GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

MODELO PREDICTIVO DEL PRECIO DE BILLETES DE AVIÓN MEDIANTE LA COMPARACIÓN DE MÉTODOS DE MODELADO TEMPORAL

Autor: Pablo Alonso Sánchez

Director: Emanuel Gastón Mompó Pavesi

Madrid

Declaración de originalidad

Declaro bajo mi responsabilidad que el Proyecto presentado con el título **Modelo predictivo del precio de billetes de avión mediante la comparación de métodos de modelado temporal** e la ETS de Ingeniería – ICAI de la Universidad Pontificia Comillas en el curso académico **2025-2026** es de mi autoría y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Uso de Inteligencia Artificial[1]

Declaro bajo mi responsabilidad que (indicar la opción correcta):

☐ No he utilizado Inteligencia Artificial en la elaboración del presente documento.

■ He utilizado Inteligencia Artificial en la elaboración del presente documento y/o del Anexo B siempre en las condiciones permitidas por la Universidad Pontificia Comillas, es decir, aplicando el Nivel 2 de la Escala de Evaluación de Perkins et al. (2024): "*La IA puede utilizarse para actividades previas a la tarea, como la lluvia de ideas, la descripción y la investigación inicial. Este nivel se centra en el uso de la IA para la planificación, las síntesis y la generación de ideas, pero las evaluaciones deben hacer hincapié en la capacidad de desarrollar y refinar estas ideas de forma independiente*". En concreto, las Inteligencia Artificial ha sido empleada para:

La Inteligencia Artificial se ha utilizado para la comprensión de conceptos técnicos relacionados con el modelado temporal, la investigación preliminar del tema, la generación de ideas iniciales y la exploración de posibles enfoques metodológicos. Asimismo, se ha empleado como apoyo para la planificación de la estructura del proyecto, la organización de tareas y la resolución de dudas conceptuales puntuales surgidas durante su desarrollo.

[1] Esta declaración se refiere al uso de la Inteligencia Artificial generativa para realizar los documentos del Proyecto (Anexo B y Memoria). No aplica a Proyectos donde, por su naturaleza, deban emplear inteligencia artificial como parte de los mismos (aplicación de técnicas de aprendizaje automático, redes neuronales, análisis de datos…)


Firmado (alumno): Pablo Alonso Sánchez
Fecha: 18/06/2026

Autorización para la entrega del Proyecto

El Director del Proyecto	El co-Director del Proyecto (si aplica)
Firmado por MOMPO PAVESI EMANUEL GASTON - ***0224** el día 19/06/2026 con un certificado emitido por AC FNMT Usuarios	
Fdo:	Fdo:
Fecha:	Fecha:

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

MODELO PREDICTIVO DEL PRECIO DE BILLETES DE AVIÓN MEDIANTE LA COMPARACIÓN DE MÉTODOS DE MODELADO TEMPORAL

Autor: Pablo Alonso Sánchez

Director: Emanuel Gastón Mompó Pavesi

Madrid

Agradecimientos

A mis padres, por la oportunidad que me han dado y por los valores, el esfuerzo y el apoyo que siempre han transmitido a sus hijos.

A mis hermanos, por su ejemplo de fortaleza, resiliencia y perseverancia.

MODELO PREDICTIVO DEL PRECIO DE BILLETES DE AVIÓN MEDIANTE LA COMPARACIÓN DE MÉTODOS DE MODELADO TEMPORAL

Autor: Alonso Sánchez, Pablo.
Director: Mompó Pavesi, Emanuel Gastón.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

La previsión del precio de un billete de avión se ha convertido en una preocupación habitual para el viajero promedio. Este trabajo compara cuatro métodos de predicción (Naive, ARIMA, LSTM y XGBoost) aplicados a los precios de los vuelos con origen en Madrid y concluye que los precios se comportan como un paseo aleatorio en horizontes cortos, siendo la ruta Madrid-Barcelona una excepción reproducible y el eje de días hasta la salida la única regularidad aprovechable, con un recargo del 10 % en las dos últimas semanas antes del vuelo.

Palabras clave: Precios de los billetes de avión, previsiones, series temporales, paseo aleatorio, ARIMA, LSTM, extracción de datos de la web.

1. Introducción

El transporte aéreo se ha convertido en un medio de transporte esencial, y conseguir billetes a precios asequibles es una preocupación recurrente para los viajeros. Dos cuestiones han marcado la línea académica sobre este tema: si se pueden predecir los precios de los billetes de avión y si existe un momento óptimo para comprarlos. Etzioni e investigadores (Etzioni et al., 2003) iniciaron esta línea de investigación, y las dos décadas de investigación posteriores han generado resultados diferentes según las rutas, los mercados y los métodos elegidos.

Una carencia común a toda esa investigación es la ausencia de un punto de referencia evidente. Los estudios suelen presentar cifras de error absoluto en torno al 9 o al 10 % sin compararlas con una previsión *naïve* que se limite a repetir el precio del día anterior, lo que dificulta saber si las mejoras observadas son significativas en la práctica. El presente trabajo se diseñó para subsanar esa carencia mediante una comparación rigurosa de los métodos de previsión clásicos y modernos sobre los mismos datos, bajo el mismo protocolo de evaluación, utilizando la previsión *naïve* como referencia explícita en todas las métricas.

2. Definición del proyecto

El proyecto plantea cinco objetivos específicos. El primero es la creación de dos conjuntos de datos mediante la intercepción de la API GraphQL de *Kiwi.com* siguiendo dos estrategias de extracción: una sesión única en seis rutas (conjunto de datos A) y una campaña longitudinal de cincuenta días en cuatro rutas con siete fechas de salida cada una (conjunto de datos B). El segundo es la aplicación de un proceso ARIMA con la metodología completa de Box-Jenkins a ambos conjuntos de datos, incluyendo pruebas de dual estacionariedad, selección automatizada del orden y diagnóstico de residuos. El

tercero es la implementación de una LSTM multivariante con características diseñadas en el conjunto de datos B. El cuarto es el análisis de la relación entre el momento de la compra y el precio pagado. El quinto es la documentación de las consecuencias metodológicas de las dos estrategias de scraping. La cuestión se plantea de forma deliberada como una comparación, más que como una búsqueda del modelo más preciso. El objetivo es establecer en qué rutas, horizontes y condiciones los métodos más elaborados aportan valor predictivo respecto a la línea de base más sencilla posible.

3. Descripción del modelo

El sistema de recopilación de datos utiliza Playwright para controlar un navegador Chromium e intercepta las respuestas GraphQL emitidas por el punto final de búsqueda de *Kiwi.com*, extrayendo el precio más barato del billete para cada combinación de ruta y fecha de salida. El conjunto de datos A contiene en torno a 198 observaciones por ruta en seis destinos (Barcelona, Berlín, París-CDG, Estambul, Londres-Heathrow y Singapur), todos ellos consultados en una única sesión. El conjunto de datos B contiene 1371 observaciones reales repartidas en 28 series individuales de fechas de vuelo recopiladas diariamente entre el 8 de abril y el 5 de junio de 2026, agregadas también en cuatro series a nivel de ruta para el modelado clásico.

Los cuatro métodos evaluados son: una previsión base como referencia, un proceso ARIMA con selección de orden mediante `auto_arima`, una LSTM de 32 unidades entrenada en una ventana de 7 días sobre cuatro características continuas (precio, días hasta la salida, seno y coseno del día de la semana) más una codificación de rutas *one-hot*, y un modelo XGBoost con 500 árboles. Todos se evalúan en cuatro horizontes (1, 3, 7 y 14 días) según cuatro métricas de error (MAE, RMSE, MAPE y MASE). El esquema general se muestra en la *Ilustración 1*.

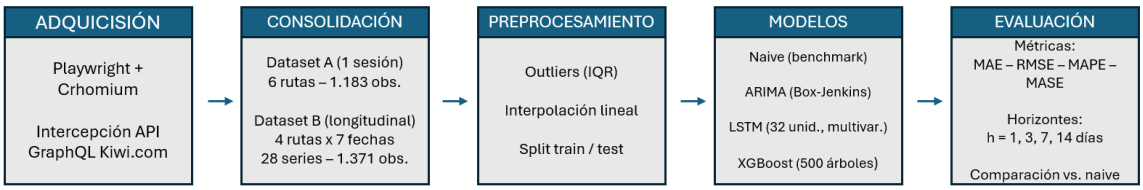

Ilustración 1. Esquema del flujo de trabajo del sistema.

4. Resultados

Las series de precios estudiadas se comportan, en la ventana diaria previa a la salida, como paseos aleatorios. Su evolución está dominada por ruido a corto plazo que ninguno de los métodos más elaborados capta mejor que la previsión ingenua. En términos generales, tanto LSTM como XGBoost igualan a la previsión ingenua en $h=1$, convergen hacia ella en $h=3$ y obtienen peores resultados que está en $h=7$ y $h=14$, con penalizaciones de entre el 13 % y el 22 %, dependiendo del horizonte y del método. El análisis por ruta muestra que este patrón no es uniforme: de las 32 combinaciones de modelo-ruta-horizonte, siete superan a la predicción ingenua, y cuatro de esas siete se concentran en la ruta Madrid-Barcelona. ARIMA también supera al modelo base en esa

misma ruta en un 15,5 % en el RMSE de prueba en *h=1*, por lo que MAD-BCN se perfila, según tres métodos independientes (ARIMA, LSTM y XGBoost), como la única ruta del análisis que presenta una estructura realmente previsible más allá de la continuidad (*Ilustración 2*).

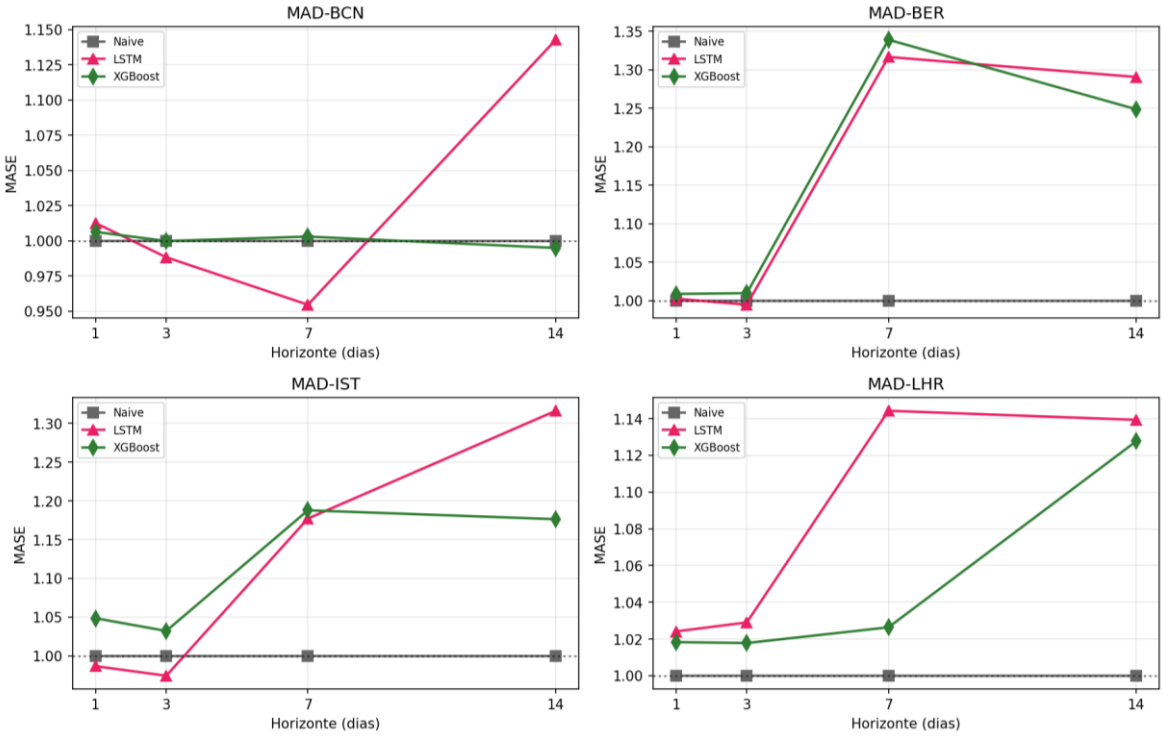

Ilustración 2. MASE por ruta y horizonte para LSTM y XGBoost, tomando como referencia horizontal el valor 1,0 del modelo de referencia naive.

Se comprobaron dos regularidades transversales. La prueba de Kruskal-Wallis aplicada a los residuos sin tendencia por día de la semana arroja un valor de H = 3,41 con p = 0,755, lo que descarta cualquier efecto calendario aprovechable. El análisis de los días previos a la salida ofrece un panorama diferente: tres enfoques estadísticos independientes coinciden en la misma dirección. Los análisis descriptivos sitúan el periodo más barato entre 46 y 60 días antes de la salida, y los tres enfoques coinciden en que las dos últimas semanas antes del vuelo conllevan un recargo de aproximadamente el 10 % sobre el precio típico de la serie.

5. Conclusiones

Los resultados aportan un soporte empírico a la hipótesis del paseo aleatorio (Fama, 1965) aplicada a la dinámica de las tarifas aéreas a corto plazo, un resultado previsto por la teoría de la economía financiera pero que aquí se documenta comparándolo con un punto de referencia ingenuo explícito y desde cuatro ángulos de análisis independientes. La excepción de la ruta Madrid-Barcelona, identificada de forma consistente por tres métodos de predicción independientes, apunta a una ruta que merece un estudio más detallado con ventanas de prueba más largas y datos adicionales. La normalidad de los precios en función de los días que faltan para la salida se traduce en una única

recomendación práctica y sólida: evitar comprar billetes en las dos últimas semanas antes de la salida, cuando los precios se sitúan aproximadamente un 10 % por encima del nivel habitual. La documentación metodológica del efecto de cardinalidad producido por la recopilación de datos en una sola sesión se deja como una contribución separada para cualquiera que intente replicar estudios similares en el futuro.

6. Referencias

Belobaba, P., Odoni, A., & Barnhart, C. (2015). *The Global Airline Industry* (2nd ed.). Wiley.

Box, G. E. P., & Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day.

Etzioni, O., Tuchinda, R., Knoblock, C. A., & Yates, A. (2003). To buy or not to buy: mining airfare data to minimize ticket purchase price. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 119–128.

Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business*, 38(1), 34–105.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

PREDICTIVE MODEL FOR AIRFARE PRICES USING A COMPARISON OF TIME-SERIES MODELING METHODS

Author: Alonso Sánchez, Pablo.
Supervisor: Mompó Pavesi, Emanuel Gastón.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

Predicting the price of a plane ticket has become a common concern for the average traveler. This study compares four forecasting methods (Naive, ARIMA, LSTM, and XGBoost) applied to flight prices for flights departing from Madrid and concludes that prices behave like a random walk over short time horizons, with the Madrid-Barcelona route being a reproducible exception and the number of days until departure being the only exploitable pattern, featuring a 10% surcharge in the last two weeks before the flight.

Keywords: Airfare, forecasting, time series, random walk, ARIMA, LSTM, web scraping.

1. Introduction

Air travel has become an essential mode of transportation, and finding affordable tickets is a recurring concern for travelers. Two questions have shaped the academic discourse on this topic: whether airfare prices can be predicted and whether there is an optimal time to purchase tickets. Etzioni and colleagues (Etzioni et al., 2003) pioneered this line of research, and the two decades of research that followed have yielded varying results depending on the routes, markets, and methods used.

A common shortcoming across all this research is the absence of a clear benchmark. Studies typically report absolute error figures of around 9% or 10% without comparing them to a *naïve* forecast that simply repeats the previous day's price, making it difficult to determine whether the observed improvements are practically significant. This study was designed to address that shortcoming through a rigorous comparison of classical and modern forecasting methods applied to the same data, under the same evaluation protocol, using the *naïve* forecast as an explicit benchmark across all metrics.

2. Project Definition

The project sets out five specific objectives. The first is the creation of two datasets by scraping the GraphQL API of *Kiwi.com* using two extraction strategies: a single session across six routes (dataset A) and a 50-day longitudinal campaign across four routes, each with seven departure dates (dataset B). The second is to apply an ARIMA process using the full Box-Jenkins methodology to both datasets, including tests for dual stationarity, automated order selection, and residual diagnostics. The third is to implement a

multivariate LSTM with designed features on dataset B. The fourth is to analyze the relationship between the time of purchase and the price paid. The fifth is the documentation of the methodological implications of the two scraping strategies. The question is deliberately framed as a comparison, rather than a search for the most accurate model. The goal is to determine under which paths, time horizons, and conditions the more elaborate methods provide predictive value relative to the simplest possible baseline.

3. Model Description

The data collection system uses Playwright to control a Chromium browser and intercepts the GraphQL responses issued by the *Kiwi.com* search endpoint, extracting the cheapest ticket price for each combination of route and departure date. Dataset A contains approximately 198 observations per route across six destinations (Barcelona, Berlin, Paris-CDG, Istanbul, London-Heathrow, and Singapore), all queried in a single session. Dataset B contains 1,371 actual observations spread across 28 individual series of flight dates collected daily between April 8 and June 5, 2026, also aggregated into four route-level series for classical modeling.

The four methods evaluated are: a baseline forecast as a reference, an ARIMA process with order selection via `auto_arima`, a 32-unit LSTM trained on a 7-day window using four continuous features (price, days until departure, sine, and cosine of the day of the week) plus *one-hot* route encoding, and an XGBoost model with 500 trees. All models are evaluated over four horizons (1, 3, 7, and 14 days) using four error metrics (MAE, RMSE, MAPE, and MASE). The general framework is shown in *figure 3*.

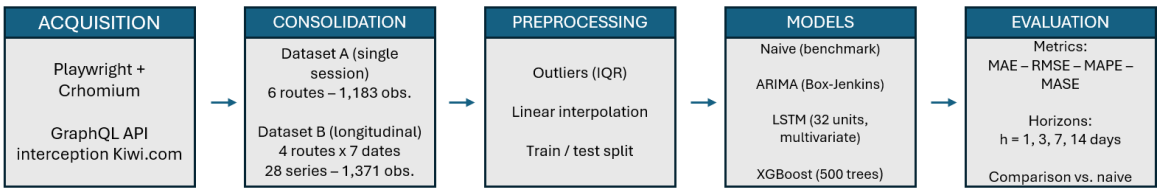

Figure 3. Pipeline scheme of the system.

4. Results

The price series studied behave like random walks in the daily window prior to the forecast. Their behavior is dominated by short-term noise that none of the more sophisticated methods captures any better than the naive forecast. In general terms, both LSTM and XGBoost match the naive forecast at $h=1$, converge toward it at $h=3$, and perform worse than the naive forecast at $h=7$ and $h=14$, with performance losses ranging from 13% to 22%, depending on the horizon and the method. The route-by-route analysis shows that this pattern is not uniform: of the 32 model-route-horizon combinations, seven outperform the naive forecast, and four of those seven are concentrated on the Madrid-Barcelona route. ARIMA also outperforms the baseline model on that same

route by 15.5% in the test RMSE at *h=1*, so MAD-BCN emerges, according to three independent methods (ARIMA, LSTM, and XGBoost), as the only route in the analysis that exhibits a truly predictable structure beyond mere continuity (*Figure 4*).

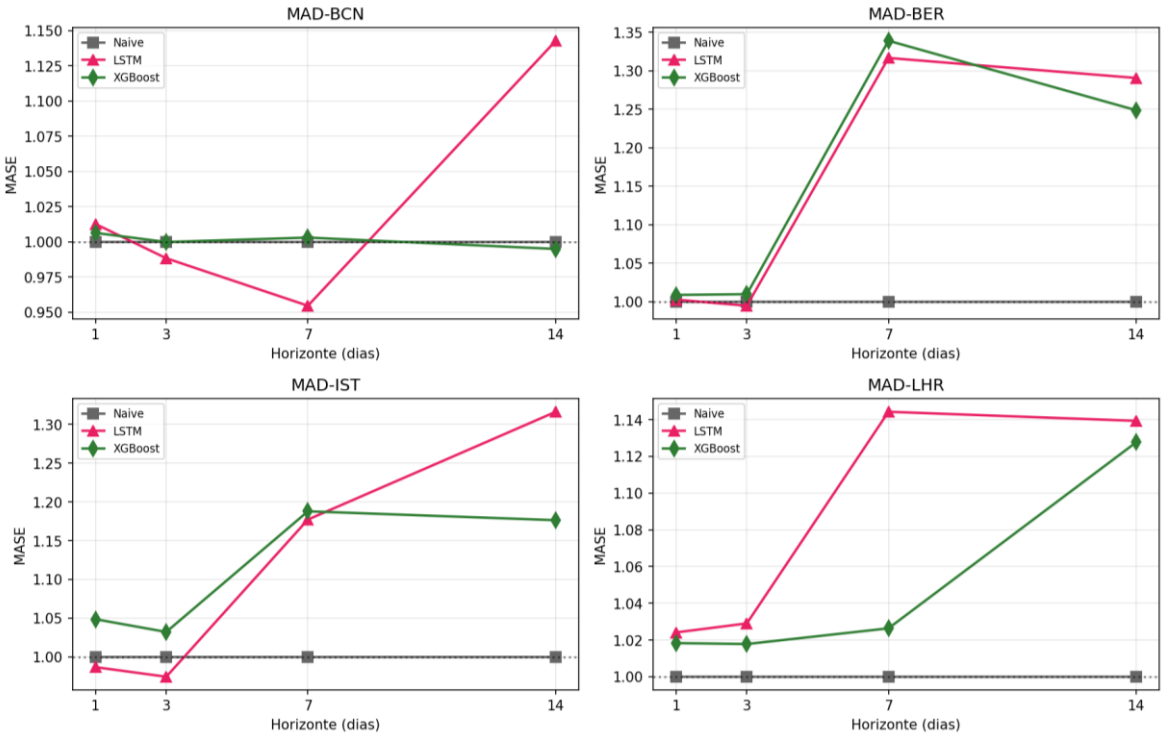

Figure 4. MASE per route and horizon for LSTM and XGBoost, with the naive baseline as the horizontal reference at 1.0.

Two cross-sectional patterns were identified. The Kruskal-Wallis test applied to the trend-free residuals by day of the week yielded a value of H = 3.41 with p = 0.755, which rules out any exploitable calendar effect. Analysis of the days leading up to departure paints a different picture: three independent statistical approaches point in the same direction. Descriptive analyses place the cheapest period between 46 and 60 days before departure, and all three approaches agree that the last two weeks before the flight carry a surcharge of approximately 10% on the typical price in the series.

