja la importancia de la visualización de las series y sus pronósticos.

En términos generales, la serie de atenciones mensuales en guardia resultó la más difícil de pronosticar entre las tres analizadas, lo que se refleja en los valores elevados de MAPE. La presencia de alta variabilidad y episodios abruptos, particularmente durante los meses de invierno y el período de pandemia, limitó la capacidad de los modelos para capturar patrones regulares.

Serie 2. Trabajadores asalariados del sector educativo privado

La Figura 51 presenta la comparación de métricas de evaluación entre modelos y horizontes para la serie del número de trabajadores asalariados en el rubro de la enseñanza privada.

Modelo	Horizonte 3			Horizonte 6			Horizonte 12		
	MAPE	Interval Score		MAPE	Interval Score		MAPE	Interval Score	
ARIMA	0.0071	29.1066	0.0068		10.3768	0.0085		14.8479	
XGBoost	0.0083	20.5140	0.0091		20.5140	0.0069		21.9308	
LightGBM	0.0144	32.4170	0.0092		23.8802	0.0089		25.2876	
LSTM	0.0075	20.8112	0.0058		5.6342	0.0070		14.3802	
TimeGPT	0.0015	2.6232	0.0025		11.1076	0.0101		19.6513	
Chronos	0.0318	117.8766	0.0212		67.4687	0.0097		15.1275	

Figura 51: Comparación de métricas de evaluación entre distintos modelos y horizontes para la serie de trabajadores.

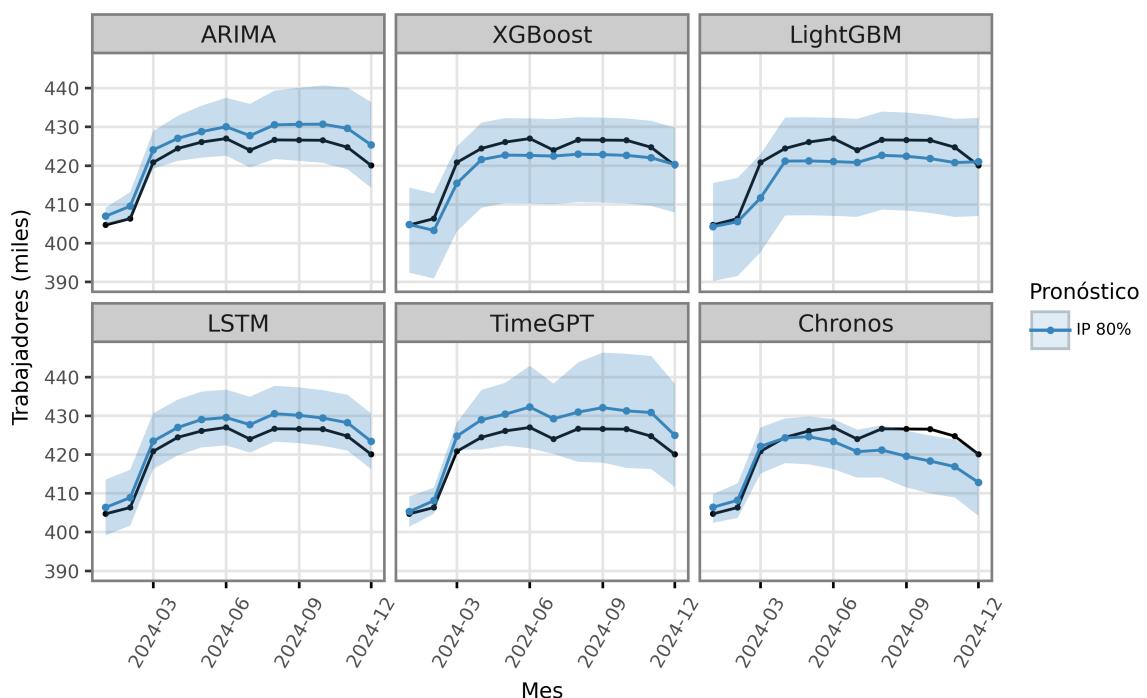


Figura 52: Comparación de pronósticos un año hacia adelante entre distintos modelos para la serie de trabajadores.

Se destacaron principalmente TimeGPT en los horizontes cortos y LSTM en los horizontes más largos. Esta diferencia se asocia con la estructura de los modelos: mientras TimeGPT, basado en transformadores, capta con mayor precisión la dinámica de corto plazo, LSTM logra un mejor desempeño en horizontes extendidos gracias a su capacidad de retener dependencias de largo alcance en la secuencia temporal.