5. Conclusions

The results provide empirical support for the random walk hypothesis (Fama, 1965) as applied to the short-term dynamics of airfares—a finding predicted by financial economics theory but documented here through comparison with an explicit naive benchmark and from four independent analytical perspectives. The exception of the Madrid–Barcelona route, consistently identified by three independent forecasting methods, points to a route that warrants a more detailed study with longer test windows and additional data. The normal distribution of prices as a function of the number of days remaining until departure leads to a single practical and robust recommendation: avoid purchasing tickets in the last two weeks before departure, when prices are approximately 10% above the usual level. The methodological documentation of the cardinality effect

resulting from data collection in a single session is provided as a separate contribution for anyone attempting to replicate similar studies in the future.

6. References

Belobaba, P., Odoni, A., & Barnhart, C. (2015). *The Global Airline Industry* (2nd ed.). Wiley.

Box, G. E. P., & Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day.

Etzioni, O., Tuchinda, R., Knoblock, C. A., & Yates, A. (2003). To buy or not to buy: mining airfare data to minimize ticket purchase price. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 119–128.

Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business*, 38(1), 34–105.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

Índice de la memoria

Índice de figuras

Índice de tablas

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

Un viajero que compare el precio del mismo vuelo en dos días consecutivos observará con bastante frecuencia una cifra diferente el segundo día. Los precios de los billetes de avión cambian continuamente, a veces varias veces al día en la misma ruta, y lo hacen por razones que rara vez son visibles para la persona que realiza la compra. Detrás de esa aparente volatilidad se esconde toda una disciplina de fijación dinámica de precios conocida como gestión de ingresos, formalizada en la década de 1980 tras la desregulación del mercado del transporte aéreo estadounidense y tratada en profundidad en obras de referencia como la de Belobaba, (Belobaba et al., 2015). Cada asiento es un producto perecedero, y su precio se ajusta en tiempo real para maximizar los ingresos antes de que el avión despegue. El resultado, desde el punto de vista del comprador, es una curva de precios que varía continuamente y sin una causa evidente detrás de cada cambio.

La asimetría es evidente entre vendedor y comprador ha motivado la aparición de herramientas comerciales que intentan ayudar al viajero a lidiar con la volatilidad. Los busacdores como Skyscanner, Kayak y Google Flights recopilan precios de diferentes fuentes, mientras que aplicaciones como Hopper añaden recomendaciones del tipo “comprar o esperar” basadas en sus datos internos. Los algoritmos que hay detrás de estas herramientas son propios y no se publican, lo que significa que nadie ajeno a la empresa puede verificar sus afirmaciones sobre la precisión. En cambio, los textos académicos son públicos y reproducibles. Cabe destacar la investigación de hace dos décadas, que se remontan al artículo fundacional de Etzioni e investigadores (Etzioni et al., 2003), han demostrado que los datos sobre tarifas aéreas de acceso público pueden recopilarse y convertirse en modelos predictivos, aunque los resultados publicados varían considerablemente según las rutas, los mercados y las opciones de modelización.

El trabajo presentado en esta memoria toma esa bibliografía como punto de partida e intenta ampliarla en dos frentes específicos. El primero es la comparación rigurosa de métodos de

predicción clásicos y modernos sobre el mismo conjunto de datos y bajo el mismo protocolo de evaluación, con un punto de referencia explícito e ineludible en forma de una predicción ingenua. El segundo es un análisis detallado de cómo la propia estrategia de recopilación de datos influye en las propiedades estadísticas de las series resultantes, una cuestión que rara vez se aborda en la bibliografía existente. Ambos frentes se abordaron en el transcurso del proyecto y enmarcan el debate que sigue.

1.2 PLANTEAMIENTO DEL PROBLEMA

Este proyecto examina si la evolución a corto plazo de los precios de los billetes de avión con salida desde Madrid puede predecirse utilizando modelos estadísticos clásicos y arquitecturas modernas de aprendizaje automático y si los datos presentan regularidades aprovechables en las dimensiones del día de la semana y los días previos a la salida. Las series de precios se extraen de *Kiwi.com* mediante un rastreo web automatizado con interceptación de la API, en un conjunto de seis rutas europeas e intercontinentales para la estrategia de sesión única y cuatro rutas europeas para la estrategia longitudinal. Los métodos de predicción elegidos son una cadena de modelos ARIMA[2] que sigue la metodología clásica de Box-Jenkins, una red neuronal recurrente LSTM multivariante y un modelo de refuerzo de gradientes XGBoost, todos ellos evaluados frente a una referencia *naïve* que se limita a repetir el precio observado más reciente.

La cuestión se plantea deliberadamente como una comparación, más que como la búsqueda de un único modelo óptimo. A raíz de la laguna identificada en la bibliografía existente, donde con frecuencia se presentan cifras de precisión absoluta sin compararlas con la persistencia, lo que dificulta la interpretación del panorama resultante. Surge que el objetivo no es alcanzar a la arquitectura con el menor error, sino establecer en qué rutas, en qué horizontes temporales y en qué condiciones los métodos más elaborados aportan valor predictivo respecto a la referencia más simple posible.

[2] ARIMA, Acrónimo de *AutoRegressive Integrated Moving Average* (media móvil autorregresiva integrada), introducido en su formulación moderna por Box y Jenkins (1970). El modelo sigue siendo la especificación paramétrica estándar para la previsión de series temporales univariantes

1.3 CONTRIBUCIÓN PREVISTA

El trabajo pretende aportar tres contribuciones principales.

La primera es ofrecer una comparación rigurosa de cuatro métodos de previsión sobre un único conjunto de datos, siguiendo un protocolo de evaluación unificado, con la previsión ingenua como referencia explícita para todos los métodos. La segunda es documentar las consecuencias metodológicas de las dos estrategias de recopilación de datos adoptadas, incluidos los posibles efectos que el propio proceso de extracción de datos introduce en las series resultantes. El tercero es llevar a cabo un análisis aplicado de la relación entre el momento de la compra y el precio pagado, utilizando los datos longitudinales para identificar en qué momento del proceso de reserva los precios tienden a ser metódicamente más bajos. Estas tres contribuciones previstas se corresponden con los cinco objetivos específicos expuestos en el *capítulo 4* y se evalúan de nuevo, una vez se tengan los resultados, en el *capítulo 7*.

1.4 ESTRUCTURA DE LA MEMORIA

El resto del documento se organiza de la siguiente manera:

El *capítulo 2* presenta las tecnologías en las que se basa el trabajo. Extracción de datos web con interceptación de API, los fundamentos del análisis de series temporales, las pruebas estadísticas utilizadas para el diagnóstico, la familia de modelos ARIMA, las métricas de error empleadas en la evaluación y la arquitectura de la red neuronal LSTM. El *capítulo 3* revisa la literatura académica sobre la predicción de precios de vuelos, organizada cronológicamente por familia metodológica y que concluye con las lagunas de investigación que motivan el presente análisis. El *capítulo 4* expone la justificación del proyecto, sus objetivos específicos, la metodología adoptada y la planificación del proyecto. El *capítulo 5* describe detalladamente el sistema desarrollado, desde el proceso de extracción de datos y la construcción de los dos conjuntos de datos hasta el preprocesamiento, las características empeladas, los modelos implementados y el marco de evaluación. El *capítulo 6* presenta los resultados de la aplicación de estos métodos a los datos, incluidos los análisis del día de la

semana y del momento de la compra. El *capítulo 7* cierra la memoria con las conclusiones, las limitaciones identificadas durante el trabajo y las líneas en las que el análisis podría ampliarse en futuras investigaciones.

2 DESCRIPCIÓN DE LAS TECNOLOGÍAS

Este capítulo expone los fundamentos técnicos sobre los que se basa el resto del trabajo. El proyecto requiere la integración de varios campos que no suelen aparecer juntos en una sola investigación de grado, por lo que las secciones siguientes abordan cada uno de ellos con la profundidad necesaria para comprender las decisiones de modelización que se tomarán más adelante. La *sección 2.1* describe cómo se obtuvieron los datos sobre los precios de los vuelos mediante la interceptación de una API comercial a través de sesiones automatizadas en el navegador, un proceso conocido como *web scraping*. A continuación, las *secciones 2.2* y *2.3* introducen el lenguaje formal del análisis de series temporales y las pruebas estadísticas (ADF, KPSS, Ljung-Box, Shapiro-Wilk) que determinan si una serie es estacionaria, si los residuos de un modelo se comportan como ruido blanco y si las diferencias entre grupos son significativas. La *sección 2.4* aborda la familia de modelos ARIMA y el procedimiento de identificación de Box-Jenkins utilizado para ajustarlos a la serie de precios de los vuelos. La *sección 2.5* define las métricas de error (MAE, RMSE, MAPE, MASE) y explica por qué es necesario utilizar varias de ellas simultáneamente, prestando especial atención al MASE, que permite una comparación directa con el punto de referencia *naïve*. La *sección 2.6* presenta la arquitectura recurrente LSTM, el segundo método de predicción analizado en este trabajo, y la *sección 2.7* cierra el capítulo con una breve descripción general de las herramientas de software y las bibliotecas utilizadas a lo largo del proyecto.

2.1 WEB SCRAPING E INTERCEPTACIÓN DE API

El web scraping consiste en la extracción automatizada de información de sitios web mediante agentes de software que simulan el comportamiento del usuario humano que navega por la web. Como método de investigación, ha ganado un gran impulso durante la última década en campos que van desde la economía y las finanzas hasta el periodismo y las ciencias sociales. Principalmente porque permiten la recolección de grandes conjuntos de datos que no están disponibles de manera pública o a través de API oficiales (Khder, 2021).

Este trabajo de estudio se basa en el web scraping como único medio de adquisición de datos, y las decisiones técnicas tomadas al respecto se detallan a lo largo de esta sección.

El primer reto técnico que tiene que hacer frente un scraper moderno es la naturaleza de las páginas webs contemporáneas. Mientras que la web primitiva consistía en gran medida en documentos HTML estáticos servidos directamente por el servidor web, las páginas actuales se consideran de forma dinámica: el servidor devuelve inicialmente un documento HTML esquelético junto con código JavaScript, que luego se ejecutan en el navegador del cliente. Este script envía solicitudes asíncronas a las API internas, recuperar datos (normalmente en formato JSON) y construye los elementos visibles de la página sobre la marcha mediante la manipulación del Modelo de Objetos de Documento (DOM)[3]. Es por ello por lo que una solicitud ingenua a la URL de la página no produce mucho más que una plantilla vacía. Los datos de interés no están presentes hasta que la capa de JavaScript haya completado su trabajo.

Esta circunstancia marca la diferencia entre dos generaciones de herramientas de scraping. La primera generación, representada por la combinación en Python de *request* y *Beautifulsoup*[4], opera a nivel de HTML sin procesar y resulta muy adecuada para documentos estáticos, pero es fundamentalmente incapaz de acceder a contenidos generados dinámicamente (Mitchell, 2018). La segunda generación, que incluye Selenium[5], Pupperteer y Playwright, se basa en un motor de navegador real controlado mediante programación. El navegador ejecuta JavaScript de la misma manera que lo haría para un usuario humano, del cual se puede inspeccionar y analizar el DOM final.

[3] DOM (Document Object Model), Una representación en forma de árbol de un documento HTML o XML que utilizan los navegadores web para mostrar las páginas. Cada elemento del documento se convierte en un nodo del árbol, que puede consultarse y modificarse dinámicamente mediante JavaScript.
[4] BeautifulSoup, Una biblioteca de Python lanzada por primera vez en 2004, diseñada para analizar documentos HTML y XML. Permite recorrer el árbol del documento y extraer elementos por etiqueta, clase o atributo, y sigue siendo una de las herramientas clásicas para extraer datos de páginas web estáticas.
[5] Selenium, Un marco de automatización de navegadores desarrollado originalmente en 2004 como herramienta de pruebas. Permite el control programático de navegadores reales y se ha utilizado ampliamente para extraer contenido web dinámico, aunque su rendimiento suele ser inferior al de alternativas más modernas, como Playwright.

Existe, sin embargo, un tercer enfoque que resulta especialmente idóneo cuando el sitio de destino expone una API interna bien definida: la intercepción de respuestas. En lugar de analizar el HTML renderizado, el rastreador observa el tráfico de red generado por el navegador y captura las respuestas JSON devueltas por los propios puntos finales de la API del sitio. Los datos así obtenidos ya están estructurados, lo que elimina la necesidad de frágiles selectores CSS o expresiones XPath[6] y proporciona una robustez mucho mayor frente a cambios cosméticos en el front-end. Este enfoque es el adoptado en el presente trabajo.

La API interna utilizada por *Kiwi.com* se implementa mediante GraphQL, un lenguaje de consulta y entorno de ejecución desarrollado por Facebook y publicado como estándar abierto en 2015 (GraphQL Foundation, 2021). A diferencia de las API REST tradicionales, GraphQL expone un único punto final que el cliente envía consultas estructuradas especificando exactamente qué campos se requieren, y el servidor responde con el subconjunto de datos solicitado. En el caso de Kiwi.com, los resultados de la búsqueda de vuelos se obtienen a través de una consulta denominada `SeachOneWayItinerariesQuery` dirigida al punto final `api.skypicker.com/umbrela/v2/graphql`, que el rastreador intercepta y analiza.

Entre los marcos de automatización de navegadores disponibles, se seleccionó Playwright para este proyecto. Desarrollado por Microsoft y lanzado en 2020, Playwright tiene tres ventajas con respecto a sus predecesores que resultaron decisivas en el presente contexto: compatibilidad nativa, con distintos navegadores (Firefox, webkit y Chromium), una moderna API asíncrona basada en el paradigma `async/await` y con compatibilidad de primer nivel para interceptar respuestas de red a través del controlador de eventos `page.on("response",…)`. Este último es el mecanismo mediante el cual se captura el tráfico GraphQL de interés.

[6] XPath, Un lenguaje de consulta normalizado por el W3C que se utiliza para localizar nodos específicos dentro de un documento XML o HTML. Junto con los selectores CSS, constituye uno de los mecanismos estándar para identificar elementos en las técnicas clásicas de scraping.

Por lo tanto, el flujo de trabajo aplicado puede resumirse de la siguiente manera. Para cada combinación de ruta y fecha de vuelo de destino, el rastreador abre una instancia controlada de Chromium, navega a la URL de búsqueda correspondiente de Kiwi.com y registra una llamada de retorno en el evento response, filtrando por punto final y nombre de la consulta. Una vez que la página se ha cargado por completo y se ha capturado la respuesta GraphQL de interés, se analiza lla carga útil JSON para extraer, para cada itinerario, el precio total en euros, los nombres de las aerolíneas operadoras y el número de segmentos, del que se tiene el número de escalas. A continuación, se cierra el contexto del navegador y se procesa la siguiente combinación.

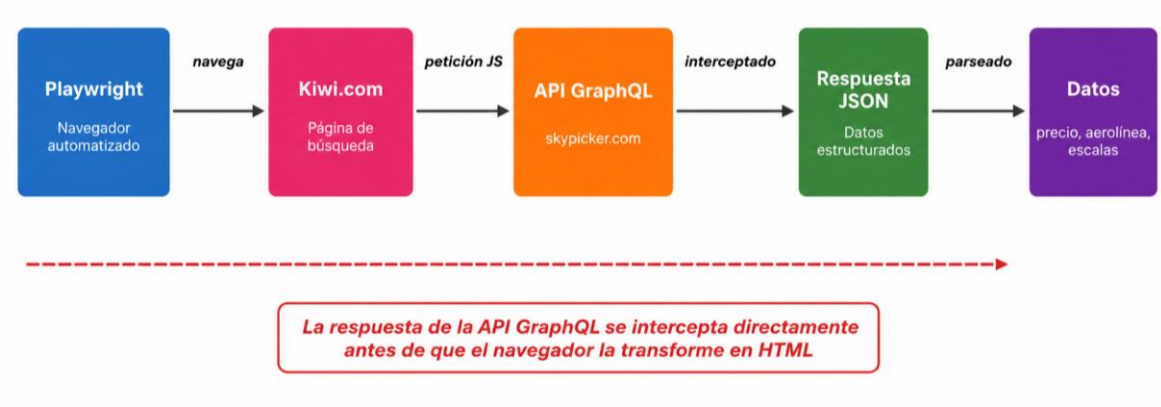

Ilustración 5 Flujo de intercepción de la API de Kiwi.com seguido por el scraper

Por último, ningún análisis del web scraping está completo sin abordar sus dimensiones éticas y legales. La situación legal del scraping de datos públicos ha sido objeto de importantes litigios, sobre todo en Estados Unidos en el caso *hiQ Labs, Inc.* contra *LinkedIn Corp.* (*hiQ Labs, Inc. V. LinkedIn Corp.*, 2019), donde se estableció que la recopilación automatizada de datos de acceso público no constituye una violación de la ley federal contra el Fraude y el Abuso Informático. En la Unión Europea, la situación está más matizada y depende de si hay datos personales de por medio, en cuyo caso se aplica el Reglamento General de Protección de Datos (RGPD) (Vlad Krotov et al., 2020). Más allá de la estricta legalidad, una práctica ética sólida dicta que los rastreadores deben respetar la convención

robots.txt[7], identificarse claramente a través de la cadena User-Agent[8] cuando sea posible y evitar imponer una carga excesiva al servidor de destino. El scraper utilizado en este trabajo introduce pausas estocásticas entre solicitudes consecutivas, extraídas uniformemente del intervalo [2,5; 4,5] segundos, con el fin tanto de mitigar el riesgo de sobrecargar la infraestructura de Kiwi.com como de reducir la probabilidad de ser identificado como tráfico automatizado por los mecanismos antibots. Los datos recopilados se utilizan exclusivamente para investigación académica no comercial y no se redistribuyen.

Una vez establecido el marco de adquisición de datos, el resto del capítulo presenta las técnicas analíticas aplicadas a los conjuntos de datos resultantes, comenzando por los fundamentos del análisis de series temporales.

2.2 FUNDAMENTOS DEL ANÁLISIS DE SERIES TEMPORALES

2.2.1 PROCESOS ESTOCÁSTICOS Y SERIES TEMPORALES

Una serie temporal es la realización observada de un proceso estocástico $\{Y_t\}_{t\in T}$, donde cada Y_t es una variable aleatoria indexada por el tiempo t. En el presente trabajo, t denota días naturales, por lo que el conjunto de índices T es discreto; no obstante, el marco formal se aplica igualmente a los procesos de tiempo continuo que se encuentran, por ejemplo, en el análisis de señales físicas. La característica definitoria de una serie temporal, a diferencia de una muestra multivariante genérica, es que se supone que las observaciones consecutivas son estadísticamente dependientes. Y es precisamente esa dependencia, formada a través del concepto de autocorrelación, la que constituye el objeto principal de la modelización.

[7] Robots.txt, Un archivo de texto que un sitio web puede colocar en su directorio raíz para indicar a qué secciones del sitio pueden o no pueden acceder los agentes automatizados. Definido por el *Protocolo de exclusión de robots* (RFC 9309, 2022), no es jurídicamente vinculante, pero su cumplimiento se considera una buena práctica.

[8] User-Agent, Una cadena que envía un cliente HTTP al servidor en cada solicitud para identificar el software que establece la conexión (navegador, versión, sistema operativo). Su modificación es una práctica habitual en el scraping con el fin de emular diferentes entornos de navegador.

En la literatura sobre series temporales han surgido dos amplias perspectivas analíticas. El enfoque del dominio del tiempo examina la serie observación por observación, caracterizando su estructura de dependencia mediante funciones de autocorrelación y autocovarianza, y construyendo modelos paramétricos como la serie ARIMA que se analiza más adelante en este capítulo. Por otro lado, el enfoque del dominio de la frecuencia descompone la serie en sus frecuencias constituyentes mediante el análisis espectral, aprovechando la transformada de Fourier de la función de autocovarianza (Shumway & Stoffer, 2006). Mientras que este último se emplea considerablemente en aplicaciones geofísicas y de ingeniería, el primero es el estándar en la predicción económica, y por ello, es el enfoque adoptado a lo largo de este estudio.

2.2.2 COMPONENTES DE UNA SERIE TEMPORAL

Una serie temporal se descompone normalmente en cuatro componentes potenciales. La tendencia T_t describe la dirección a largo plazo de la serie, sin tener en cuenta las fluctuaciones a corto plazo. La estacionalidad S_t capta los patrones que se repiten con un período fijo y conocido, como los ciclos semanales o anuales. El componente cíclico C_t da respuesta a las variaciones recurrentes pero aperiódicas, habitualmente asociadas a ciclos económicos cuya duración no está fija en el calendario. Y, por último, el componente irregular ε_t recoge la variación estocástica residual que queda una vez que se han tenido en cuenta los otro tres.

La combinación de estos elementos admite dos formas canónicas. La descomposición aditiva $Y_t = T_t + S_t + C_t + \varepsilon_t$ es adecuada cuando la amplitud de las fluctuaciones estacionales es independiente del nivel de la serie. La descomposición multiplicativa $Y_t = T_t \cdot S_t \cdot C_t \cdot \varepsilon_t$ se aplica cuando la amplitud estacional varía proporcionalmente al nivel de la serie, y puede linealizarse tomando logaritmos, reduciendo así el modelo multiplicativo a uno aditivo en el espacio logarítmico.

Se ha desarrollado varios procedimientos para estimar estos componentes de forma empírica. La descomposición clásica mediante medias móviles sigue siendo la más usada desde el punto de vista educativo, aunque presenta límites y asume un patrón estacional rígido. Entre

las alternativas más sofisticadas se incluyen el software X-13-ARIMA-SEATS[9] desarrollado por la Oficina del Censo de los Estados Unidos, que es el estándar de facto para el ajuste estacional de las estadísticas económicas oficiales (Findley et al., 1998), y la descomposición STL (descomposición de tendencia estacional mediante LOESS[10]) introducida por (CLEVELAND, 1990), que emplea la regresión ponderada localmente para extraer la tendencia y la estacionalidad de una manera flexible y robusta frente a valores atípicos. STL es el procedimiento más fácilmente disponible en los paquetes de recursos modernos de Python y es el que se utiliza en los capítulos posteriores de este trabajo cuando se requiere una descomposición exploratoria.