La Figura 52 muestra como TimeGPT y Chronos pronostican con mayor exactitud que otros modelos los primeros meses, sin embargo, a medida que aumenta el tiempo los pronósticos puntuales se desvían y los intervalos de predicción se ensanchan. Por otro lado, LSTM genera pronósticos estables a través del tiempo y con poca incertidumbre.

En este caso, la estabilidad de la serie y la clara repetición anual de los ciclos facilitaron el ajuste de modelos tanto clásicos como de aprendizaje profundo, logrando métricas de error reducidas en todos los escenarios.

Serie 3. Temperatura por hora en la ciudad de Rosario

En la tercera serie se incorporaron la humedad y la presión atmosférica como variables exógenas, a fin de mejorar la capacidad explicativa del modelo.

La Figura 53 muestra para los pronósticos de la serie de temperatura por hora en la ciudad de Rosario la comparación de métricas entre los modelos evaluados para distintos horizontes.

Modelo	Horizonte 6			Horizonte 12			Horizonte 24		
	MAPE	Interval Score	MAPE	Interval Score	MAPE	Interval Score	MAPE	Interval Score	MAPE
ARIMA	0.0829	4.8344	0.1869	7.3556	0.1974	10.9150			
XGBoost	0.2775	35.6635	0.2995	36.3026	0.0583	2.7636			
LightGBM	0.2650	33.1287	0.2507	29.8732	0.0931	4.6673			
LSTM	0.0804	1.9593	0.1896	6.6552	0.1690	7.2241			
TimeGPT	0.2380	9.2120	0.1219	5.1091	0.2625	18.0171			
Chronos	0.3489	26.5617	0.4502	49.5046	0.1250	6.2749			

Figura 53: Comparación de métricas de evaluación entre distintos modelos y horizontes para la serie de temperaturas.

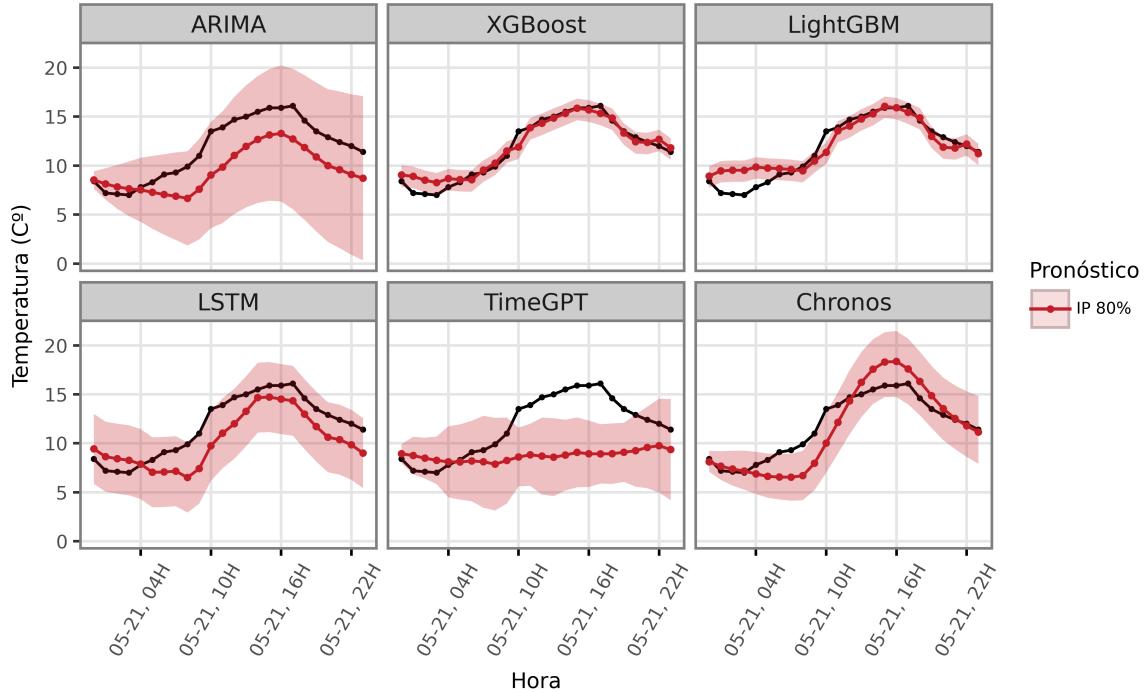


Figura 54: Comparación de pronósticos un día hacia adelante entre distintos modelos para la serie de temperaturas.

Tanto ARIMA como LSTM obtuvieron resultados superiores al resto de los modelos en el corto plazo, y fueron superados únicamente por TimeGPT en el pronóstico a 6 horas. Por otro lado, en el horizonte de un día completo, los modelos basados en *boosting* fueron los que obtuvieron mejores métricas.

La gran diferencia entre los pronósticos de los algoritmos basados en aprendizaje automático y el resto de modelos se visualiza mejor en la Figura 54. Tanto XGBoost como LightGBM realizan pronósticos acertados y con baja incertidumbre. Se destacan también la falta de ajuste de TimeGPT y la gran amplitud en los intervalos generados por el modelo SARIMAX elegido.

Este comportamiento sugiere que los modelos no lineales, especialmente los basados en árboles de decisión, logran capturar de manera más eficaz las interacciones entre temperatura, humedad y presión cuando se amplía el horizonte temporal.

5. Conclusiones

Se evidenció que ningún modelo es universalmente superior al resto ni existe combinación única de hiperparámetros que logre el mejor ajuste en cualquier circunstancia. A lo largo del trabajo se observaron las diferencias, ventajas y desventajas de cada método.

Los modelos ARIMA requieren un ajuste manual, que además de ser demandante temporalmente, su aplicación a diferentes series implicaría comenzar desde el inicio. En términos de resultados, logró MAPEs bajos en comparación al resto de modelos en casi todos los escenarios, destacándose principalmente en la serie de atenciones en guardia por patologías respiratorias.

Los algoritmos de aprendizaje automático precisan la definición de una serie de características que facilitan a los modelos a ajustarse a los datos. Ante nuevas series, sería prudente cambiar las características a otras más acordes a los datos. En relación a otros modelos, no se destacaron especialmente en ninguna instancia, con excepción de XGBoost pronosticando a largo plazo la serie de temperaturas. No obstante, los algoritmos de aprendizaje automatizado obtuvieron resultados acertados consistentemente en todas las series analizadas y con baja incertidumbre.

El único requerimiento para ajustar redes neuronales LSTM es la definición de la estructura del modelo, ya que tanto el ajuste de hiperparámetros como la elección de características se realizan de forma automática. A pesar de haber sido el modelo de pronóstico más exigente computacionalmente, obtuvo métricas favorables a través de todas las series y horizontes.

Finalmente, los modelos basados en arquitecturas *transformer*, como TimeGPT y Chronos, no precisan de ningún ajuste manual, únicamente los datos y su periodicidad son necesarios como entrada para los modelos. TimeGPT tuvo problemas para detectar los patrones estacionales de las series, sin embargo, logró buenas métricas en el corto y medio plazo. Chronos, por otro lado, presentó resultados menos satisfactorios. Esto podría solucionarse eligiendo otro modelo de la familia de Chronos diferente a **bolt-small**, o haciendo uso del ajuste fino. Cabe destacar que, si bien TimeGPT y Chronos destacan por su facilidad de uso, son poco personalizables, dejando pocas alternativas ante ajustes deficientes. TimeGPT tiene además la problemática de no ser de código abierto, lo que evita conocer el proceso con el que se ajustan los modelos.

A la fecha en la que se comenzó este trabajo, la API de TimeGPT ofrecía un plan gratuito con un número de consultas mensuales limitadas, sin embargo, a lo largo del desarrollo los planes de suscripción cambiaron. Se eliminó el plan gratuito y se reemplazó por un mes de prueba con consultas limitadas para usuarios nuevos únicamente. Este cambio afectó significativamente el desarrollo del trabajo.

Este documento presenta tres contribuciones claves al campo de la estadística. En primer lugar, la introducción de los modelos transformadores para el pronóstico de series temporales. En segundo lugar, se exploró el *Interval Score* como medida del error para evaluar el desempeño de los modelos ante pronósticos probabilísticos. Por último, se presenta la construcción de intervalos de pronóstico por medio de *conformal predictions*, que no depende de conocer la distribución de los residuos.

6. Mejoras y extensiones 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 numerosos temas interesantes que se podrían tratar en otros trabajos.

En primer lugar, en la selección de los modelos se utilizó el MAPE como métrica para elegir el mejor ajuste, sin embargo, la elección se pudo haber realizado en base al *Interval Score* o alguna otra medida de error. A su vez, podría explorarse el uso de distintas medidas de error probabilísticas diferentes al *Interval Score*, tales como *Scaled quantile loss*, *Weighted quantile loss* o *Implicit quantile loss*.