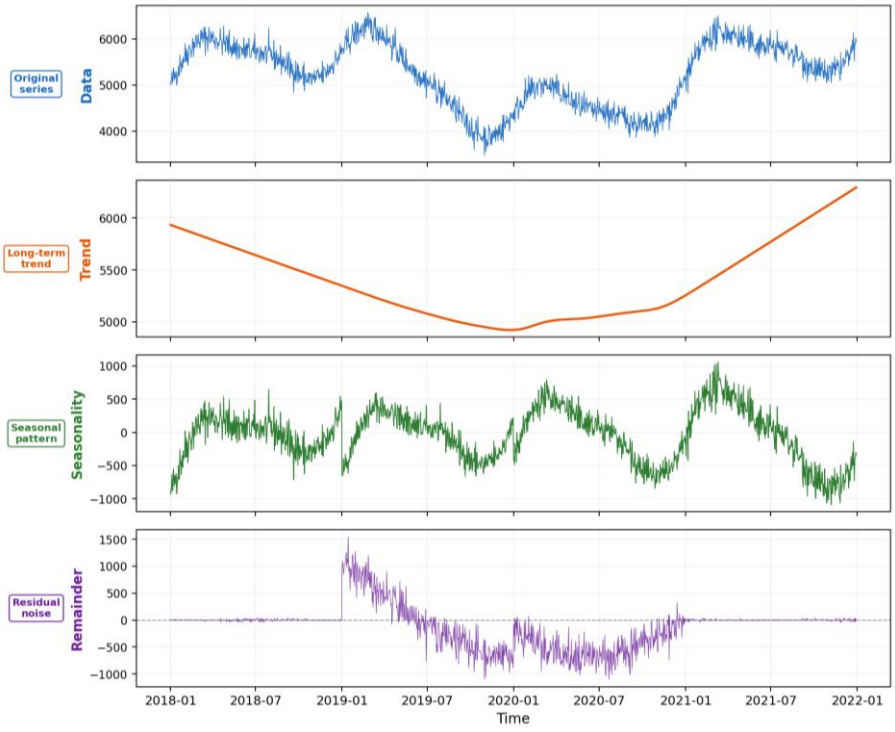

Ilustración 6 Descomposición STL de una serie temporal sintética en sus cuatro componentes. Los datos originales (arriba), la tendencia a largo plazo, el patrón estacional y el ruido residual (abajo). La descomposición ilustra cómo una serie compleja puede dividirse en componentes interpretables.

[9] X-13-ARIMA-SEATS, Un programa de ajuste estacional desarrollado por la Oficina del Censo de los Estados Unidos, ampliamente utilizado en las estadísticas oficiales. Combina el modelo regARIMA para los efectos de valores atípicos y de calendario con la extracción de señales basada en SEATS.

[10] LOESS (locally Estimated Scatterplot Smoothing), Una técnica de regresión local no paramétrica que ajusta polinomios de bajo grado a subconjuntos localizados de los datos, lo que da lugar a una estimación suavizada de la tendencia subyacente. Propuesta originalmente por Cleveland (1979), constituye la función de suavizado fundamental de la descomposición STL..

2.2.3 ESTACIONARIEDAD

El concepto de estacionariedad es fundamental en la teoría clásica de las series temporales, de hecho, sustenta la validez de los modelos ARIMA que se presentan más adelante en este capítulo. En la documentación especializada se suelen distinguir dos definiciones de diferente nivel de exigencia.

Se dice que una serie es estrictamente estacionaria si la distribución conjunta de cualquier subconjunto de observaciones es invariante ante desplazamientos temporales. Es decir, para cualquier h y cualquier índice $t_1, \ldots, t_n$:

$$F(Y_{t_1}, Y_{t_2}, \ldots, Y_{t_n}) = F(Y_{t_1+h}, Y_{t_2+h}, \ldots, Y_{t_n+h})$$

Esta condición es sumamente estricta, ya que exige que todos los momentos de la distribución sean invariantes en el tiempo, y en la práctica es imposible verificarla de forma empírica. Por lo tanto, su valor es principalmente teórico.

Un concepto con más operatividad es el de estacionariedad débil (también denominada estacionariedad de segundo orden o de covarianza) que solo requiere que los dos primeros momentos sean invariantes en el tiempo. Formalmente, una serie es débilmente estacionaria si:

1. $E[Y_t] = \mu$ es constante para todo t
2. $Var[Y_t] = \sigma^2 < \infty$ es constante para todo t
3. $Cov(Y_t, Y_{t+h}) = \gamma_k$ depende únicamente del retraso k, no de t

La estacionariedad estricta implica estacionariedad débil siempre que existan los momentos de segundo orden. Lo contrario es válido para los procesos gaussianos, donde la estacionariedad débil implica estacionariedad estricta en virtud del hecho de que una distribución gaussiana se caracteriza completamente por su media y su covarianza.

La importancia de la estacionariedad para la predicción es triple. En primer lugar, los parámetros de los modelos ARMA y ARIMA se definen bajo el supuesto de estacionariedad,

y sus procedimientos de estimación desarrollan estimaciones inestables cuando se aplica a series no estacionarias. Por otro lado, las distribuciones asintóticas de los estimadores de parámetros (utilizados para construir intervalos de confianza y pruebas de hipótesis) se derivan bajo la hipótesis de la estacionariedad, por lo que su validez depende de esta hipótesis. En último lugar, y de manera más fundamental, la propia lógica de aprender de las observaciones pasadas para pronosticar el futuro presupone que las propiedades estadísticas del pasado son también las del futuro; si el proceso generador de datos cambia con el tiempo, la extrapolación carece de base sólida.

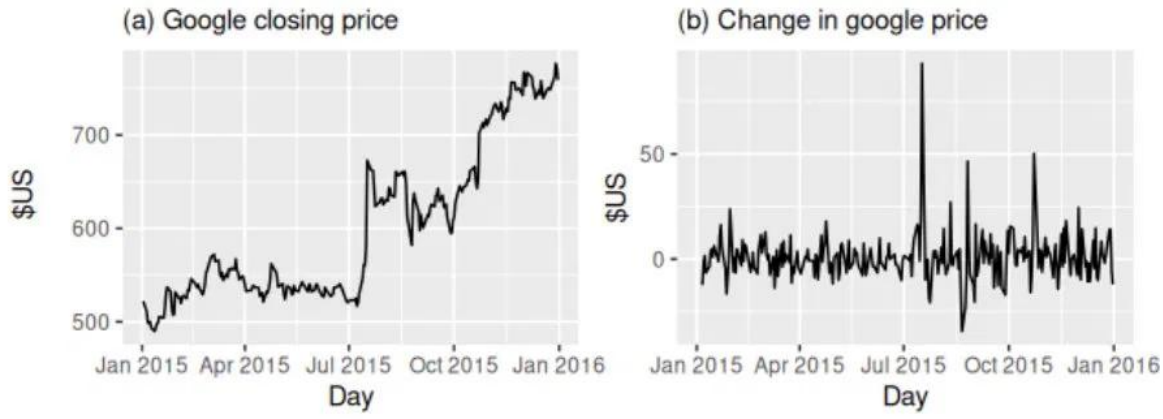

Ilustración 7 Comparación entre una serie estacionaria (izquierda) y una serie no estacionaria con tendencia (derecha). - Fuente: https://otexts.com/

2.2.4 OBTENCIÓN DE LA ESTACIONARIEDAD

En el mundo real, son pocas las series que son estacionarias en sus niveles. Las dos fuentes principales de no estacionariedad son una media no constante (normalmente debido a la tendencia) y una varianza no constante (heterocedasticidad). Ambas pueden abordarse mediante transformaciones estándar.

La diferenciación elimina la tendencia. Introduciendo el operador retardo B[11] definido por $B_{y_t} = y_{t-1}$, la primera diferencia se escribe como:

$$\Delta y_t = y_t - y_{t-1} = (1 - B)y_t$$

Las diferencias de orden superior se obtienen mediante aplicación sucesiva, con $\Delta^d y_t = (1 - B)^d y_t$. Se dice que una serie que se vuelve estacionaria tras d diferencias es integrada de orden d, denotada I(d). Las series con una tendencia lineal suelen ser I(1), que es con diferencia el caso más común en los datos económicos y financieros. Las series son una tendencia cuadrática o más fuerte pueden requerir I(2).

Para series con estacionalidad de período s, la diferencia estacional $\Delta^s y_t = y_t - y_{t-s} = (1 - B)^s y_t$ elimina el componente estacional, y puede combinarse con la diferenciación ordinaria dentro del marco SARIMA que se analiza en la siguiente sección.

Una cuestión que merece la pena destacar es el peligro de la sobrediferenciación. Aplicar más diferencias de las estrictamente necesarias introduce una autocorrelación negativa artificial en el retraso uno o infla la varianza de la serie resultante. Por lo tanto, la práctica recomendada es diferenciar solo lo necesario para lograr la estacionariedad, utilizando las pruebas de raíz unitaria presentadas en la *sección 2.3* como criterio formal de parada.

Cuando la varianza de la serie es en sí misma no constante, la diferenciación por sí sola es insuficiente. La serie de transformaciones de Box-Cox[12] (G. E. Box & Cox, 1964) proporciona un mecanismo paramétrico para la estabilización de la varianza:

[11] Operador de retardo B, Operador definido porque proporciona una notación algebraica compacta para los polinomios de diferencia y AR/MA. En la literatura econométrica, a veces también se denota como *lag*.
[12] Transformaciones Box-Cox, Una familia paramétrica de transformaciones de potencia introducida por Box y Cox (1964) para estabilizar la varianza de una serie y aproximarse a la normalidad. El parámetro se estima normalmente mediante el método de máxima verosimilitud.

$$Y_t^{(\lambda)} = \begin{cases} \dfrac{Y_t^{\lambda} - 1}{\lambda} & si \ \ \lambda \neq 0 \\ \log Y_t & si \ \ \lambda = 0 \end{cases}$$

Donde el parámetro λ se estima típicamente mediante máxima verosimilitud. El caso especial $\lambda = 0$ corresponde a la transformación logarítmica, que es muy habitual en el análisis de series temporales de precios y economías.

2.2.5 AUTOCORRELACIÓN: ACF Y PACF

La estructura de dependencia de una serie temporal estacionaria se resume mediante su función de autocovarianza, definida por el retraso o lag *k* como:

$$\gamma_k = Cov(Y_t, Y_{t-k}) = E[(Y_t - \mu)(Y_{t-k} - \mu)]$$

Normalizando por la varianza se obtiene la función de autocorrelación (ACF):

$$\rho_k = \frac{\gamma_k}{\gamma_0} \ \epsilon \ [-1, 1]$$

Que mide la correlación lineal entre la serie y su propio pasado en el lag *k*. En la práctica, la ACF se estima a partir de datos muestrales como:

$$\hat{\rho}_k = \frac{\sum_{t=k+1}^{T}(y_t - \overline{y})(y_{t-k} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2}$$

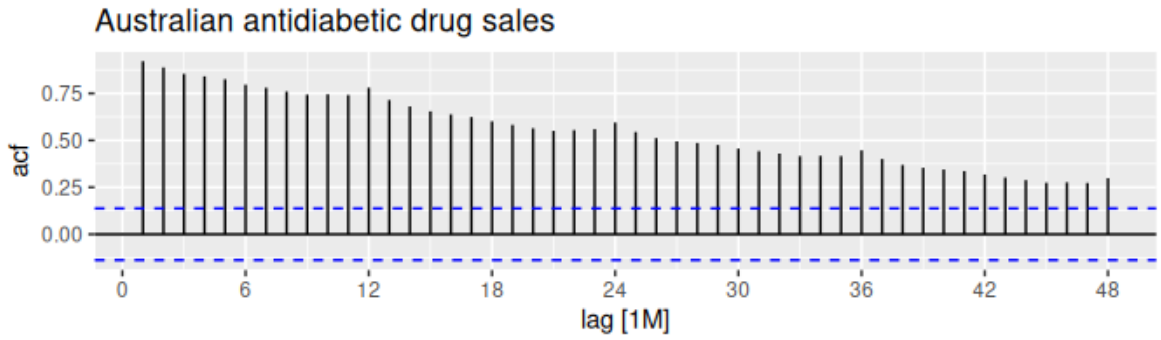

Ilustración 8 ACF de las ventas de medicamentos para diabéticos en Australia - fuente: https://otexts.com/

Se visualiza como un correlograma, un gráfico de barras que muestra $\hat{\rho}_k$ en función de k, complementado con bandas de confianza en +-1.96/$\sqrt{T}$ que indican la región en la que caerían las autocorrelaciones de un proceso de ruido blanco[13] con una probabilidad del 95 %. Las barras que se extienden más allá de estas bandas se interpretan como autocorrelaciones estadísticamente significativas.

La función de autocorrelación parcial (PACF) complementa a la ACF midiendo la correlación entre Y_t y Y_{t-k} una vez eliminada la influencia de las observaciones intermedias $Y_{t-1}, \ldots, Y_{t-k+1}$. Formalmente, la autocorrelación parcial de orden k es el coeficiente ϕ_{kk} en la regresión:

$$Y_t = \phi_{k0} + \phi_{k1} Y_{t-1} + \ldots + \phi_{kk} Y_{t-k} + \varepsilon_t$$

El comportamiento conjunto de la ACF y la PACF proporciona el criterio clásico de Box-Jenkins (G. E. P. Box et al., 2015) para identificar el orden de un modelo ARMA, resumido en la siguiente tabla:

Modelo	Comportamiento de la ACF	Comportamiento de la PACF
AR(p)	Decae gradualmente	Se corta tras el lag *p*
MA(q)	Se corta tras el lag *q*	Decae gradualmente
ARMA(p,q)	Decae gradualmente	Decae gradualmente

Tabla 1 Criterios para la identificación de un modelo ARMA.

Esta heurística de identificación se aplica sistemáticamente en el capítulo 6 de este trabajo para determinar los órdenes adecuados para los modelos ARIMA ajustados a la serie de precios de los vuelos.

[13] Ruido blanco, Un proceso estocástico estacionario con media nula, varianza constante y autocorrelación nula en todos los retardos distintos de cero. El término proviene de la física, donde se refiere a una señal con una densidad espectral plana en todas las frecuencias.

2.2.6 RUIDO BLANCO Y EL PASEO ALEATORIO

El proceso estacionario más simple es el ruido blanco, definido como una sucesión $\{\varepsilon_t\}$de variables aleatorias con media nula, varianza finita constante σ^2 y autocorrelación nula en todos los retrasos distintos de cero:

$E[\varepsilon_t]= 0$, $Var[\varepsilon_t]= \sigma^2$, $Cov(\varepsilon_t, \varepsilon_s)= 0$ para $t \neq s$

Cuando además se supone que las variables siguen una distribución gaussiana, el proceso se denomina ruido blanco gaussiano. La relevancia del ruido blanco va mucho más allá de su papel como referencia teórica: constituye la estructura objetivo para los residuos de un modelo de series temporales correctamente especificado. Si un modelo ha captado toda la dependencia sistemática presente en los datos, los residuos resultantes deberían ser indistinguibles del ruido blanco. Por contra, cualquier estructura restante en los residuos indica que el modelo puede mejorarse. Este principio motiva las pruebas de diagnóstico examinadas en la *sección 2.3*.

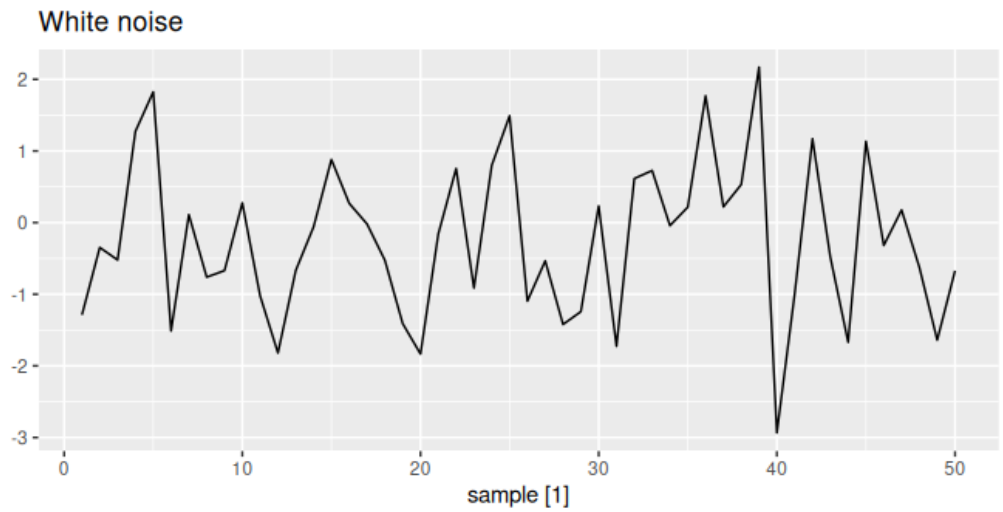

Ilustración 9 Una serie temporal con ruido blanco - Fuente: https://otexts.com/

Un proceso muy relacionado y con importancia en aplicaciones económicas y financieras es el paseo aleatorio, definido como:

$$Y_t = Y_{t-1} + \varepsilon_t$$

Donde $\{\varepsilon_t\}$ es ruido blanco. El paseo aleatorio es el ejemplo de un proceso I(1) no estacionario: su media y varianza crecen con el tiempo, pero su primera diferencia $\Delta Y_t = \varepsilon_t$ es estacionaria por definición. En economía financiera, la hipótesis del paseo aleatorio (*International Journal of Forecasting*, 1995) postula que los precios de los activos siguen un proceso de este tipo, con la implicación inmediata de que la mejor previsión del precio de mañana es el precio de hoy. La relevancia empírica de esta hipótesis para los precios de los vuelos se examina en el capítulo de resultados de este trabajo. Una variante que conserva interés teórico es el paso aleatorio con deriva $Y_t = c + Y_{t-1} + \varepsilon_t$, cuya primera diferencia es ruido blanco centrado en la constante c.

2.2.7 APLICACIÓN A LAS SERIES DE PRECIOS DE VUELOS

Los conceptos introducidos en esta sección se aplican directamente a las series de precios de vuelos analizadas en los capítulos siguientes de este trabajo. Las series objeto de estudio consisten en observaciones diarias de precios a lo largo de una ventana de aproximadamente cincuenta días, derivadas del procedimiento de web scraping descrito en el *capítulo 3*. Una inspección preliminar de estas series revela un comportamiento heterogéneo entre las rutas: algunas como MAD-LHR, muestran una clara tendencia al alza a medida que se acerca la fecha de salida, lo que sugiere no estacionariedad en los niveles y justifica la diferenciación de primer orden; otras como MAD-BCN, oscilan en torno a una media estable y parecen más cercanas a la estacionariedad débil en los niveles. La varianza se mantiene relativamente constante a lo largo del intervalo de observación en todas las rutas, lo que sugiere que no es necesaria una transformación de Box-Cox antes de modelización. La verificación formal de estas impresiones visuales, mediante pruebas de raíz unitaria que se presentan en la siguiente sección, y la posterior identificación de los órdenes ARIMA adecuados mediante inspección de la ACF y PACF, se desarrollan en el *capítulo 6*.

2.3 TESTS ESTADÍSTICOS PARA SERIES TEMPORALES

2.3.1 PRUEBAS DE HIPÓTESIS Y EL P-VALOR

Los conceptos introducidos en la *sección 2.2* (estacionariedad, ruido blanco y falta de autocorrelación) pueden evaluarse de manera aproximada mediante la inspección visual de una serie o de su correlograma. Por lo contrario, dichos análisis visuales son necesariamente subjetivos y no resultan óptimos como base para conclusiones científicas reproducibles. Los tests de hipótesis estadísticas proporcionan el mecanismo formalmente necesario para convertir estas percepciones visuales en decisiones objetivas y cuantificables.

Cada prueba estadística sigue la misma estructura lógica. Con una hipótesis nula H0 codifica la suposición por defecto, a la que se contrasta una hipótesis alternativa H1. A partir de los datos se calcula una estadística de prueba, de la que se concluye el p-valor[14], definido como la probabilidad de observar datos al menos en extremos como los obtenidos en la hipótesis H0 es verdadera. Si el p-valor está por debajo de un nivel de significación predeterminado (normalmente $\alpha = 0.05$), se rechaza la hipótesis nula, sino, no se rechaza. Cabe destacar que no rechazar H0 no genera una prueba de que H0 sea verdadera, sino que indica que la evidencia disponible es insuficiente para desmentirla. Esta asimetría es precisamente importante en lo que sigue, debido a que los cuatro test utilizados en el trabajo colocan la propiedad de interés (estacionariedad, independencia o normalidad) en extremos diferentes de la hipótesis nula.

2.3.2 PRUEBAS DE RAÍZ UNITARIA: ADF Y KPSS

Una raíz unitaria es la característica principal de no estacionariedad. Fijándonos en el paseo aleatorio $Y_t = Y_{t-1} + \varepsilon_t$, el coeficiente que multiplica a Y_{t-1} es justamente la unidad. Dicha unidad es la raíz unitaria, y su existencia implica que las variaciones en la serie tienen

[14] P-valor, La probabilidad de obtener una estadística de prueba al menos tan extrema como la observada, bajo el supuesto de que la hipótesis nula es cierta. No se trata de la probabilidad de que la hipótesis nula sea cierta, como se suele interpretar erróneamente.

un efecto permanente en lugar de temporal, lo que provoca qué la serie sea no estacionaria. Por tanto, comprobar la presencia de una raíz unitaria permite comprobar este tipo específico de no estacionariedad.

La prueba de Dickey-Fuller aumentada (ADF)[15] (Dickey & Fuller, 1979) aborda este dilema directamente. Sus hipótesis nula y alternativa son:

- H0: la serie contiene una raíz unitaria (no estacionaria).
- H1: la serie no contiene una raíz unitaria (estacionaria).

Por ello, un p-valor inferior al umbral de significación indica rechazo de H0 y confirma la conclusión de que la serie es estacionaria. El término "aumentada" se refiere a la incorporación de términos de diferencia con retardo en la regresión de la prueba, que corrigen la autocorrelación de orden superior en los residuos de la especificación de Dickey-Fuller. En la aplicación utilizada en este trabajo, el número de retardos de aumento se seleccionan automáticamente minimizando así, el Criterio de información de Akaike.

La prueba KPSS (Shin, 1992) examina la misma propiedad, pero con las hipótesis invertidas:

- H0: la serie es estacionaria (en torno a un nivel o tendencia).
- H1: la serie es no estacionaria (contiene una raíz unitaria).

Para este caso, un p-valor inferior al umbral deriva al rechazo de la estacionariedad. El hecho de invertir la hipótesis con respecto a ADF, es lo que provoca que ambas pruebas sean complementarias. Debido a que ninguna prueba de raíz unitaria por sí sola es totalmente eficaz (es sabido que la prueba ADF tiene baja potencia frente a ciertas alternativas), el uso conjunto de ADF y KPSS, a menudo denota un análisis confirmatorio, ofreciendo una

[15] Dickey-Fuller aumentada, Una extensión de la prueba de Dickey-Fuller que incluye términos de diferencia con retardo en la regresión de la prueba para tener en cuenta la correlación serial de orden superior, propuesta originalmente por Dickey y Fuller (1979).

evaluación más sólida. A continuación, se resumen las cuatro combinaciones posibles de resultados:

Resultado ADF	Resultado KPSS	Conclusión
Estacionario	Estacionario	Estacionario (fuerte concordancia)
No estacionario	No estacionario	No estacionario (fuerte concordancia)
Estacionario	No estacionario	No concluyente (se requiere un análisis más detallado)
No estacionario	Estacionario	No concluyente (posiblemente estacionario en diferencias)

Tabla 2 Resumen de las cuatro combinaciones posibles de resultados.

En el presente trabajo el uso combinado de ambas pruebas es uno de los ajustes metodológicos adoptados, y los casos no concluyentes que surgen en algunas rutas se tratan en el capítulo de resultados.

2.3.3 DIAGNÓSTICO DE LOS RESIDUOS: LA PRUEBA DE LJUNG-BOX

Mientras que las pruebas de raíz unitaria se aplican a las series antes de modelizar, la prueba de Ljung-Box (LJUNG & BOX, 1978) se aplica a los residuos de un modelo ajustado con la finalidad de evaluar si queda alguna relación estadística pendiente por detectar. Sus hipótesis son:

- H0: los residuos se distribuyen de forma independiente (sin autocorrelación)
- H1: los residuos presentan autocorrelación en uno o más retrasos.

Por ello, un p-valor por debajo del umbral indica la aparición de autocorrelación residual, lo que indicaría que el modelo no ha logrado capturar toda la estructura de los datos. En vez de examinar un único retraso aislado, esta prueba añade las autocorrelaciones al cuadrado de los primeros *h* retrasos en una única estadística, con lo que responde así a la pregunta de si existe una autocorrelación significativa en cualquiera de esos retrasos de forma conjunta. En el presente trabajo, la prueba se toma con retrasos acordes a la longitud de la serie. Además, se considera que un modelo cuyos residuos satisfacen la prueba Ljung-Box- es decir, que

son idénticos al ruido blanco- está correctamente definido en lo que respecta a l estructura de autocorrelación.

2.3.4 PRUEBA DE NORMALIDAD: LA PRUEBA DE SHAPIRO-WILK

Este diagnóstico final tata la forma de distribución de los residuos. La prueba de Shapiro-Wilk (SHAPIRO & WILK, 1965) evalúa si una muestra es extraída de una distribución normal, con las hipótesis:

- H0: los residuos se distribuyen normalmente.
- H1: los residuos no se distribuyen normalmente.

La probabilidad estadística de prueba W oscila entre cero y uno, siendo los valores cercanos a la unidad una mayor coincidencia con la normalidad. Como en las otras pruebas, en este caso el p-valor por debajo del umbral indica rechazo de la hipótesis de normalidad.

La búsqueda de la normalidad de los residuos es una de las características principales deseable poque los intervalos de confianza asociados a las previsiones ARIMA se construyen bajo el supuesto de novedades gaussianas. En cambio, si se incumple este supuesto, las estimaciones de intervalo suelen ser menos fiables, aunque las previsiones puntuales en sí siguen siendo válidas. Por tanto, cabe destacar que el modelo sigue siendo válido, aunque no se haya logrado esta prueba, de hecho, matiza la interpretación de sus previsiones por intervalos. La prueba de Shapiro-Wilk se considera generalmente la prueba de normalidad más sofisticado para muestras pequeñas. Como pruebas alternativas, destaca la prueba de Jaque-Bera (Jarque & Bera, 1980), muy utilizada en econometría y basada en la asimetría y la curvatura de la muestra, de igual modo, las pruebas de Kolmogorov-Smirnov y Anderson-Darling.

2.3.5 RESUMEN

Las cuatro pruebas descritas en esta sección componente el conjunto de herramientas de diagnóstico utilizado a lo largo de los capítulos destinados a la modelización de este trabajo. Sus funciones son complementarias y se resumen en la siguiente tabla:

Prueba	Propiedad evaluada	H0	Aplicada a
ADF	Estacionariedad	Raíz unitaria (no estacionaria)	Serie
KPSS	Estacionariedad	Estacionaria	Serie
Ljung-Box	Autocorrelación	Independencia (ruido blanco)	Residuos
Shapiro-Wilk	Normalidad	Distribución normal	Residuos

Tabla 3 Resumen de las cuatro pruebas diferentes.

Las pruebas de raíz unitaria determinan si es necesario diferenciar la serie y de qué forma antes de modelizar, mientras que el análisis de los residuos comprueba, tras la estimación, que el modelo ajustado ha captado correctamente la estructura de dependencia de los datos. Una vez constituido este diagnóstico, la siguiente sección trata la familia de modelos ARIMA a ala que se aplicaran dichas pruebas.

2.4 MODELO ARIMA

El modelo ARIMA (acrónimo de *AutoRegressive Integrated Moving Average*) es una de las familias de modelos más utilizadas para la predicción de series temporales univariantes. Desarrollado en su versión moderna por Box y Jenkins en su importante libro de 1970 (G. Box & Jenkins, 1976), el modelo descompone la dependencia temporal de una serie en tres componentes: una parte autorregresiva (AR), una parte de integración (I) y una parte de media móvil (MA). En esta sección se estudian cada uno de los componentes por separado, antes de describir la metodología mediante la cual se identifica, estima y valida el modelo en la práctica.

2.4.1 EL MODELO AUTORREGRESIVO AR

El modelo autorregresivo de orden p, denotado como AR(p), plantea que el valor actual de una serie puede expresarse como una combinación lineal de sus p-valores pasados más recientes, más una alteración estocástica:

$$y_t = c + \emptyset_1 y_{t-1} + \emptyset_2 y_{t-2} + \ldots + \emptyset_p y_{t-p} + \varepsilon_t$$

Donde c es una constante, $\emptyset_1, \ldots, \emptyset_p$ son los coeficientes autorregresivos y ε_t es un término de ruido blanco. De forma intuitiva, un modelo AR(p) recoge la "inercia" de una serie: el grado en que el valor actual viene determinado por los valores pasados más recientes. La especificación que no es insignificante más simple , AR(1), supone que $y_t = c + \emptyset_1 y_{t-1} + \varepsilon_t$, luego la dependencia de la observación actual con respecto a su pasado se agrupa en un único coeficiente. El funcionamiento del proceso depende notablemente del valor de ese coeficiente: cuando $|\emptyset_1| < 1$, las perturbaciones caen exponencialmente y el proceso es estacionario. En cambio, cuando $|\emptyset_1| = 1$, las perturbaciones tienen un comportamiento permanente y el proceso se reduce a un paseo aleatorio, el ejemplo clásico de no estacionariedad.

De forma más genérica, un procesamiento AR(p) es estacionario si y solo si las raíces de su polinomio característico $1 - \emptyset_1 z - \ldots - \emptyset_p z^p = 0$ se hallan fuera de la circunferencia unidad. Dicha condición es comprobada automáticamente por el software moderno durante el ajuste del modelo.

2.4.2 EL MODELO DE MEDIA MÓVIL MA

El modelo de media móvil de orden q, representado como MA(q), indica el valor actual de la serie como una combinatoria lineal de los q términos de error más recientes:

$$y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}$$

Donde μ denota la media incondicional y $\theta_1, \ldots, \theta_q$ son los coeficientes de la media móvil. A pesar de la terminología, el modelo MA(q) no tiene relación directa con las medias móviles de ventana deslizante utilizadas para el nivelado. En este modelo, el término se refiere a una combinación ponderada de innovaciones pasadas, es decir, de las fluctuaciones aleatorias que han influido al conjunto en períodos recientes. Si bien un modelo AR(p) capturas dinámicas de refuerzo persistentes a sí mismas, un modelo MA(q) capta efectos temporales:

perturbaciones que tienen una influencia finita y de duración corta de la serie antes de desaparecer.

Cabe destacar, que un proceso MA(q) es siempre estacionario, pero se puede considerar invertible, es decir, se puede representar como un proceso AR pero con orden finito cuando las raíces de su polinomio se hallan fuera del círculo unitario. Esta inversión, es una propiedad de la estacionariedad compartida con AR, y garantiza la singularidad de la representación.

A continuación, se presentan ejemplos de MA(q), a la izquierda: MA(1) con $y_t = 20 + \varepsilon_t + 0{,}8\varepsilon_{(t-1)}$. A la derecha: MA(2) con$y_t = \varepsilon_t - \varepsilon_{(t-1)} + 0{,}8\varepsilon_{(t-2)}$. En ambos casos, ε_t es ruido blanco con distribución normal, con media cero y varianza uno:

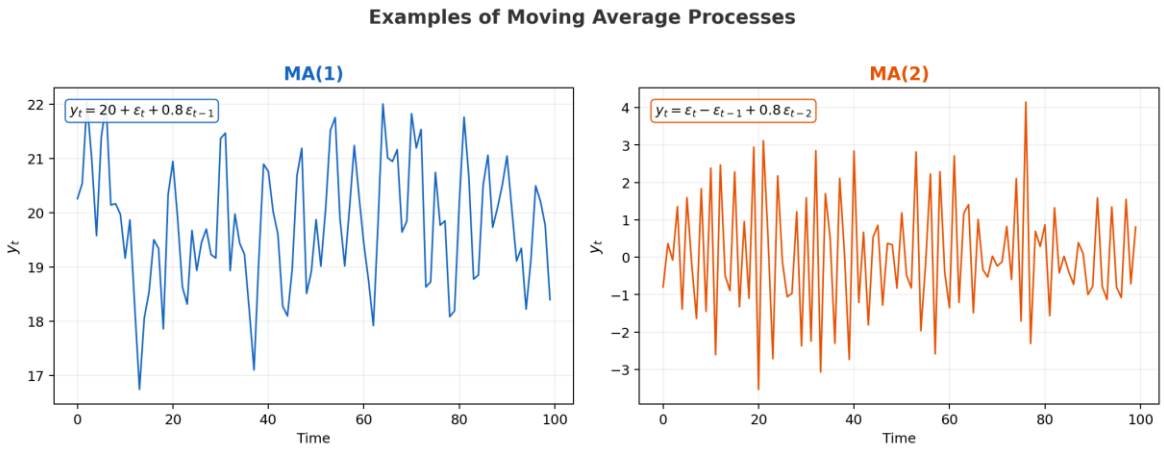

Ilustración 10 Dos ejemplos de procesos de media móvil con diferentes parámetros.

2.4.3 EL MODELO ARMA

La combinación de los elementos autorregresivos y de media móvil da como resultado al modelo ARMA(p,q), definido como:

$$y_t = c + \sum_{i=1}^{p} \emptyset_i y_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

Los modelos ARMA dan respuesta a series que presentan al mismo tiempo una dependencia temporal y persistente, y son el pilar sobre el elemento básico del que desprenden la familia ARIMA. Sin embargo, un requisito importante es que los modelos ARMA solo son capaces de aplicarse a series estacionarias. Cuando la serie constituye tendencias u otros ingredientes de no estacionariedad, el enfoque ARMA debe incluir diferenciación, y de ahí nace la especificación ARIMA.

2.4.4 EL MODELO ARIMA

El modelo ARIMA(p,d,q) generaliza el enfoque ARMA a series no estacionarias mediante el uso de un paso de "integración". Específicamente, la serie original se diferencia *d* veces hasta lograr la estacionariedad, y posteriormente se puede ajustar un modelo ARMA(p,q) a la nueva serie ya diferenciada. El número entero *d* es el orden de integración, *p* es el orden del componente autorregresivo y *q* el orden del componente de media móvil. Empleando el operador de retardo *B*, el modelo puede plantearse de forma compacta como:

$$\emptyset(B)(1-B)^d y_t = c + \theta(B)\varepsilon_t$$

Donde $\emptyset(B) = 1 - \emptyset_1 B - \ldots - \emptyset_p B^p$ y $\theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q$ son los polinomios AR y MA, respectivamente.

Es bien sabido que varios de los procesos conocidos nacen como casos concretos de la familia ARIMA. Un ARIMA(0,0,0) es una serie de ruido blanco; un ARIMA(0,1,0) es un paseo aleatorio; un ARIMA(p,0,0) es un proceso AR(p) aplicable a una serie estacionaria. Por todo ello, esta versatilidad del modelo ARIMA (la capacidad de incluir dinámicas tan diversas bajo una misma especificación) es una de las principales razones de su eterna popularidad en la predicción aplicada.

2.4.5 LA METODOLOGÍA DE BOX-JENKINS

La utilización práctica de los modelos ARIMA se estructura normalmente según la metodología en tres etapas propuesta por Box y Jenkins(1976). La primera de las etapas, la identificación trata de determinar los valores correctos de p ,d y q. El orden de integración d se obtiene mediante la aplicación continuada de pruebas de raíz unitaria, como ADF y KPSS, pudiendo diferenciar la serie hasta que se establece la estacionariedad. Los órdenes p y q se obtienen mediante inspección visual de las funciones de autocorrelación y autocorrelación parcial de la serie estacionaria, según el esquema clásico resumido en la *sección 2.2*: una ACF que decae progresivamente junto con una PACF que se interrumpe bruscamente sugiere un proceso AR, a su vez el patrón inverso indica un proceso MA, y la caída gradual en ambas funciones apunta a una especificación ARMA mixta.

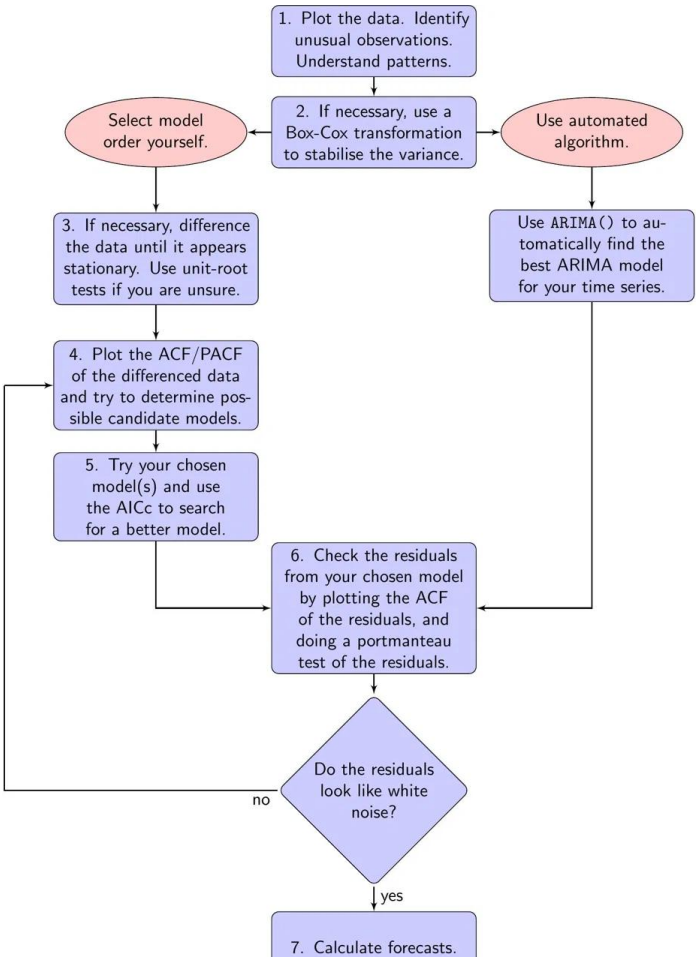

Ilustración 11 Procedimiento general para el modelado ARIMA, que muestra tanto la ruta de identificación manual (izquierda) como la ruta del algoritmo automatizado (derecha), con un bucle de validación basado en el diagnóstico de residuos. Adaptado de Hyndman y Athanasopoulos (2021).

La segunda etapa, la estimación, trata sobre ajustar los coeficientes de la especificación seleccionada anteriormente mediante máxima

verosimilitud[16]. En el marco de las innovaciones gaussianas, la función de verosimilitud permite una expresión en forma cerrada que puede maximizarse numéricamente. Además, aplicaciones modernas como `statsmodels` y `pmdarima` realizan dicho paso de forma automática con muy bajo coste computacional para un tamaño de serie considerable.

La tercera y última etapa, la validación, consiste en verificar que el modelo ajustado ha captado correctamente la estructura de los datos. Esto es posible gracias a la prueba de Ljung-Box aplicada a los residuos, que deberían ser idénticos del ruido blanco si el modelo está correctamente estructurado y de forma opcional es interesante la comprobación de la normalidad de los residuos mediante la prueba de Shapiro-Wilk. Si esta última fracasa, no quiere decir que el modelo sea inválido, pero matiza la fiabilidad de los intervalos de predicción asociados.

2.4.6 CRITERIOS DE INFORMACIÓN: AIC Y BIC

Es preciso su utilización en situaciones en las que la inspección de la ACF y la PACF no proporcionen una especificación clara, o en las que es necesario comparar varios modelos sobre una base en común. Los criterios de información proporcionan un método automático para la selección de modelos. Cabe destacar dos de estos criterios que son ampliamente utilizados:

El criterio de información de Akaike (Akaike, 1974), definido como:

$$AIC = 2k - 2\ln(L)$$

Donde K indica el número de parámetros estimados y L indica el valor maximizado de la función de verosimilitud. El AIC penaliza tanto el mal ajuste (verosimilitud baja) como la

[16] Estimación de la máxima verosimilitud (MLE), Método estadístico que estima los parámetros maximizando la probabilidad de observar los datos reales en el marco del modelo supuesto. En el caso de los modelos ARMA gaussianos, el MLE proporciona estimadores asintóticamente eficientes.

complejidad excesiva (*k* grande), dando preferencia a los modelos más sencillos que de todas formas explican bien los datos. Lo valores más bajos de AIC indican los modelos preferidos.

El criterio de información Bayesiano (Schwarz, 1978) presenta una estructura similar pero penalizando la complejidad de forma más agresiva:

$$BIC = k\ln(n) - 2\ln(L)$$

Donde *n* es el tamaño de la muestra. Como se puede apreciar en la ecuación, el término de penalización aumenta con el logaritmo del tamaño de la muestra, el BIC suele seleccionar modelos más parsimoniosos o simples que el AIC, concretamente para muestras grandes. De manera general, cuando el objetivo central es la precisión de la predicción se de cantará por el criterio AIC. En cambio, es preferible BIC cuando el objetivo trata de identificar el proceso generador de datos más razonable. En el presente trabajo se emplea AIC, en línea con la orientación predictiva del ejercicio de modelización.

2.4.7 SELECCIÓN AUTOMÁTICA DEL ORDEN: AUTO_ARIMA

La aplicación manual de la metodología de Box-Jenkins, aunque desde el punto de vista pedagógico es clara, resulta laboriosa cuando se aplica a múltiples series. La función `auto_arima` de la biblioteca `pmdarima`, basada en el algoritmo por pasos de Hyndman y Khandakar (Hyndman & Khandakar, 2008), automatiza el paso de identificación explorando un rango de órdenes establecido y seleccionando el modelo con el valor más bajo del criterio de información como hemos detallado anteriormente. EL algoritmo se inicia con un pequeño conjunto de modelos iniciales, evalúa los cambios locales en *p* y *q*, y repite hasta que no es posible ninguna mejora más.

En este trabajo, la función `auto_arima` se configura con los límites de búsqueda p ϵ [0,5], q ϵ [0,2], d $\leq$ 2, sin componentes adicionales y el AIC como criterio de selección. Estos límites muestran el tamaño limitado de la muestra de las series (normalmente entre 50 y 200 observaciones diarias por ruta) y la ausencia de un ciclo estacional evidente dentro de la ventana temporal de los datos. Como complemento a esta selección automatizada, se emplea

una inspección posterior de los diagnósticos de los residuos, permitiendo garantizar que el modelo elegido cumple la propiedad de ruido blanco para su validez.

2.4.8 PREVISIÓN CON ARIMA

Una vez estimado un modelo, se pueden generar dos tipos de predicción. Las previsiones *in-sample* reproducen las observaciones de entrenamiento, dando lugar a una medida de lo bien que el modelo se ajusta a los datos que ya ha visto previamente. Si bien son útiles en propósito de diagnóstico, no son capaces de interpretarse como un aprueba de la capacidad predictiva, ya que el modelo se ha optimizado a raíz de las mismas observaciones. Por otro lado, las observaciones *out-of-sample*, son capaces de predecir observaciones que el modelo no ha visto durante la estimación y generan la prueba definitiva del rendimiento de la previsión.

Una característica principal de las previsiones ARIMA es su convergencia hacia la media incondicional de la serie a medida que aumenta el horizonte de previsión. Este fenómeno sucede debido a la estacionariedad de la serie diferenciada. La ausencia de una gran base de información imposibilita el poder predecir desviaciones de la media a largo plazo, y la incertidumbre de la previsión (capturada por la amplitud de los intervalos de confianza) aumenta con el horizonte. Por todo lo anterior, las previsiones a largo plazo van perdiendo progresivamente su valor informativo, una limitación común de los métodos de previsión univariantes.

Cabe destacar que los intervalos de confianza de las previsiones ARIMA se calculan bajo el supuesto de innovaciones gaussianas. Cuando los residuos se alejan de la normalidad, las previsiones puntuales siguen siendo válidas, pero ya no se garantiza la cobertura nominal de los intervalos.

2.4.9 AMPLIACIÓN ESTACIONAL: SARIMA

Cuando la serie presenta un patrón estacional regular de periodo conocido s, el enfoque ARIMA puede expandirse al modelo ARIMA estacional o SARIMA, definido como SARIMA(p,d,q)$(P, D, Q)_s$. El trío estacional (P,D,Q) es equivalente al no estacional donde

P y Q controlan los componentes autorregresivo estacional y de media móvil, D es el orden de diferenciación estacional $\Delta_s y_t = y_t - y_{t-s}$, y s es el período estacional.

En el trabajo que nos ocupa no se emplean modelos SARIMA. La franja temporal de los datos (aproximadamente cincuenta observaciones diarias por ruta) nos facilita muy pocos ciclos semanales (s=7 daría solo siete ciclos) para estimar los componentes estacionales con fiabilidad estadística. Por ello, la elección del ARIMA no estacional es una decisión metodológica impulsada por consideraciones de tamaño de la muestra, y no por la ausencia de posibles patrones semanales, que se tratan en su lugar mediante análisis descriptivo en el capítulo de resultados.