El ensamble de modelos, como *Boosting* o *Bagging*, no está limitado únicamente a los árboles de decisión, por lo que se propone indagar 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 existen otras alternativas, como *Natural Gradient Boosting* (NGBoost).

Al comparar los modelos también se pudo haber estudiado el tiempo de cómputo en los ajustes. Una prueba de esto fue realizada, pero los métodos utilizados para obtener estos resultados no fueron del todo adecuados, y ante la falta de alternativas se decidió recortar estos resultados y no mostrarlos empíricamente. Sin embargo, se hicieron menciones a los resultados a lo largo del trabajo. A raíz de esto, se propone estudiar distintas metodologías para medir los tiempos de cómputo de los modelos.

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 abordar 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. También se puede explorar la aplicación del ajuste fino en modelos fundacionales preentrenados.

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>
- Keith, M.** (22 de septiembre de 2023). *Five Practical Applications of the LSTM Model for Time Series, with Code*. Towards data science. <https://towardsdatascience.com/five-practical-applications-of-the-lstm-model-for-time-series-with-code-a7aac0aa85c0/>
- 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-langs-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 Salidas de modelos ARIMA

Tabla 9: Salida del modelo $SARIMA(0, 1, 0)(0, 1, 1)_{12}$ para el logaritmo del número de atenciones en guardia por enfermedades respiratorias en el Hospital de Niños Víctor J. Vilela de la ciudad de Rosario.

Componente	Coeficiente	IC(0.025)	IC(0.975)	p-value
intercept	0.003600	-0.055800	0.062900	0.906600
ma.S.L12	-0.586100	-0.857400	-0.314800	0.000000
sigma2	0.166400	0.107200	0.225600	0.000000
Modelo	$SARIMA(0, 1, 0)(0, 1, 1, 12)$		AIC	72.653300

Tabla 10: Salida del modelo $SARIMA(1, 1, 0)(1, 0, 1)_{12}$ para la serie del número de trabajadores asalariados del sector educativo privado en Argentina.

Componente	Coeficiente	IC(0.025)	IC(0.975)	p-value
intercept	0.000100	-0.001600	0.001700	0.941500
ar.L1	0.348400	0.203200	0.493700	0.000000
ar.S.L12	0.999700	0.994700	1.004700	0.000000
ma.S.L12	-0.906100	-1.620800	-0.191300	0.013000
sigma2	2.717600	1.205200	4.230000	0.000400
Modelo	$SARIMA(1, 1, 0)(1, 0, 1, 12)$		AIC	423.847900

Tabla 11: Salida del modelo $SARIMAX(1, 1, 1)(2, 0, 1)_{24}$ para la serie de temperaturas por hora en la ciudad de Rosario.

Componente	Coeficiente	IC(0.025)	IC(0.975)	p-value
intercept	-0.000859	-0.013286	0.011568	0.892200
HUM	-0.007626	-0.010491	-0.004761	0.000000
PNM	-0.133841	-0.252529	-0.015154	0.027100
ar.L1	0.225100	0.035248	0.414951	0.020100
ma.L1	0.180523	-0.010026	0.371073	0.063300
ar.S.L24	0.979171	0.807724	1.150618	0.000000
ar.S.L48	-0.033935	-0.169718	0.101849	0.624300
ma.S.L24	-0.749386	-0.894063	-0.604709	0.000000
sigma2	0.496032	0.447138	0.544926	0.000000
Modelo	$SARIMA(1, 1, 1)(2, 0, 1, 24)$		AIC	1067.051400

Tabla 12: Salida del modelo $SARIMAX(1, 1, 0)(1, 0, 1)_{24}$ para la serie de temperaturas por hora en la ciudad de Rosario.

Componente	Coeficiente	IC(0.025)	IC(0.975)	p-value
intercept	0.000200	-0.003100	0.003400	0.918700
HUM	-0.007300	-0.010100	-0.004600	0.000000
PNM	-0.128700	-0.246700	-0.010800	0.032400
ar.L1	0.496900	0.431000	0.562800	0.000000
ar.S.L24	0.981200	0.952400	1.010100	0.000000
ma.S.L24	-0.882500	-0.981300	-0.783700	0.000000
sigma2	0.483800	0.434900	0.532600	0.000000
Modelo	SARIMA(1, 1, 0)(1, 0, 1, 24)		AIC	1047.239600