2.4.10 LIMITACIONES DE LOS MODELOS ARIMA

Pese a su flexibilidad y elegancia teórica, los modelos ARIMA presentan varias limitaciones que llevan a considerar enfoques alternativos. En primer lugar, se asume que la relación entre valores pasados y presentes es lineal, no pudiendo capturar las dinámicas no lineales (muy común en los sistemas complejos). En segundo lugar, ARIMA es un método univariante, que explota solamente el historial de la propia serie e ignora variables exógenas con gran información. Extensiones como ARIMAX[17] solucionan esta limitación introduciendo regresores externos. En tercer lugar, los saltos en la estructura o los cambios de régimen se adaptan de forma imperfecta, debido a que los parámetros del modelo son constantes a lo largo del tiempo. Por último, como se ha destacado antes, las previsiones tienden hacia la media a largo plazo en horizontes largos, lo que indica la ausencia de información más allá de la contenida en el pasado reciente.

Estas limitaciones impulsan la exploración, en los siguientes capítulos de este trabajo, de métodos de previsión alternativos. Sobre todo, redes neuronales recurrentes del tipo LSTM,

[17] ARIMAX, Una extensión del modelo ARIMA que incorpora variables explicativas exógenas, lo que permite incluir variables externas (efectos de calendario, acontecimientos, series relacionadas) en el modelo de previsión.

capaces de captar dependencias no lineales e incorporar fuentes de información adicionales. La *sección 2.6* presenta el marco teórico de estos modelos.

2.5 MÉTRICAS DE ERROR Y EVALUACIÓN DEL MODELO

2.5.1 LA NECESIDAD DE UTILIZAR MÚLTIPLES MÉTRICAS

Una vez ajustado un modelo de predicción, es recomendable cuantificar su rendimiento comparando sus predicciones $\hat{y}_t$ con los valores observados y_t en un conjunto de observaciones no usadas para la estimación. No obstante, no hay una única métrica de error que sea óptima universalmente válida: cada medición hace énfasis en un aspecto diferente de la precisión de la predicción y tiene sus limitaciones. Por ello, habitualmente es preciso hacer uso de varias métricas que se complementen en lugar de basarse en una sola de manera aislada. En esta sección se detallan las cuatro métricas empleadas a lo largo de este trabajo, así como el punto de referencia *naive* con respecto al cual se interpretan.

2.5.2 MÉTRICAS DEPENDIENTES DE LA ESCALA: MAE Y RMSE

El error absoluto medio (MAE) es la media de las diferencias absolutas entre valores observados y los valores previstos:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|$$

Su ventaja principal es la interpretabilidad: al expresarse en las mismas unidades que los datos, permite la lectura directa de "el error medio en euros". Debido a que no eleva al cuadrado los errores, el MAE es bastante sólido frente a los valores atípicos, ya que trata todas las desviaciones en función de su tamaño.

En cambio, el error cuadrático medio (RMSE) eleva al cuadrado los errores antes de promediarlos y calcular la raíz cuadrada:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$$

La elevación al cuadrado hace que los errores grandes influyan de manera desigual al valor final, por lo que el RMSE penaliza las desviaciones grandes con mayor fuerza que el MAE.

Por tanto, el RMSE es más sensible a los valores atípicos, pero resulta particularmente revelador cuando los errores grandes son indeseables. Ambas métricas comparten las mismas unidades que la serie original, pero ninguna de ellas permite una comparación objetiva entre series de escala distinta. Por su definición, el RMSE es siempre mayor o igual que el MAE, una gran diferencia entre ambos indica que la existencia de unos pocos errores de predicción excesivamente grandes.

2.5.3 MÉTRICAS PORCENTUALES: MAPE

El error absoluto medio (MAPE) expresa el error como un porcentaje del valor absoluto:

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right|$$

Su principal ventaja es la independencia de la escala: un MAPE del cuatro por ciento tienen el mismo significado independiente de si los precios base son del orden de setenta euros como de seiscientos euros, lo que provoca que el MAPE sea particularmente adecuado para comparar el rendimiento de las previsiones entre rutas con niveles de precios diferentes. Esta propiedad en concreto es relevante en este dicho trabajo, donde las rutas objeto de estudio cubren una amplia gama de magnitudes de tarifas. En cambio, el MAPE no está libre de inconvenientes: se vuelve inestable cuando los valores observados se acercan a cero y penaliza la sobreestimación y la subestimación. La primera de la limitación es intrascendente, ya que en el presente trabajo los precios de los vuelos están muy por encima de cero.

2.5.4 MÉTRICAS ESCALADAS: MASE

El error medio escalado (MASE), propuesto por Hyndman y Koehler (Hyndman & Koehler, 2006), mide la precisión de la previsión en relación con la de un punto de referencia sencillo. Se define como la relación entre el error absoluto medio del modelo y el error absoluto medio de una previsión ingenua de un paso calculada sobre el conjunto de entrenamiento:

$$MASE = \frac{MAE_{modelo}}{\frac{1}{m-1}\sum_{t=2}^{m}|y_t - y_{t-1}|}$$

Donde el denominador es el MAE calculado de la previsión ingenua sobre las m observaciones del entrenamiento. La interpretación es inminente: un MASE inferior a la unidad indica que el modelo supera al modelo de referencia ingenuo, un valor igual a la unidad indica que son equivalentes, y un valor superior a la unidad indica que el modelo funciona peor que la simple predicción anterior. El MASE hereda la independencia de escala del MAPE, al igual que evita su inestabilidad cerca de cero, y tiene la ventaja de introducir una línea de base directamente a su definición.

Una de las características destacables del MASE nace cuando la serie de entrenamiento contiene largos tramos de valores constantes o casi constantes. En dichos casos, el error ingenuo dentro de las muestras de nuestro denominador, provocan que el MASE resultante se vuelva absurdamente grande, incluso cuando el error de predicción absoluto es pequeño Como explicaremos con más detenimiento en capítulos posteriores, este fenómeno se observa en el presente trabajo para el conjunto de datos de una sola sesión, donde la estructura escalonada de la serie hace que el error ingenuo dentro de la muestra sea muy pequeño. Por ello, la importancia de interpretar el MASE junto con las métricas absolutas en lugar de hacerlo de forma aislada.

2.5.5 EL MODELO DE REFERENCIA INGENUA

En base del MASE se halla el modelador ingenuo, que trata de predecir el siguiente valor de la serie sea igual al valor absoluto más reciente: $\widehat{y_{t+1}} = y_t$. A pesar de su simplicidad, el modelador "naive" se usa como una referencia indispensable: no se puede afirmar que

ningún modelo que no consiga superarlo aporte valor predictivo, independientemente de su complejidad. La incorporación de un punto de referencia ingenuo en la evaluación, da lugar a una salvaguarda contra la impresión engañosa de precisión que pueden transmitir los errores absolutos bajos. En concreto, en el contexto de las series financieras y de precios, la hipótesis del paseo aleatorio implica que es difícil superar al modelador ingenuo, ya que la mejor predicción del precio de mañana suele ser el de hoy. Por ello, es fundamental la comparación de los modelos ajustados con este punto de referencia, tal y como se estudiará en el capítulo de resultados.

2.5.6 RESUMEN

Las cuatro métricas descritas previamente forman un conjunto complementario, resumido en la siguiente tabla:

Métrica	Unidad	Independiente de la escala	Dependiente de la escala	Incorpora la baseline
MAE	Original	No	No	No
RMSE	Original	No	Sí	No
MAPE	Porcentaje	Sí	No	No
MASE	original	Sí	No	Sí

Tabla 4 Resumen de las cuatro métricas y sus dependencias.

En línea con las recomendaciones de la bibliografía sobre predicción, las cuatro métricas se presentan conjuntamente a lo largo de este trabajo, junto con el punto de referencia ingenuo, de modo que la magnitud absoluta de error, su sensibilidad a grandes desviaciones y su comparación con una línea de base mínima puedan evaluarse al mismo tiempo.

2.6 REDES NEURONALES RECURRENTES Y LSTM

2.6.1 DE LOS MODELOS LINEALES A LAS REDES RECURRENTES

El enfoque ARIMA presentado en la *sección 2.4*, a pesar de su elegancia teórica y su utilidad en la práctica, está basado en suponer que la relación entre los valores pasados y presentes de una serie es lineal. En cambio, muchos fenómenos del mundo real presentan cambios no lineales que los modelos lineales no pueden representar. Las redes neuronales artificiales ofrecen un enfoque complementario que permite aproximar funciones no lineales y, entre ellas, una arquitectura concreta (la red de memoria a corto y largo plazo, LSTM) se ha diseñado expresamente para modelar datos de forma secuencial. En esta sección se desarrollan los fundamentos teóricos de la tecnología, desde la neurona artificial hasta llegar a una LSTM completa.

2.6.2 LA NEURONA ARTIFICIAL

La unidad fundamental de una red neuronal es la neurona artificial, una función matemática que asigna un conjunto de entradas a una única salida. Dadas las entradas $x_1, \ldots, x_n$, la neurona calcula una suma ponderada de estas entradas, añade un término de sesgo b y aplica una función de activación[18] no lineal f:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

[18] Función de activación, Una función no lineal que se aplica a la suma ponderada de las entradas de una neurona. Sin ella, una red multicapa equivaldría a una simple transformación lineal.

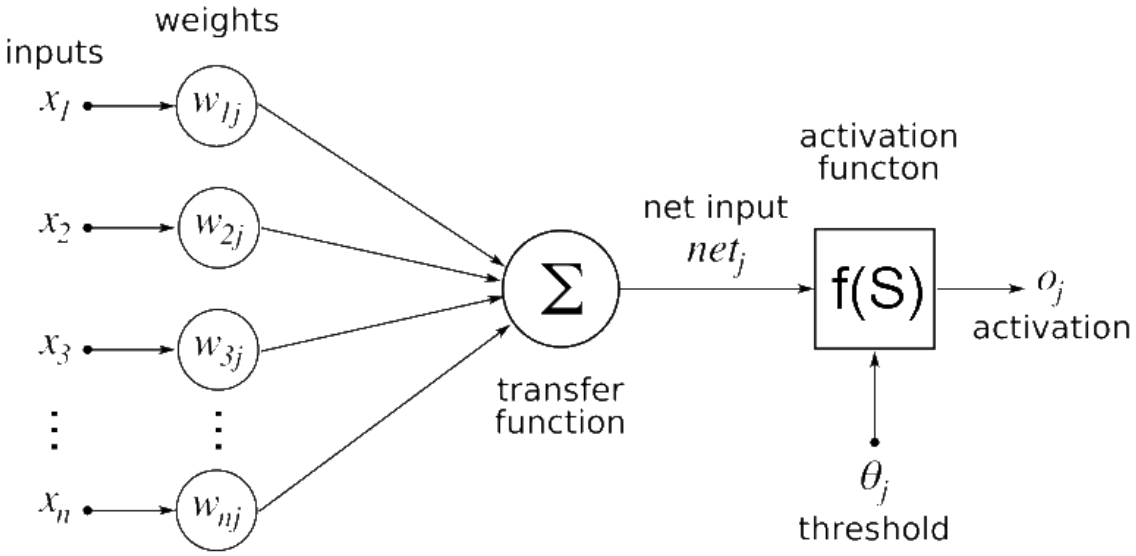

Ilustración 12 Estructura de una neurona artificial. (Fuente: elaboración propia - Fuente: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.png

Los pesos w_i determinan la importancia que tiene cada entrada y son las magnitudes que se ajustan durante el aprendizaje. La función de activación es vital. Sin ella, la combinación de varias neuronas se derrumbaría en una sola transformación lineal, y la red no sería más compleja que una regresión lineal. Entre las opciones más comunes se encuentra la función sigmoide, que mapea su argumento al intervalo (0,1); la tangente hiperbólica, que mapea al intervalo (−1,1); y la unidad lineal rectificada (ReLU), que se define como max(0,x), y que se ha consolidado como la opción predeterminada en las capas ocultas modernas debido a su simplicidad algorítmica y sus ventajosas propiedades de gradiente.

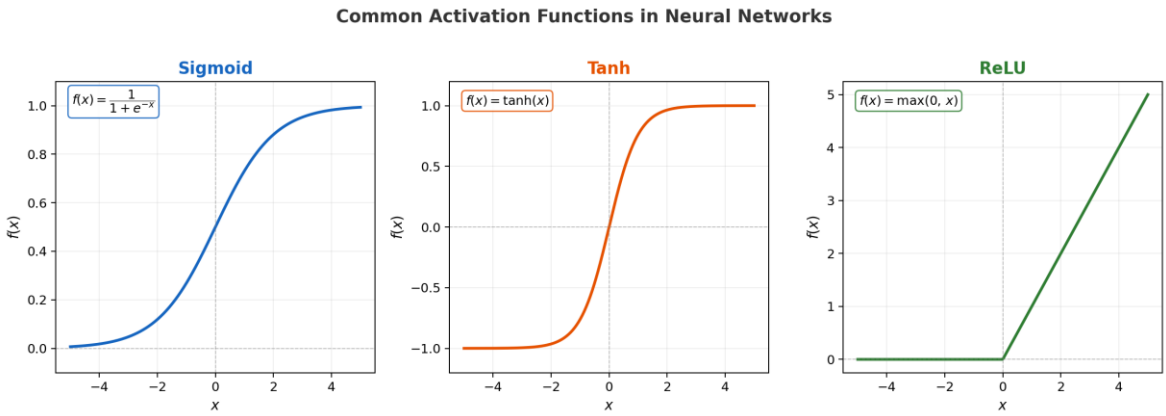

Ilustración 13 Funciones de activación habituales: sigmoide, tanh y ReLU.

2.6.3 REDES NEURONALES DE PROPAGACIÓN DIRECTA

Las neuronas se organizan en capas: una capa de entrada, una o más capas ocultas y una capa de salida. En una red de propagación hacia adelante, la información circula en una sola dirección, de la entrada a la salida, sin bucles. Estas redes se entrenan mediante retropropagación, un algoritmo que calcula el gradiente de una función de pérdida con relación a cada peso difundiendo el error de predicción hacia atrás a través de las capas, tras lo cual los pesos se actualizan mediante descenso por gradiente para reducir el error. Una pasada completa por los datos de entrenamiento se denomina epoch (época), y el entrenamiento suele requerir muchas épocas para alcanzar la convergencia.

Aunque las redes feedforward son potentes funciones de aproximación, sufren una limitación esencial en el contexto de los datos secuenciales ya que carecen de memoria. Cada entrada se procesa de forma independiente, de modo que una red feedforward a la que se le presentan los precios de los últimos cinco días los trata como cinco valores sin orden, ignorando el orden temporal que es fundamental en las series temporales. Esta carencia justifica la introducción de estructuras recurrentes.

2.6.4 REDES NEURONALES RECURRENTES

Una red neuronal recurrente resuelve el problema de la memoria añadiendo un bucle de retroalimentación. La salida de la red en el instante t se retroalimenta como una entrada

adicional en el instante *t+1*. Esto genera un estado oculto h_t que actúa como memoria, conservando un resumen de todo lo que la red ha procesado hasta ese momento. El estado oculto se actualiza así:

$$h_t = f(W_x x_t + W_h h_{t-1} + b)$$

donde $x_t$es la entrada actual y h_{t-1} el estado oculto anterior. Dado que h_t depende de h_{t-1}, la red puede, en principio, detectar patrones que se extienden a lo largo del tiempo. Leer una frase funciona de la misma manera: se entiende cada palabra a la luz de las que la preceden.

En la práctica, las RNN simples son difíciles de entrenar con secuencias largas, y la razón es el problema del gradiente que se desvanece y explota. Cuando el error se propaga hacia atrás a través de muchos pasos temporales, se multiplica una y otra vez por el mismo tipo de factor. Si ese factor es inferior a uno, el gradiente se reduce hacia cero y la influencia de las observaciones más antiguas desaparece, por lo que la red las olvida. Si es superior a uno, el gradiente crece sin control y el entrenamiento se vuelve inestable. En cualquier caso, la RNN simple tiene dificultades para aprender dependencias de largo alcance, y esa limitación es lo que llevó a arquitecturas con puertas como la LSTM.

2.6.5 MEMORIA DE CORTO Y LARGO PLAZO

La red de memoria de corto y largo plazo, introducida por Hochreiter y Schmidhuber en 1997 (Hochreiter & Schmidhuber, 1997), sortea el problema del gradiente de desaparición con un diseño interno más cuidadoso construido en torno a una celda de memoria C_t. La celda actúa como una cinta transportadora: la información puede desplazarse a lo largo de ella a través de muchos pasos temporales con solo pequeños cambios lineales, lo que evita que el gradiente se extinga. Lo que entra en esa cinta, lo que permanece y lo que sale está controlado por tres puertas, cada una de ellas una pequeña capa neuronal con una activación sigmoide que genera valores entre 0 y 1.

La puerta de olvido decide qué parte del estado anterior de la celda se conserva:

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)$$

Un valor de 0 descarta esa parte de la memoria; un valor de 1 la mantiene intacta. La puerta de entrada controla qué nueva información se escribe, trabajando junto con un vector candidato $\tilde{C}_t$ que contiene el posible nuevo contenido:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_c)$$

El estado de la célula se actualiza entonces conservando parte del antiguo y añadiendo parte del nuevo:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

Por último, la puerta de salida decide qué parte del estado de la celda se convierte en el estado oculto en este paso:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot tanh(C_t)$$

Una forma útil de imaginar las tres puertas es un cuaderno. La puerta de olvido tacha lo que ya no importa, la puerta de entrada anota lo que vale la pena conservar y la puerta de salida elige qué parte de la página leer en un momento dado. Dado que el estado de la celda puede transmitir información hacia adelante casi sin degradación, la LSTM conserva el gradiente a lo largo de secuencias largas y aprende las dependencias de largo alcance que una RNN simple no puede.

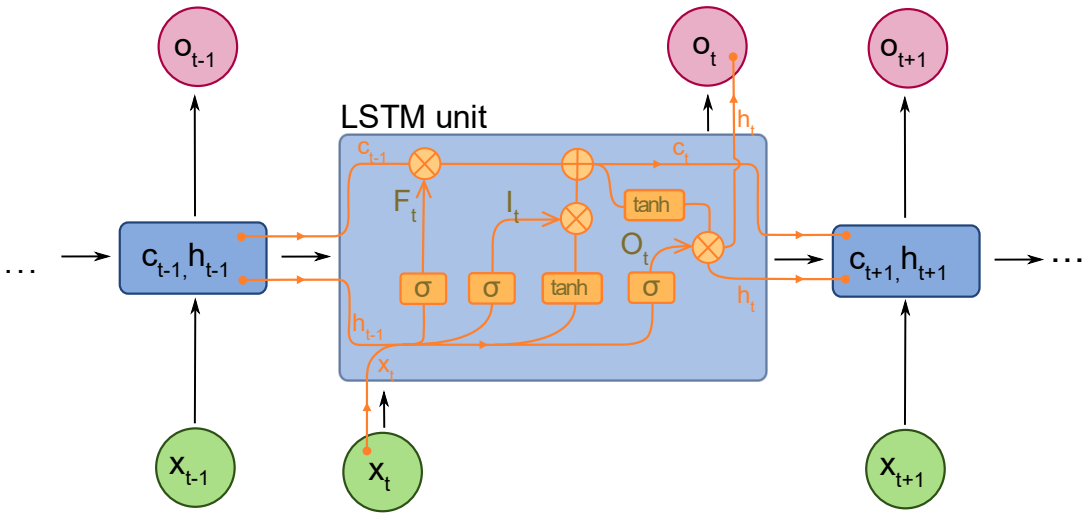

Ilustración 14 Estructura interna de una célula LSTM en la que se muestran las tres puertas y la célula de memoria. - Fuente: https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg

2.6.6 ENTRENAMIENTO DE REDES LSTM

Las LSTM se entrenan mediante retropropagación[19] a través del tiempo. La retropropagación en la que la red recurrente se despliega a lo largo de los pasos temporales de la secuencia y el error recorre todos ellos hacia atrás. Varios elementos forman este proceso. La función de pérdida mide el error y para una tarea de regresión como la predicción de precios, suele ser el error cuadrático medio. El optimizador actualiza los pesos, y el más común es Adam[20], que adapta la tasa de aprendizaje por separado para cada parámetro. La tasa de aprendizaje determina la magnitud de esas actualizaciones. El número de épocas determina cuántas veces la red ve el conjunto de entrenamiento completo, y el tamaño del lote determina cuántos ejemplos procesa antes de cada actualización. Estos hiperparámetros tienen un fuerte efecto tanto en la rapidez con la que converge la red como en la calidad del modelo final.

[19] Retropropagación, Un algoritmo para calcular de forma eficiente el gradiente de la función de pérdida con respecto a cada peso mediante la aplicación de la regla de la cadena en sentido inverso a lo largo de la red, popularizado por primera vez por Rumelhart, Hinton y Williams (1986).
[20] Optimizador Adam, Un algoritmo de optimización adaptativo basado en gradientes propuesto por Kingma y Ba (2015) que mantiene tasas de aprendizaje específicas para cada parámetro basadas en estimaciones de los momentos primero y segundo de los gradientes.

2.6.7 APLICACIÓN DE LSTM A SERIES TEMPORALES

No se puede introducir una serie temporal sin procesar directamente en una LSTM. Primero hay que dividirla en ejemplos de entrada-salida mediante una ventana deslizante[21]. Con una ventana de tamaño *w*, cada ejemplo toma w observaciones consecutivas como entrada y la siguiente observación como objetivo, y la ventana se desplaza entonces hacia adelante paso a paso a lo largo de la serie. Esto convierte la predicción en un problema de aprendizaje supervisado de forma que dados los últimos *w* valores se pueda predecir el siguiente.

Antes del entrenamiento hay que dar dos pasos más. Los datos deben escalarse, ya que las redes neuronales se entrenan mejor cuando las entradas se encuentran en un rango acotado; la opción habitual es una normalización min-max al intervalo de 0 a 1, o una estandarización a media cero y varianza unitaria, y la transformación se invierte tras la predicción para recuperar valores interpretables. Los datos también deben dividirse en conjuntos de entrenamiento y de prueba en un orden temporal estricto. Mezclar los datos al azar, lo cual está bien para problemas no secuenciales, filtraría aquí información del futuro al pasado.

2.6.8 LSTM EN EL PRESENTE TRABAJO Y MÁS ALLÁ

En este trabajo, la LSTM es la versión moderna y no lineal del enfoque clásico ARIMA, y la comparación entre ambos es una de las partes centrales del análisis. Una dificultad práctica es la longitud de las series de precios individuales, alrededor de cincuenta observaciones diarias por ruta, lo que no es suficiente para entrenar una red separada para cada una. El enfoque adoptado aquí consiste en entrenar una única red con todas las series juntas, con características adicionales como la ruta, el número de días que faltan para la salida y el día de la semana, lo que aumenta el número efectivo de ventanas de entrenamiento. Dado lo cerca que están las series de precios de un paseo aleatorio, como se ha establecido anteriormente, es posible que la LSTM no supere por mucho el punto de referencia baseline.

[21] Ventana deslizante, Una técnica de preprocesamiento que convierte una serie temporal en pares de entrada-salida supervisados, tomando como entradas subsecuencias de longitud fija y la observación siguiente como objetivo.

Aun así, comparar los enfoques clásicos y de aprendizaje profundo e informar con honestidad sobre su rendimiento relativo es un resultado valioso en sí mismo.

También cabe señalar que la LSTM ya no es la arquitectura más novedosa para el modelado de secuencias. La Gated Recurrent Unit ofrece una alternativa más sencilla de dos puertas con un rendimiento similar, y el Transformer, que se basa en la atención en lugar de en la recurrencia, es actualmente lo último en el campo de las secuencias, aunque requiere una cantidad considerablemente mayor de datos. Estos, junto con modelos específicos para la predicción como N-BEATS, se señalan aquí como líneas de trabajo futuras.

2.7 ECOSISTEMA TECNOLÓGICO

Todo el software utilizado en este proyecto es de código abierto, por lo que el flujo de trabajo puede reproducirse sin coste alguno. A continuación, se resumen las herramientas por fase.

Python es el lenguaje utilizado en todo el proyecto, elegido por su maduro ecosistema de bibliotecas de análisis de datos. Los datos se recopilan con Playwright, que controla un navegador real e intercepta las respuestas GraphQL de Kiwi.com (*Sección 2.1*). El procesamiento y el análisis utilizan pandas y NumPy. Los modelos ARIMA utilizan statsmodels para las pruebas estadísticas y SARIMAX, y pmdarima para la selección automatizada de órdenes. El LSTM se ha construido con TensorFlow/Keras, con scikit-learn para el escalado y las métricas. Las figuras se generan con matplotlib y seaborn, y las referencias se gestionan con Zotero en estilo APA 7.

Etapa	Herramienta	Objetivo
Adquicisión de datos	Playwright	Automatizar navegador, interceptación de GraphQl
Procesamiento	Pandas, Numpy	Manipulación de datos
Modelización estadística	Statsmodels, pmdarima	Test, Arima
Deep learning	TensorFlow/Keras	LSTM,scalado
Visualización	Matplotlib, seaborn	Plots, correlación
Bibliografía	zotero	Referencias (APA 7)

Tabla 5 Herramientas usadas para cada etapa y sus objetivos.

El análisis se realizó en un ordenador portátil personal estándar; ninguno de los modelos requirió aceleración por GPU dedicada. A continuación, se enumeran las versiones del software utilizadas.

Componente	Versión
Sistema Operativo	Windows 11 (Intel Core i7)
Python	3.12
Pandas/Numpy	2.2
Statsmodels/ pmdarima	0.14/2.0
TensorFlow	2.16
Matplotlib/ seaborn	3.9/0.13

Tabla 6 Componentes usados y sus versiones.

Ilustración 15 Iconos herramientas usadas

3 ESTADO DE LA CUESTIÓN

3.1 EL PROBLEMA DE PREDECIR LOS PRECIOS DE LOS VUELOS

Predecir el precio de un billete de avión es complicado por razones que van más allá de la dificultad de cualquier modelo de predicción concreto. Las aerolíneas no fijan los precios una vez y los mantienen así. Utilizan una disciplina conocida como gestión de ingresos, formalizada en la década de 1980 tras la desregulación del mercado estadounidense y tratada en profundidad en Talluri y van Ryzin (Talluri & Van Ryzin, 2006a) y Belobaba, Odoni y Barnhart (Belobaba et al., 2015). Dentro de este marco, cada asiento es un producto perecedero cuyo precio cambia continuamente para maximizar los ingresos obtenidos antes de que el avión despegue. El resultado, desde el punto de vista del comprador, es una superficie de precios que puede variar varias veces al día en una misma ruta, sin que haya una razón visible detrás de cada cambio.

Existen diversas herramientas comerciales para ayudar al viajero a lidiar con esta volatilidad. Skyscanner, Kayak y Google Flights agregan los precios de distintos proveedores, mientras que Hopper va un paso más allá y ofrece recomendaciones de “comprar o esperar” basadas en sus propios datos internos. Los algoritmos que hay detrás de estas herramientas son de propiedad exclusiva y no se han publicado, lo que hace imposible verificar sus afirmaciones de precisión desde fuera de la empresa. Por esta razón, el conjunto de trabajos académicos revisados en el resto de este capítulo se ha convertido en la principal fuente de conocimiento reproducible sobre cómo se comportan los precios de los vuelos y en qué medida se pueden predecir.

3.2 BIBLIOGRAFÍA ACADÉMICA SOBRE LA PREDICCIÓN DE PRECIOS DE VUELOS

El interés académico por predecir las tarifas aéreas se remonta a más de dos décadas. El trabajo en este ámbito puede organizarse cronológicamente, con tres grandes oleadas que se corresponden con las técnicas de modelización utilizadas en cada momento: un periodo inicial de enfoques basados en reglas y estadísticos, una segunda oleada dominada por el aprendizaje automático clásico y una tercera oleada centrada en el aprendizaje profundo y los modelos ensamblados.

3.2.1 EL PUNTO DE PARTIDA: ETZIONI ET AL. (2003)

El artículo de referencia en este campo es "*To Buy or Not to Buy: Mining Airfare Data to Minimize Ticket Purchase Price*", de Etzioni y sus compañeros de la Universidad de Washington (Etzioni et al., 2003). Su sistema, denominado HAMLET[22], combinaba el aprendizaje basado en reglas mediante el algoritmo RIPPER[23] con un filtro de series temporales, y generaba una recomendación para comprar ahora o esperar. Recopilaron datos de dos rutas de ida y vuelta (Los Ángeles a Boston y Seattle a Washington D.C.), con precios observados durante los 21 días previos a la salida. El ahorro obtenido en comparación con una estrategia ingenua de "comprar ahora" alcanzó alrededor del 24 % en su simulación de 41 días. La contribución de este artículo radica tanto en el método como en el resultado: estableció que los datos de tarifas aéreas de acceso público podían recopilarse, procesarse y transformarse en una regla de decisión útil.

[22] HAMLET, Conjunto de técnicas utilizadas por las aerolíneas y otras industrias con inventario perecedero para maximizar los ingresos mediante la fijación dinámica de precios y el control de inventario.

[23] RIPPER, Recorte incremental repetido para producir reducción de errores, un algoritmo de aprendizaje basado en reglas propuesto por Cohen (1995).

3.2.2 ENFOQUES BASADOS EN LA REGRESIÓN

Una línea de trabajo que siguió a Etzioni se centró en métodos de regresión cada vez más sofisticados.

Janssen (Janssen et al., 2014) aplicó un modelo de regresión lineal de cuantiles mixtos a las tarifas diarias de la ruta de San Francisco a Nueva York-JFK, utilizando datos proporcionados por infare.com. Su modelo incluía dos características: el número de días que faltaban para la salida y si la fecha de salida caía en fin de semana. El modelo funcionó bien en horizontes temporales largos, pero perdió precisión a medida que se acercaba la fecha de salida, un patrón que se repite en estudios posteriores.

Lantseva y sus colaboradores (Lantseva et al., 2015) estudiaron el mercado de la aviación ruso, centrándose en la ruta de Moscú a San Petersburgo, utilizando datos de *AviaSales* y *Sabre* para vuelos tanto nacionales como internacionales hasta 90 días antes de la salida. Sus resultados indicaron que, en el caso de los vuelos internacionales, las tarifas más baratas se obtenían reservando con antelación, mientras que en los vuelos nacionales el patrón se invertía y los precios de última hora tendían a ser más bajos. Este tipo de comportamiento dependiente de la ruta, en el que una única estrategia de predicción no se puede generalizar a todos los mercados, es una de las conclusiones recurrentes en este campo.

3.2.3 APRENDIZAJE AUTOMÁTICO CLÁSICO

Un tercer grupo de artículos introdujo métodos de aprendizaje automático como las máquinas de vectores de soporte, los bosques aleatorios y la clasificación bayesiana ingenua.

Ren y sus colaboradores (Ren et al., 2014) utilizaron regresión lineal, Bayes ingenuo, regresión softmax y máquinas de vectores de soporte para clasificar los precios de los billetes en cinco bandas en relación con el precio medio de la ruta. Trabajaron con más de nueve mil observaciones y seis características, incluyendo el inicio de la semana de salida, la fecha de cotización y el número de escalas en la ruta. El mejor error de entrenamiento fue cercano al 22,9 %, obtenido mediante regresión lineal.

Wohlfarth e investigadores (Wohlfarth et al., 2011) añadieron un paso de agrupamiento antes de la clasificación. Utilizaron K-means para agrupar vuelos con un comportamiento de precios similar, y luego aplicaron un árbol de decisión CART[24] y un bosque aleatorio para interpretar las reglas y clasificar la importancia de las características. Destacaron, en particular, la relevancia del número de asientos restantes en el avión, una variable que generalmente no es visible para un rastreador externo.

Tziridis y sus compañeros (Tziridis et al., 2017) compararon ocho modelos de aprendizaje automático en un conjunto de datos de Aegean Airlines, incluyendo redes neuronales artificiales, bosques aleatorios, máquinas de vectores de soporte y regresión logística. El mejor modelo de regresión alcanzó una precisión del 88 %. En su comparación, el árbol de regresión se identificó como el modelo más robusto, menos sensible a los cambios en el conjunto de características de entrada que las alternativas.

3.2.4 DEEP LEARNING Y CONJUNTOS MODERNOS

La ola más reciente de trabajos se ha centrado en el aprendizaje profundo y en conjuntos que combinan varios aprendices básicos.

Huang (Antonio et al., 2017) aplicó redes neuronales artificiales combinadas con un algoritmo genético para predecir los ingresos por venta de billetes de una agencia de viajes. Las variables de entrada incluían indicadores macroeconómicos como los precios internacionales del petróleo, el índice bursátil de Taiwán y la tasa de desempleo local. El algoritmo genético se utilizó para seleccionar el subconjunto óptimo de características para la red neuronal. El modelo alcanzó un error porcentual absoluto medio de alrededor del 9,11 % en su conjunto de prueba.

Santana y compañeros (Santana & Mastelini, 2017) propusieron un modelo Deep Regressor Stacking en el que se combinaban el bosque aleatorio y la regresión de vectores de soporte

[24] CART, Árboles de clasificación y regresión, un algoritmo de árboles de decisión introducido por Breiman et al. (1984).

en un conjunto multiobjetivo. Argumentaron que este tipo de arquitectura podría transferirse a otros ámbitos problemáticos con una adaptación limitada.

Wang y sus investigadores (Wang et al., 2019) utilizaron redes neuronales recurrentes de tipo LSTM para pronosticar las tarifas en rutas nacionales chinas, con resultados prometedores en horizontes cortos. Trabajos más recientes han comenzado a explorar arquitecturas Transformer y modelos híbridos que combinan el aprendizaje profundo con componentes estadísticos clásicos, aunque la bibliografía al respecto aún se está consolidando.

3.2.5 ARIMA APLICADO A SERIES DE PRECIOS DE VUELOS

Aunque los modelos ARIMA son una herramienta de larga tradición en la previsión de series temporales, su aplicación directa a los datos de precios de vuelos está menos representada en la literatura que los enfoques de aprendizaje automático discutidos anteriormente. Algunos estudios han utilizado ARIMA como referencia para comparar métodos más complejos, y en general informan de un rendimiento modesto en términos absolutos, con el modelo convergiendo hacia la media del periodo de entrenamiento a medida que aumenta el horizonte de previsión. Este patrón es coherente con el comportamiento de tipo paseo aleatorio de las series de precios en mercados eficientes, formalizado por Fama (Fama, 1965), y tiene implicaciones directas sobre lo que cabe esperar razonablemente de cualquier método de predicción univariante aplicado a la predicción de tarifas aéreas a corto plazo.

3.2.6 LSTM APLICADO A SERIES DE PRECIOS

La aplicación de las redes LSTM a series temporales financieras y de precios se ha expandido sustancialmente desde 2015, impulsada por su capacidad para modelar dependencias no lineales y patrones de largo alcance. El trabajo de Wang et al. (2019) citado anteriormente es uno de los pocos estudios que aplica esta arquitectura directamente a las tarifas aéreas, pero existe una bibliografía más amplia sobre el uso de LSTM para precios de activos, tipos de cambio y series de demanda en el comercio minorista. Dos observaciones de esa bibliografía más amplia son especialmente relevantes para este trabajo. La primera es que

las redes LSTM suelen requerir conjuntos de datos más grandes que los modelos clásicos para superar su rendimiento, y obtienen peores resultados en series cortas. La segunda es que, incluso con conjuntos de datos más grandes, su mejora con respecto a un punto de referencia ingenuo suele ser más modesta de lo que sugieren las cifras de precisión generales, especialmente cuando los horizontes de previsión son cortos.

3.3 METODOLOGÍAS DE RECOPILACIÓN DE DATOS

Una dificultad común a todos los trabajos citados anteriormente es que los datos sobre tarifas aéreas no están disponibles en un formato limpio y estandarizado. Los investigadores han utilizado tres estrategias principales. Las aerolíneas estadounidenses comunican datos agregados sobre tarifas a las bases de datos públicas T100 y DB1A/DB1B[25], que son de acceso libre pero tienen una resolución limitada y rara vez contienen suficiente información específica sobre los vuelos como para ser utilizadas de forma aislada ((Mumbower et al., 2015);(Newman et al., 2014)). Algunos autores, como Rama-Murthy (2006), han combinado estas fuentes públicas con datos privados de proveedores como Sabre AirPrice[26], lo que conlleva la limitación de representar únicamente a los propios clientes de los proveedores en lugar de a todo el mercado. Una tercera estrategia, más habitual en trabajos recientes, consiste en obtener los datos directamente de motores de búsqueda de acceso público mediante web scraping, como en el artículo fundacional de Etzioni y sus compañeros. Las condiciones legales y éticas bajo las cuales dicho scraping es aceptable se analizan en detalle en la *sección 2.1* de este trabajo.

La elección de la fuente de datos tiene consecuencias que van más allá de la calidad de los datos. La propia estrategia de recopilación determina las propiedades estadísticas de las

[25] T100 y DB1A/DB1B, conjuntos de datos públicos mantenidos por la Oficina de Estadísticas del Transporte de EE. UU., que recogen datos agregados sobre la operación y las tarifas de las aerolíneas.

[26] Sabre AirPrice, producto de datos de precios comerciales ofrecido por Sabre Corporation, un importante proveedor de tecnología para el sector de los viajes.

series temporales resultantes, un aspecto que ha recibido poca atención en los trabajos publicados y que se analiza en profundidad en el capítulo de metodología de este proyecto.

3.4 LIMITACIONES IDENTIFICADAS Y OPORTUNIDADES DE INVESTIGACIÓN

Si se considera la bibliografía en su conjunto, se observan varias limitaciones recurrentes.

La primera es el uso desigual de las líneas de referencia. Una parte sustancial de los trabajos publicados presenta cifras de precisión absoluta, como errores porcentuales absolutos medios del 9 o el 10 por ciento, sin compararlas con un pronosticador ingenuo o una referencia de paseo aleatorio. Como se analiza en la *sección 2.5* de este trabajo, esta omisión dificulta evaluar si las mejoras comunicadas son significativas en la práctica o si simplemente reflejan la inercia de la serie de precios.

La segunda es la fuerte dependencia de los resultados del mercado, la ruta y el conjunto de datos específicos utilizados. El mismo modelo que funciona bien en un conjunto de rutas puede funcionar mal en otro, y no hay consenso sobre qué técnica debería preferirse en general. Esto requiere estudios que comparen los enfoques clásicos y de aprendizaje profundo en el mismo conjunto de datos y bajo el mismo protocolo de evaluación.

La tercera es el tratamiento limitado de la propia metodología de recopilación de datos. La elección entre recopilar el mismo conjunto de rutas diariamente a lo largo de una campaña prolongada y capturar muchas rutas a la vez en una sola sesión tiene consecuencias claras para la estructura estadística de la serie resultante, pero esta cuestión rara vez se aborda en la bibliografía.

Estas tres lagunas enmarcan la contribución del presente trabajo. Los objetivos, la metodología y la forma específica en que este proyecto aborda cada una de ellas se presentan en el siguiente capítulo.

A modo de resumen de la revisión presentada en este capítulo, la figura siguiente recoge en orden cronológico los principales trabajos académicos sobre predicción de precios de vuelos desde 2003 hasta la actualidad, agrupados por la familia metodológica a la que pertenecen. Se pueden observar cuatro etapas razonablemente diferenciadas. La primera, inaugurada por el trabajo de Etzioni y sus colegas, está dominada por los sistemas basados en reglas y la minería de datos. La segunda abarca aproximadamente la década siguiente y se caracteriza por la consolidación de la regresión estadística y el aprendizaje automático clásico, con modelos como Random Forest, SVM y la regresión cuantílica. La tercera etapa introduce el aprendizaje profundo, y en particular las redes recurrentes LSTM, en este ámbito. La cuarta y más reciente etapa está marcada por el auge de las arquitecturas basadas en la atención, cuya aplicación específica a la predicción de tarifas aéreas sigue siendo, por el momento, limitada.

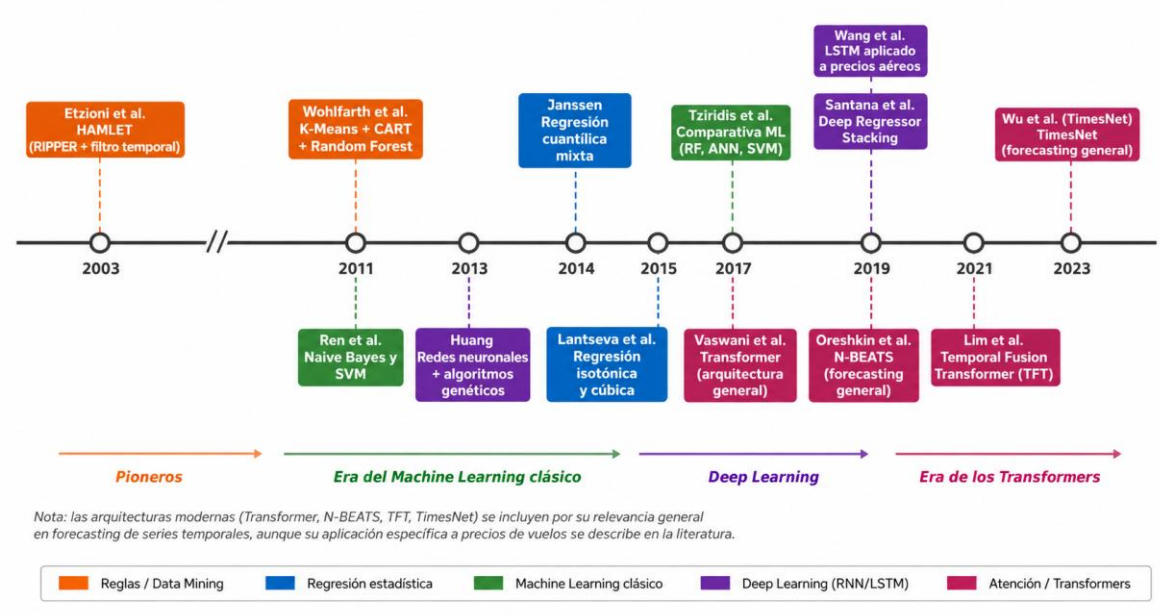

Ilustración 16 Evolución cronológica de las técnicas de predicción aplicables a precios de vuelos, agrupadas por familia metodológica.

4 DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

La revisión de los trabajos existentes presentada en el capítulo anterior pone de manifiesto tres lagunas que el presente proyecto pretende abordar.

La primera es el uso limitado de referencias formales en la bibliografía publicada sobre la predicción de precios de vuelos. La mayoría de los estudios presenta cifras de precisión absoluta sin comparar sus modelos con un predictor ingenuo (naive). Como resultado, es difícil determinar si las mejoras comunicadas reflejan un valor predictivo genuino o simplemente la inercia de la serie de precios. Este proyecto aborda esta laguna evaluando cada modelo frente a un punto de referencia ingenuo explícito y presentando la métrica MASE, que expresa el error de predicción en relación con el del predictor ingenuo.

La segunda es la ausencia de una comparación sistemática entre los métodos clásicos de series temporales y los enfoques de aprendizaje profundo aplicados al mismo conjunto de datos y evaluados bajo el mismo protocolo. La mayoría de los estudios publicados aplican una única familia de técnicas, lo que hace que la comparación entre estudios sea poco fiable. Este proyecto aplica tanto ARIMA como LSTM al mismo conjunto de series de precios de vuelos, utilizando un preprocesamiento, divisiones de entrenamiento y prueba, y métricas de evaluación idénticos.

La tercera es la escasa atención prestada a la propia estrategia de recopilación de datos. Como se ha comentado en el *capítulo 3*, los diferentes enfoques de scraping producen series temporales con propiedades estadísticas muy diferentes. Este proyecto recopila dos conjuntos de datos distintos utilizando dos estrategias, una extracción en una sola sesión y una campaña longitudinal diaria, y documenta las consecuencias de cada una de ellas para las series y los modelos resultantes.

4.2 OBJETIVOS

El proyecto persigue los siguientes objetivos específicos:

1. Diseñar e implementar un sistema de scraping web capaz de extraer datos de precios de vuelos de Kiwi.com mediante la interceptación de la API, produciendo dos conjuntos de datos: uno procedente de una extracción en una sola sesión que abarca seis rutas desde Madrid (Conjunto de datos A), y otro procedente de una campaña longitudinal diaria que abarca cuatro rutas durante aproximadamente cincuenta días (Conjunto de datos B).
2. Aplicar modelos ARIMA a ambos conjuntos de datos, utilizando una metodología rigurosa que incluye pruebas de dual estacionariedad (ADF y KPSS), selección automatizada del orden mediante AIC, diagnósticos de residuos (Ljung-Box, Shapiro-Wilk) y comparación con una línea de base ingenua a través de múltiples métricas de error (MAE, RMSE, MAPE, MASE).
3. Implementar una red neuronal LSTM para la misma tarea de predicción en el conjunto de datos B, incorporando características adicionales como el número de días hasta la salida y el día de la semana, y comparar su rendimiento con el de ARIMA y el punto de referencia ingenuo bajo el mismo marco de evaluación.
4. Realizar un análisis aplicado de la relación entre el momento de la compra y el precio, utilizando las series individuales de fechas de vuelo del conjunto de datos B, con el objetivo de identificar patrones prácticos en la evolución de los precios a medida que se acerca la fecha de salida.
5. Documentar y discutir las consecuencias metodológicas de las dos estrategias de recopilación de datos, incluyendo el efecto escalonado observado en los datos de una sola sesión y el papel de la interpolación en series longitudinales con días faltantes.

4.3 METODOLOGÍA

La metodología sigue cuatro etapas.

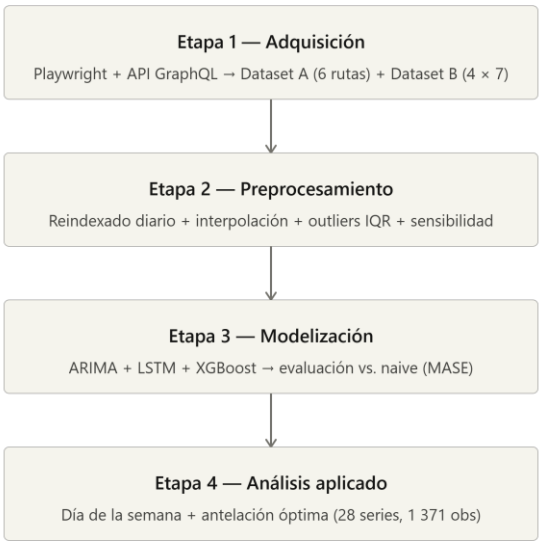

Ilustración 17. Diagrama de la metodología a seguir

- Etapa 1. Adquisición de datos. Se recogen dos conjuntos de datos de Kiwi.com utilizando un rastreador basado en Playwright que intercepta las respuestas de la API GraphQL. El conjunto de datos A captura una amplia instantánea de seis rutas desde Madrid en una sola sesión. El conjunto de datos B captura la evolución diaria de los precios para cuatro rutas y siete fechas de salida fijas durante un periodo de aproximadamente cincuenta días.
- Etapa 2. Preprocesamiento. Ambos conjuntos de datos se someten a reindexación a frecuencia diaria, interpolación lineal de los días que faltan y detección de valores atípicos utilizando el método del rango intercuartílico (factor IQR de 1,5 para el conjunto de datos A y de 3,0 para el conjunto de datos B, dado su menor tamaño de muestra). Un análisis de sensibilidad compara el ARIMA interpolado con un modelo SARIMAX ajustado a la serie original con valores faltantes para verificar que la interpolación no distorsiona los resultados.
- Etapa 3. Modelización. Se ajustan modelos ARIMA a cada ruta utilizando el algoritmo escalonado automatizado de Hyndman y Khandakar (2008), con el AIC como criterio de selección. Se entrena una red LSTM sobre las series combinadas del conjunto de datos B con una estructura de entrada de ventana deslizante y

características adicionales. Ambos modelos se evalúan en un conjunto de prueba retenido (división 80/20 para el conjunto de datos A, 70/30 para el conjunto de datos B) utilizando MAE, RMSE, MAPE y MASE, con la previsión ingenua como referencia.

- Etapa 4. Análisis aplicado. Las 28 series individuales del conjunto de datos B (una por combinación de ruta y fecha de salida) se utilizan para estudiar la relación entre los días hasta la salida y el nivel de precios, con el objetivo de identificar patrones de momento de compra.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

4.4.1 CRONOLOGÍA DEL PROYECTO

El proyecto se desarrolló entre octubre de 2025 y junio de 2026, dividido en siete fases que se solapaban parcialmente. La ilustración X muestra la distribución de estas fases a lo largo del periodo de nueve meses.

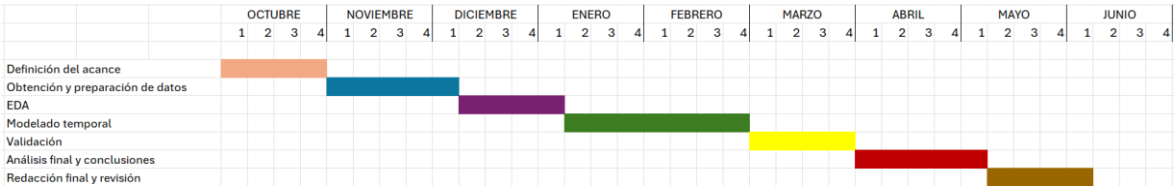

Ilustración 18. Diagrama de Gantt cronología del proyecto

La campaña para el conjunto de datos B impuso una restricción externa al calendario: el rastreador diario funcionó del 8 de abril al 5 de junio, lo que significó que los análisis de LSTM y del momento de la compra no pudieron comenzar hasta que se cerró ese periodo. Esta dependencia explica el intervalo entre el final del trabajo con ARIMA en el conjunto de datos A y el inicio del proceso de aprendizaje automático en el conjunto de datos B.

4.4.2 ESTIMACIÓN ECONÓMICA

El coste del proyecto se desglosa en tres partidas: personal, equipamiento y software. No se contó con financiación externa; el trabajo se llevó a cabo íntegramente con los recursos propios del autor y las licencias de software proporcionadas por la universidad.

Los costes de personal se estiman a partir de las 300 horas de trabajo correspondientes a un proyecto de fin de carrera de 12 ECTS, valoradas a una tarifa bruta de 15 euros por hora, lo que se ajusta al extremo inferior del rango salarial de un ingeniero junior en España.

Concepto	Horas	Coste/hora	Total
Investigación y revisión bibliográfica	40 h	15 €/h	600 €
Diseño e implementación del scraping	40 h	15 €/h	600 €
Preprocesamiento y análisis exploratorio	35 h	15 €/h	525 €
Modelización (ARIMA, LSTM, XGBoost)	70 h	15 €/h	1 050 €
Análisis de resultados	35 h	15 €/h	525 €
Redacción de la memoria	60 h	15 €/h	900 €
Depuración, revisión y preparación de la defensa	20 h	15 €/h	300 €
Total personal	**300 h**		**4 500 €**

Tabla 7. Desglose horario y económico personal

Concepto	Coste de adquisición	Vida útil	Uso en el proyecto	Coste imputado
Portátil (Intel Core i7, 16 GB RAM)	1 100 €	4 años	9 meses	206 €

Tabla 8. Desglose económico del equipo de trabajo

Todo el software utilizado en el proyecto es de código abierto o está cubierto por la licencia institucional de la universidad, por lo que no se incurrió en ningún coste adicional.

A continuación, se resume el coste total estimado del proyecto.

Partida	Importe
Personal	4 500 €
Equipamiento (amortizado)	206 €
Software y servicios	0 €
Coste total del proyecto	**4 706 €**

Tabla 9. Desglose económico total del proyecto

5 SISTEMA DESARROLLADO

5.1 DESCRIPCIÓN GENERAL DEL SISTEMA

El sistema creado para este proyecto recopila diariamente los precios de los billetes de avión con origen en Madrid y los procesa mediante tres modelos de predicción para comprobar si alguno de ellos supera una referencia elemental, predecir que el precio de mañana será el mismo que el de hoy.

La figura 5.1 ofrece una visión general del proceso. Los datos se extraen de *Kiwi.com* interceptando las consultas GraphQL que la propia aplicación web realiza a su API interna de precios. Un navegador sin interfaz manejado por Playwright se encarga de ello, guardando un archivo de texto para cada ruta, fecha de salida y fecha de extracción. A continuación, estos archivos sin procesar se recopilan en un DataFrame de pandas, se depuran (eliminando valores atípicos e interpolando los días que faltan) y se amplían con características del calendario y de las rutas. Los datos procesados sirven de entrada a tres modelos en paralelo: ARIMA, LSTM y XGBoost. Todas sus predicciones se comparan con la referencia "naive" en el mismo conjunto de prueba.

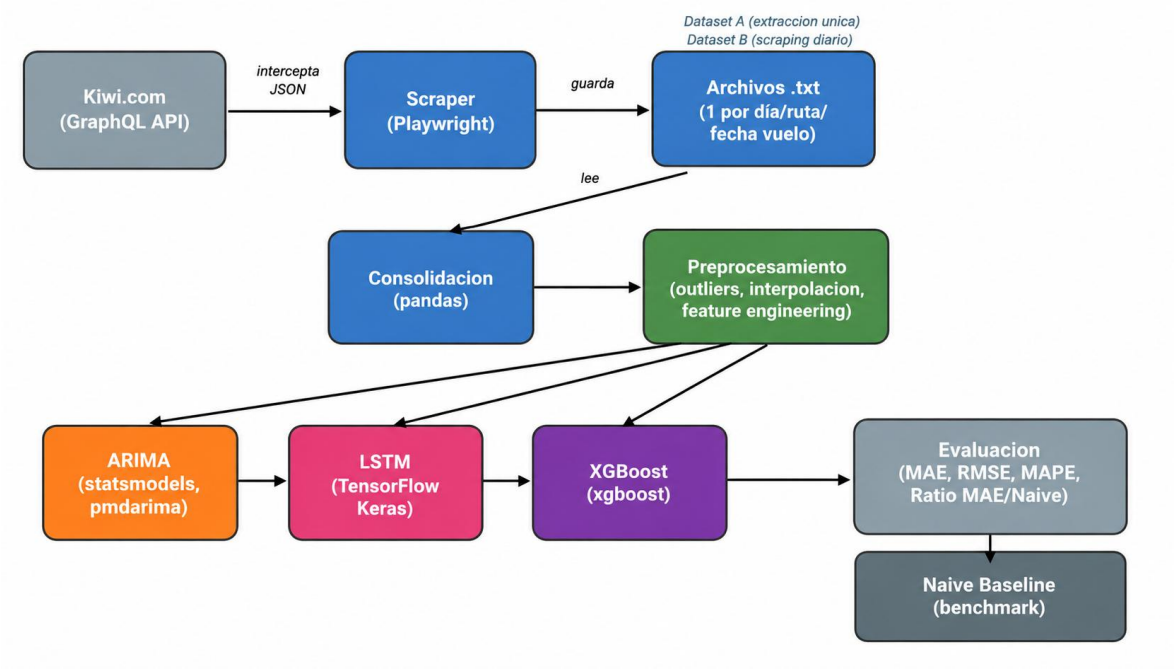

Ilustración 19. Arquitectura general del sistema de predicción de vuelos.

Se analizaron seis rutas desde Madrid-Barajas (*Ilustración 20*). Cuatro de ellas se registraron en ambos conjuntos de datos:

- MAD – BCN (483 km)
- MAD – LHR (1261 km)
- MAD – BER (1869 km)
- MAD – IST (3210 km)

Las otras dos, solo se incluyeron en la extracción inicial de una sola sesión (conjunto de datos A):

- MAD – CDG (1065 km)
- MAD – SIN (10564 km)

Estas no se trasladaron a la campaña de scraping diario debido a que la extracción diaria para rutas interoceánicas era más tediosa. Ejecutar un scraping diario durante 60 días consecutivos para cuatro rutas y siete fechas de salida para cada una de ellas que ya suponía

mantener 28 series temporales en paralelo, por lo que añadir aún más rutas habría sobrecargado la infraestructura más allá de lo que resultaba viable para un proyecto realizado por una sola persona.

Dicha selección de rutas no fue aleatoria, sino que por ejemplo para Barcelona, es el trayecto aéreo nacional más transitado de España, dominado por operadores de bajo coste. Londres y Berlín representan mercados europeos típicos de medio recorrido con una mezcla de tipos de compañías aéreas. Estambul añade un destino no incluido en la Unión Europea donde las condiciones normativas y competitivas son diferentes. Y por último, Singapur, aunque solo se analizó en el Conjunto de datos A, amplía el alcance a los precios intercontinentales de largo recorrido, donde los costes de los billetes y la volatilidad se muestran muy diferentes a los de los vuelos cortos europeos.

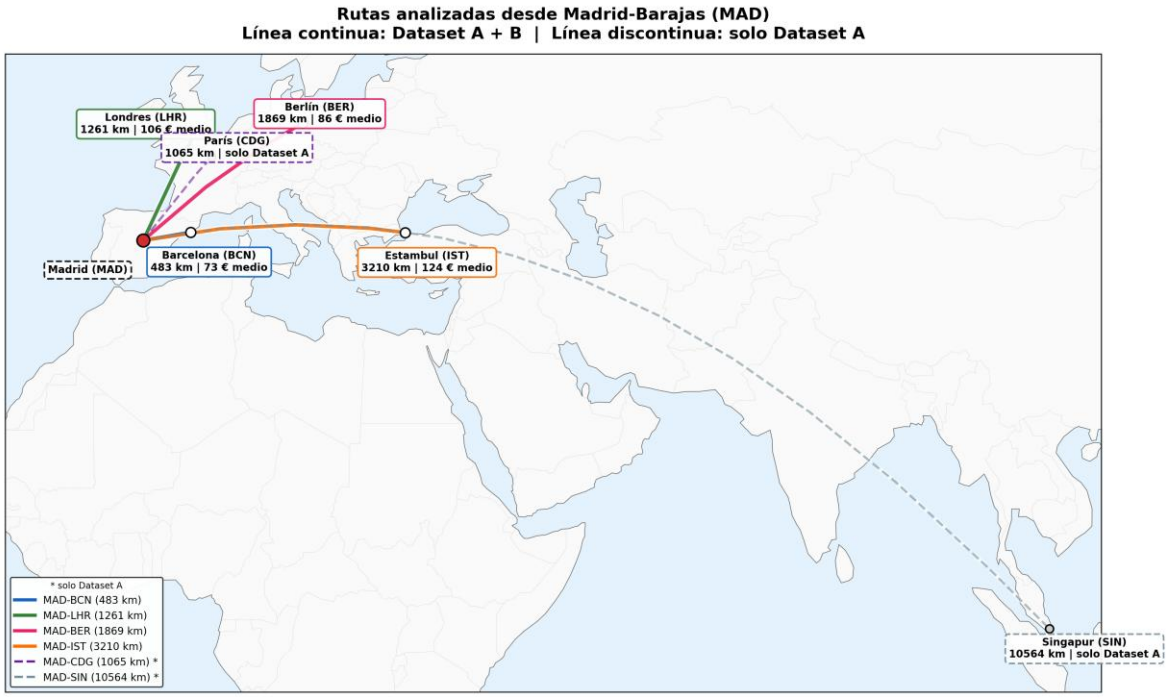

Ilustración 20 Mapa cartográfico de las rutas analizadas - Fuente: elaboración propia – Datos cartográficos: Natural Earth (dominio público).

El resto de este capítulo sigue el proceso de izquierda a derecha visto en la *ilustración 19*, recopilación de datos (*5.2*), preprocesamiento (*5.3*), desarrollo de características (*5.4*), los tres modelos y sus configuraciones (*5.5*) y el enfoque de evaluación (*5.6*).

5.2 RECOPILACIÓN DE DATOS

5.2.1 EL PROCESO DE EXTRACCIÓN

La web de extracción de los precios de billete de avión, *Kiwi.com*, no publica una API pública, pero tampoco es necesario ya que cada vez que un usuario carga una página de resultados de búsqueda, el propio JavaScript del sitio web envía una solicitud GraphQL (Hartig & Pérez, 2018) a un punto final interno, y es en este punto antes de que se lance el HTML, el que nos devuelve un bloque de datos JSON que contiene exactamente la información necesaria para este trabajo. De dicho JSON, nos ha interesado la información del precio más barato disponible, la aerolínea operadora, el número de escalas (0,1 o 2) y la fecha de salida. El scraper aprovecha esto utilizando Playwright (Shevelenkov, 2024) para lanzar una instancia de Chromium sin interfaz gráfica, simulando a una persona humana navegar a una URL de búsqueda específica y esperar respuestas de red que coincidan con el patrón del punto final GraphQL. Cuando llega una, el scraper analiza el JSON, extrae los campos que nos son de relevancia y los escribe como una única línea separada por comas en un archivo de texto. El código completo del scraper, incluida la lógica de interceptación de red y la configuración de la programación diaria, se proporciona en el *Anexo A*.

Los resultados para el caso del dataset B (extracción dinámica) se almacenan en archivos estructurados:

- Cada destino en carpetas diferentes, *figura 21.1*
- Cada destino formado por as distintas fechas analizadas, *figura 21.2*
- Cada fecha tiene archivos *.txt* con cada día de extracción, *figura 21.3*
- Para cada destino en un día concreto se almacenan las variables separadas por comas, como se muestra en la figura *21.4*

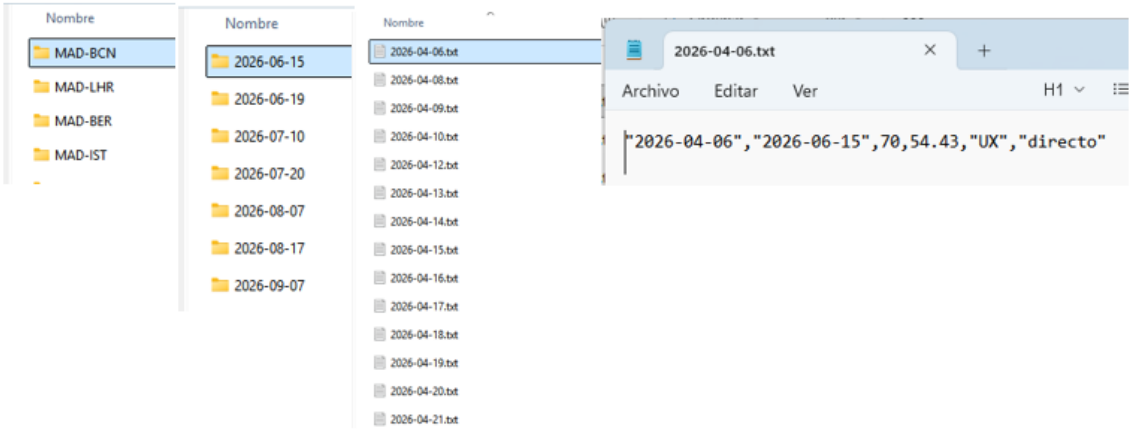

Ilustración 21 Estructura almacenamiento de archivos. Ilus.21.1 rutas destino. Ilus. 21.2 fechas analizadas. Ilus 21.3 archivos .txt con cada día de extracción. Ilus.21.4 ejemplo de un día concreto.

Para el caso del dataset con una única extracción, la distribución es similar pero para cada día de análisis todos los precios de billete posibles.

Cada archivo de texto contiene una observación, en el formato `fecha_raspado`, `fecha_vuelo`, `días_hasta`, precio, aerolínea, escalas. Esta arquitectura de archivos fue una decisión premeditada sobre la base de datos, ya que si el rastreador tenía un fallo en un día determinado (lo cual ocurría en ocasiones, como veremos más adelante), era muy sencillo identificar el archivo faltante y el resto de los datos se podían seguir leyendo de forma independiente. El número de registros con los que vamos a trabajar no justifica el uso de una base de datos más compleja. No obstante, un estudio más amplio requerirá el consiguiente cambio de sistema de base de datos.

Este enfoque tiene una desventaja importante, que causó un problema que no se hizo visible hasta la fase de análisis. Las respuestas GraphQL de *Kiwi.com* son susceptibles a la sesión. Dentro de una misma sesión de navegación, el back-end parece almacenar en caché o cuantificar los precios devolviendo el mismo valor varias veces para consultas similares en lugar de volver a calcular precios nuevos cada vez. Para un scraping de una sola ejecución, esto pasa totalmente desapercibido por lo que para el análisis que en gran medida depende de la variabilidad, esto resulto ser perjudicial. Las dos secciones siguientes explican cómo afectó esto a cada conjunto de datos.

5.2.2 CONJUNTO DE DATOS A — EXTRACCIÓN EN UNA SOLA SESIÓN

El conjunto de datos A fue la primera pasada de scraping y, en principio la única prevista. El scraper se ejecutó una vez, durante una única sesión de navegación prolongada, y consultó seis rutas desde Madrid: Barcelona, Berlín, París-CDG, Estambul, Londres-Heathrow y Singapur.

Para cada ruta, extrajo aproximadamente 198 fechas de salida futuras, lo que generó una visión general transversal de cómo varían los precios en función de los días que faltan para la salida. En la teoría, esto parecía correcto pero en la práctica, al trazar las series de precios, las seis mostraban un notable patrón en escalera (*staircase*). Que son largos periodos de estabilidad interrumpidos por saltos discretos ocasionales, con solo entre 18 y 21 valores de precio únicos en casi 200 puntos de datos (*Tabla 10*), por lo que no mostraba gran variabilidad como se esperaba. Al principio, esto parecía un fenómeno real de la fijación de precios de las aerolíneas, ya que estas utilizan categorías de tarifas discretas. Pero la cardinalidad era demasiado baja. Una estructura de tramos de tarifas auténticos mostraría docenas de valores distintos en 200 observaciones no únicamente 20.

La razón, que se averiguó más tarde, es el comportamiento de almacenamiento en caché de la sesión mencionado anteriormente. Como el rastreador consultó las 198 fechas dentro de la misma sesión de Playwright, el back-end servía respuestas repetidas almacenadas en caché en lugar de realizar recálculos independientes. El conjunto de datos A captaba la estructura de la memoria caché de una única sesión de usuario, no la distribución de precios en sí.

Ruta	Fechas	Precios únicos	Rangos de precios (EUR)
MAD-BCN	197	18	124-146
MAD-BER	198	21	137-163
MAD-CDG	197	19	142-163
MAD-IST	197	20	157-181
MAD-LHR	198	18	172-197
MAD-SIN	197	21	544-611

Tabla 10 Resumen del Dataset A, extracción en una única sesión

A pesar de las limitaciones del dataset A, este no se llegó a descartar. Sigue aportando gran útil información en el análisis, ya que sirve como ejemplo de cómo un fallo menor en la recopilación de los datos puede desvariar el modelo posterior y que se reduzca a una previsión más simplista, como se discutirá en el *capítulo 6*. También motivó todo lo relacionado con el dataset B, que se creó concretamente para solucionar todos los problemas que el anterior estaba generando.

5.2.3 CONJUNTO DE DATOS B: SCRAPING PERIÓDICO DIÁRIO

La solución para resolver era teóricamente sencilla, pero técnicamente tediosa. En vez de ejecutar el scraper una sola vez y ver las observaciones disponibles, se trataba de cambiar la estrategia. Para este caso se buscó consultar un conjunto pequeño y fijo de fechas scrapeando diariamente, iniciando cada vez una nueva sesión de navegación. De esta manera, una sesión nueva cada día, obligaría a realizar una nueva consulta incluso si *Kiwi.com* almacena los precios en caché dentro de una sesión.

Este proceso comprendía entre el 8 de abril y el 5 de junio de 2026, abarcando un total de 49 días de información útiles, ya que el periodo original fueron de 60 días menos 10 días en los que el rastreador no se ejecutó de manera correcta por diversas razones externas. En esta

ocasión se hizo un seguimiento de cuatro rutas: Barcelona, Berlín, Estambul y Londres-Heathrow.

Como se puede apreciar se descartó París-CDG y Singapur para que la carga de información y trabajo diaria fuera manejable. Para cada una de las rutas se estudiaron siete fechas de salida de vuelos futuras, repartidas aproximadamente entre dos semanas y cuatro meses antes del inicio del scraping. Como resultado se obtuvo 28 series temporales individuales para cada combinación (ruta, *fecha_de_vuelo*) con hasta 49 observaciones diarias de cada una, dando un total aproximado de 1371 registros válidos en total. La tarea de programación que lee todos los archivos de texto en un dataFrame forma parte del proceso principal, descrito en el *Anexo II, Fase 3A*.

Ruta	Series	Puntos/serie	Precio Medio (EUR)	Desviación estándar (EUR)	Días sin datos
MAD-BCN	7	48-49	73.2	11.8	10
MAD-BER	7	49	86.1	20.3	10
MAD-IST	7	49	124.2	26.7	10
MAD-LHR	7	49	106.2	18.4	10

Tabla 11 resumen de dataset B.

Esta extracción forma parte de la estructura principal para este trabajo, debido a que cada serie captura la evolución temporal del precio de un vuelo en concreto a medida que se acerca la fecha del vuelo. Además, la variable `days_until` disminuye en uno cada día, fijando cada una de las observaciones a un punto de la curva de reservas. El comportamiento de subida de precios, bajada o que se mantengan estables hasta los últimos días, es exactamente el tipo de estructura que se busca y que ARIMA y LSTM son capaces de modelar.

Caben destacar dos aspectos de funcionamiento, ya que aparecen más adelante en el análisis.

En primer lugar, se descartaron los datos del 6 de abril del 2026, debido a que esta fue la primera vez del scraping y los precios se desviaban notablemente de todos los posteriores (escalas diferentes, aerolíneas diferentes, y en algunos casos valores que no volvieron a aparecer en ningún otro momento del estudio). La razón más probable es que el rastreador aún estuviera perfeccionando en esa ejecución (se aplicó un tipo de búsqueda diferente, o el navegador sin interfaz aún no había cargado al completo el JavaScript de Kiwi antes de que se capturara la consulta GraphQL). En vez de intentar reconstruir que salió mal, la elección más segura fue considerar dicho día como poco fiable y descartarlo.

En segundo lugar, el archivo `MAD-BCN/2026-08-07/2026-04-21.txt` estaba vacío al momento de la consolidación. Como es evidente, dicho día se produjo el scrapeo, pero no se logró capturar la respuesta válida dejando el archivo abierto, pero sin contenido. Los archivos vacíos se consideran observaciones faltantes y se gestionan junto con los otros diez días ya faltantes, tal como se describe en la *sección 5.3.2*.

5.3 PREPROCESAMIENTO

5.3.1 DETECCIÓN Y TRATAMIENTOS DE VALORES ATÍPICOS

La detección de los valore atípicos es crucial y fue posible gracias a la utilización del método del rango intercuartílico (Páez & Boisjoly, 2022). Consiste en que para una serie de precios dados cualquier valor que se encuentre por debajo de Q1 – K·IQR o por encima de Q3 + K·IQR se marca como valor atípico. Donde el valor *k* determina el grado de exigencia del criterio. Como recomendación estándar, se suele usar *k=1,5* pero el valor idóneo depende de loque se requiera eliminar en cada situación, que en nuestro caso fue diferente para cada base de datos. Concretamente, cuando un valor se marca como atípico, se sustituye interpolando linealmente entre sus dos vecinos no atípicos más próximos dentro de la misma serie. El código de Python completo del preprocesamiento está incluido y explicado en el *Anexo II, Fase 2*.

En el caso del conjunto de datos A, se empleó un *k=1,5*. Debido a que cada punto es una fecha de salida diferente consultada en una única sesión, por lo que cada observación es

objetivamente independiente y el umbral estándar encaja a la perfección. En cambio, para el caso del dataset B se utilizó un *k=3,0*. Los motivos de un *k* mayor en este caso recaen en que este conjunto de datos realiza un seguimiento del mismo vuelo a lo largo del tiempo algunas aerolíneas fluctúan los precios en los últimos días antes del vuelo con parte de la gestión estándar de ingresos. Si fuese con un *k=1,5*, marcaría esos picos posteriores como valores atípicos y los cortaría, eliminando el comportamiento real de los precios en lugar de ruido. En cambio, con un *k=3,0*, únicamente se eliminan los valores que realmente son extremos (errores tipográficos en los datos de origen, fallos en el scraping o descargas puntuales). Todo ello que se parezca a la fijación dinámica de precios habitual de las aerolíneas si se mantiene.

5.3.2 TRATAMIENTO DE LOS DATOS FALTANTES

Como se trató en secciones anteriores, el dataset B tiene aproximadamente diez días en las que no se realizó ningún rastreo web:

- 11 de abril: aislado, un solo día
- Del 15 al 19 de mayo, y los días 21, 22 y 24 de mayo: una semana caótica de problemas con el servidor.
- 4 de junio: aislado, cerca del final del estudio.

Estos casos suponen de forma aproximada el 17% de la ventana de observación de 59 días. Con un intervalo consecutivo más largo del 15 al 19 de mayo, siendo el más crítico desde el punto de vista de la predicción ya que cinco días son suficientes para que se produzcan fluctuaciones por parte de las aerolíneas Enel precio de los billetes de avión. Estos valores se complementan mediante interpolación lineal en cada serie, lo cual es una aproximación fiable para intervalos cortos peor no tanto para el intervalo largo de mayo. Se decidió aceptar esta limitación en vez de intentar algo más avanzado (interpolación spline, imputación basada en modelos…). También, debido a que los intervalos se consideran muy cortos como para usar modelos más sofisticados.

Con la finalidad de comprobar en cuanto desvía la interpolación en el análisis, se realizó una prueba de sensibilidad sobre los resultados ARIMA. Como medida se reajustaron los modelos sobre la serie original conservando los valores NaN, para ello se utilizó el filtro de Kalman nativo de SARIMAX. Se concluye que la diferencia del RMSE entre estos dos enfoques está por debajo del 5% para 3 de las cuatro rutas estudiadas (MAD-BCN, MAD-IST, MAD-LHR). Para el caso de MAD-BER se llegó a alcanzar el 15%, el cuál es alto como para marcarlo como alerta de la ruta, aunque es normal ay que MAD-BER presenta una mayor varianza de referencia. Dicho análisis queda plasmado en el Anexo C.

5.3.3 DIVISIÓN ENTRE ENTRENAMIENTO, VALIDACIÓN Y PRUEBA

En este proyecto se han considerado todas las divisiones temporales, dado que una división aleatoria es un error común que hay que evitar al trabajar con series temporales, ya que filtraría datos futuros al entrenamiento. Para el caso del conjunto de datos A la división seleccionada es 80% entrenamiento y 20% test sin conjunto de validación. ARIMA se encarga de seleccionar su orden mediante el AIC a la parte de entrenamiento, por lo que no precisa de un conjunto de retención. En cambio, para el dataset B se decantó por un uso aproximado de 60% entrenamiento, 10% validación y 30% test calculado por serie en vez de por los datos agrupados. Concretamente si tenemos unos 49 puntos por serie, esto da lugar aproximadamente a 35 puntos de entrenamiento, 6 de validación y 17 de prueba para cada uno aplicados de forma independiente para cada serie.

5.4 DISEÑO DE CARACTERÍSTICAS

Como se vio anteriormente, los datos brutos tienen tres columnas significativas para el modelado: el precio, la fecha de recopilación y la fecha de salida. El resto de los datos como la compañía aérea o el número de escalas se registraron para completar la información pero no serán utilizados por ningún modelo del estudio. A raíz de estas tres columnas se construye un conjunto de características previas a la llegada de los datos al modelo LSTM y XGBoost. Para el caso de ARIMA, funciona de manera directa sobre la serie de precios e ignora todo

lo demás pro el diseño. Cinco de las características que acaban en la entrada del modelo son las expuestas en la *tabla 12*.

Características	Tipo	Descripción	Escalado
Price	Continuo	Precio diario del billete en euros	MinMax[0,1] por serie
Days_until	Continuo	Días naturales desde la fecha de captura hasta la salida MinMax[0,1] por serie	MinMax[0,1] por serie
Dow_sin	Continuo	sen(2π·día de la semana / 7)	MinMax[0,1]
Dow_cost	Continuo	cos(2π·día de la semana / 7)	MinMax[0,1]
Route_*	Binario	**One-hot**[27] de las cuatro rutas (4 columnas)	Ninguna

Tabla 12 Características utilizadas por LSTM y XGBoost.

Con la finalidad de explicar con más detenimiento alguna de ellas, cabe destacar la variable `days_until`. Dicha variable existe debido el funcionamiento real de la fijación de precios de las aerolíneas. La investigación sobre la gestión de la rentabilidad (Talluri & Van Ryzin, 2006b) lleva años documentando que los precios de los billetes de avión tienden a aumentar a medida que se acerca la fecha de salida. Comienzan baratos cuando se abre la venta y suben a medida que se llenan los asientos, y suelen dispararse en la última semana. Esta variable permite que el modelo no tenga que deducir, basándose solamente en la serie de precios, en qué punto de la curva se encuentra cada observación que es muy complejo cuando la curva de reservas son diferentes para cada vuelo. Por ello, gracias a dicha variable el modelo obtiene la posición directamente.

[27] One-hot, es una técnica de codificación de variables categóricas en la que cada categoría se representa mediante una columna binaria independiente, tomando valor 1 cuando la observación pertenece a dicha categoría y 0 en caso contrario

Otra variable que conviene explicación es la causa de por qué los días de la semana se codifican como un par seno-coseno en vez de como números enteros o *one-hot*. Esto es debido a que el lunes (0) y el domingo (6) son contiguos en la vida real, un vuelo el lunes y uno el domingo tiene un comportamiento más similar que un vuelo el lunes y otro el miércoles. Entonces, si la codificación es simplemente de enteros, el modelo vería que el lunes y el domingo como los días más separados (0 y 6), pero la realidad es lo contrario. Por tanto, para conservar la estructura circular de la semana, asignar el día de la semana a un punto en la circunferencia unitaria con $(\sin(2\pi \cdot d/7), \cos(2\pi \cdot d/7))$ es lo correcto. Permitiendo que el domingo se situé justo al lado del lunes en la circunferencia.

Por otro lado, se ha discutido que las rutas se codifican mediante one-hot. Dentro de una serie esta columna es constante por construcción (una serie pertenece a una ruta), pero varía entre series cuando el LSTM y el XGBoost se entrenan con las 28 series agrupadas. Esta codificación habilita a los modelos a aprender el comportamiento de cada ruta sin tener que entrenar cuatro modelos separados.

Centrando la atención en las decisiones técnicas, es preciso explicarlas ya que afectan a los resultados. La primera de ellas se refiere al escalado, el *MinMaxScaler* se ajusta por serie únicamente en la parte de entrenamiento y a las cuatro características continuas. Las columnas binarias de ruta se concatenan sin modificar tras el escalado. Dentro de una misma serie las columnas son constantes (1 para una columna, 0 para las otras tres) y este ajuste asigna a cualquier columna constante como 0,0. Esto resultó que todas las columnas de ruta fueran cero en todas partes, eliminando discretamente la información de ruta a la entrada del modelo. El error se detectó durante las pruebas, donde tanto el LSTM como el XGBoost se estaban entrenando con datos equivalentes a tres rutas sin manera de diferenciarlas. Dicho error se corrigió en el paso de escalado separando las características continuas y categóricas, y es por ello que las métricas posteriores a la corrección son notablemente mejores.

La segunda decisión es referida al objetivo. El modelo no predice el precio absoluto, sino que predice la diferencia entre el último día observado y el día objetivo expresada en el espacio escaldo [0,1] de esa serie. De hecho, el objetivo para el horizonte h y el precio

escalado en el momento *t (s(t))* es $y = s(t+h) - s(t-1)$, donde *t-1* es el último día de la ventana de entrada. Por lo que la predicción del precio real se reconstruye como:

$$\text{predicted\_price} = \text{naive\_price} + \text{delta\_predicted} \cdot \text{price\_range}$$

Donde *price_range* es el rango aprendido por el escalador en el entrenamiento. Dicha reformulación plantea el problema de nuevo. En vez de que el modelo prediga un precio y supere al naive desde cero, el modelo le pide que prediga una corrección frente al naive. Un delta previsto de cero recupera el ingenuo de manera exacta.

Por tanto, para superar al naive, el modelo únicamente necesita aprender una estructura en los incrementos, no en los niveles de precios absolutos. Como hemos visto anteriormente, la versión previa del proceso, el objetivo era el precio absoluto donde los modelos predecían valores sesgados con respecto al modelo base y los ratios MASE estaban entre 1,5 y 1,8. En cambio, tras proponer el objetivo delta, estos ratios descendieron hasta situarse de manera aproximada entre 1,00 y 1,05 en horizontes de uno a tres días. Esto nos indica que los modelos actuales operan cerca de la frontera del modelo ingenuo en lugar de muy pro debajo de ella. La programación completa del objetivo y las características está mostrada en el Anexo B.

5.5 MODELOS IMPLEMENTADOS

5.5.1 ARIMA

Los fundamentos teóricos de ARIMA (media móvil integrada autorregresiva) se discutieron anteriormente en el capítulo 2, *sección 2.4*. La implementación utilizada `pmdarima.auto_arima` (Hyndman & Khandakar, 2008), que se encarga de automatizar la metodología de Box-Jenkins buscando entre los órdenes candidatos (p, d, q) y seleccionando una combinación capaz de minimizar el criterio de información de Akaike. Antes de la exploración, es preciso ejecutar las pruebas de Dickey-Fuller aumentada y KPSS en cada una de las series para determinar el orden de diferenciación *d* idóneo. Concretamente, para la mayoría de los casos el *d=1* fue suficiente.

Merece la pena hacer hincapié en como varía esta implementación para cada uno de los conjuntos de datos. En primer lugar, para el dataset A se ajusta un modelo ARIMA independiente para cada una de las seis rutas. Con una división en 80% entrenamiento y 20% test, lo que equivale a aproximadamente 158 puntos de entrenamiento y 39 de test por ruta. En la práctica, `auto_arima` seleccionó modelos con términos autorregresivos o de media móvil para las seis rutas: tres con orden (5,1,0), una con (2,1,0), una con (1,0,1) y una con (0,1,1). Sin embargo, en cuatro de ellas (BCN, BER, LHR y SIN) las predicciones resultantes coincidieron con la previsión *naïve* hasta el quinto decimal: el patrón escalonado dejó tan poca varianza que los coeficientes AR ajustados no tuvieron nada que aprovechar, y la previsión se redujo al último valor observado. En las dos rutas restantes (CDG e IST), los modelos sí incluyeron términos AR o MA activos, pero obtuvieron resultados ligeramente peores que la referencia *naïve* en el conjunto de prueba.

En cuanto al conjunto de datos B, ARIMA se ajusta a las cuatro rutas disponibles, una por ruta, calculadas como el precio medio diario a lo largo de las siete fechas de vuelo. Se decantó por una división 70/30 proporcionando unos 41 puntos de entrenamiento y 18 de prueba por ruta. En este caso, los cuatro modelos ajustados superan los diagnósticos de residuos estándar, el Ljung-Box para la autocorrelación y Shapiro-Wilk para la normalidad, lo que quiere decir que los modelos están correctamente diseñados.

A sí mismo, se barajó el modelo SARIMA, pero se acabó descartando, debido a que con solo 49 puntos útiles y una estacionalidad semanal potencial de periodo 7 no es suficiente como para estimar contundentemente los términos estacionales AR y MA. Los pipelines del ajuste ARIMA para ambos conjuntos y el diagnóstico de los residuos están plasmados en el *Anexo II, Fase 3A*.

5.5.2 LSTM

La parte teórica del modelo LSTM esta explicado en el capítulo 2, *sección 2.6*. Con respecto a la arquitectura utilizada en este proyecto, la red neuronal presenta cuatro capas: un único bloque con 32 unidades ocultas, seguido de una capa de dropout con una tasa de 0,2, una capa densa con 8 unidades y activación ReLU, y por último, una capa densa final con una

única salida lineal. El número total de parámetro ronda los 5500. Con aproximadamente 780 ventanas de entrenamiento en las 28 series agrupadas, el modelo más pequeño se aproxima más a la relación adecuada entre capacidad y datos. Cabe nombrar que se empleó una iteración anterior que utilizada una arquitectura más amplia (LSTM 64 → LSTM 32 → Densa 16) con más de 31000 parámetros, pero resultó ser más irregular en sus curvas de validación.

Capa	Forma de salida	Parámetros
LSTM (32)	(lote, 32)	5.248
Dropout (0,2)	(lote, 32)	0
Densa (8, ReLU)	(lote, 8)	264
Densa (1, linear)	(lote, 1)	9
Total		**5.521**

Tabla 13 Arquitectura LSTM y recuento de parámetros.

La entrada a la red es una ventana deslizante de siete días consecutivos, cada uno representado por el vector de características de ocho dimensiones descrito en la *sección 5.4*. La ventana se desliza hacia adelante un día cada vez, produciendo una muestra de entrenamiento por posición por serie. Con 28 series de aproximadamente 35 puntos de entrenamiento utilizables cada una (tras restar la longitud de la ventana y la división), el conjunto de entrenamiento contiene alrededor de 780 ventanas. El conjunto de validación contiene aproximadamente 140 ventanas, y el conjunto de prueba varía según el horizonte. Esta información está ampliada en la *sección 5.6*.

El entrenamiento utiliza el optimizador Adam (Kingma & Ba, 2017) con la tasa de aprendizaje predeterminada, el error cuadrático medio como pérdida y un tamaño de lote de 32. El número máximo de epoch es 300, pero el entrenamiento rara vez dura tanto: una función de llamada de parada temprana con un valor de paciencia de 20 supervisa la pérdida

de validación y restaura los pesos de la mejor época cuando la validación deja de mejorar. La capa de dropout (Srivastava et al., 2014) está activa durante el entrenamiento y se desactiva silenciosamente en el momento de la inferencia, lo cual es su comportamiento estándar en Keras.

La reproducibilidad se garantiza fijando la semilla aleatoria al inicio de cada ejecución de entrenamiento de un horizonte, con numpy y tensorflow inicializados con *42 + horizonte*. Esto significa que la semilla para *h=1* difiere de la semilla para *h=3*, *h=7* y *h=14*, lo que evita la situación observada en una versión anterior en la que una única inicialización desafortunada produjo un resultado atípico para *h=7*. Las operaciones de TensorFlow en la CPU siguen presentando un pequeño grado de no determinismo que ninguna semilla puede controlar por completo, por lo que cabe esperar variaciones menores entre ejecuciones (del orden del uno o dos por ciento en las métricas).

Como se puede apreciar en la *ilustración 22* en las que se muestran las curvas de pérdida de entrenamiento y validación para el modelo de horizonte 1. La razón por la que la red neuronal está regularizada correctamente para este volumen de datos es que ambas curvas descienden sin desviaciones, y la diferencia entre ellas se mantiene estrecha en todo momento. La implementación completa de LSTM, así como la construcción de la ventana de entrada se encuentra descrita en el *Anexo II, Fase 3B*.

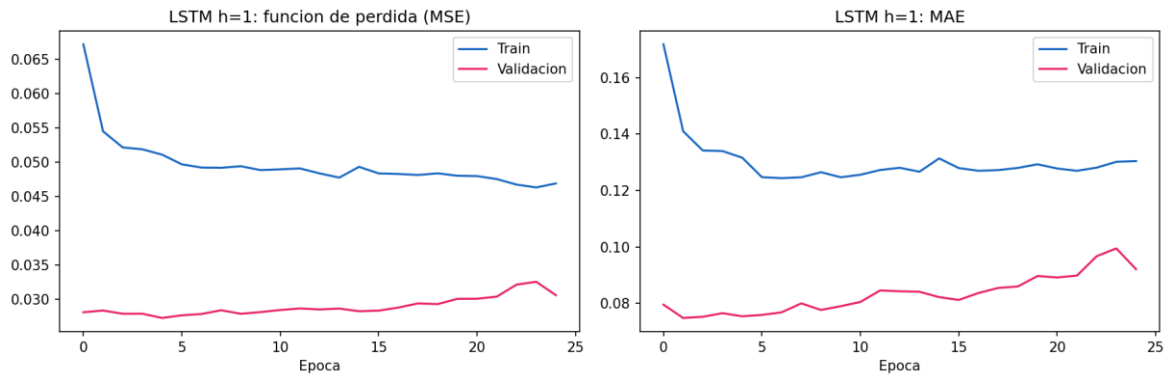

Ilustración 22 Curva pérdida de entrenamiento y validación para horizonte 1

5.5.3 XGBOOST

XGBoost se ha incluido como referencia complementaria más que como un modelo central. La motivación de su uso recae en que XGBoost no tiene concepto de secuencia (trata cada muestra como una fila tabular diferente), por lo que comparar su rendimiento con el de LSTM ayuda a determinar si la estructura temporal de los datos realmente aporta algo. Por lo que, si un modelo no secuencial funciona mejor que uno secuencial, en si mismo nos aporta datos informativos.

XGBoost consiste básicamente en la técnica del gradient boosting, formulado originalmente por Friedman. La idea central es entrenar una secuencia de pequeños arboles de decisión, donde cada nuevo árbol se ajusta a los errores residuales del conjunto ya construido en ese momento. A continuación, las predicciones de todos los árboles se suman para producir el resultado final y el procedimiento tiende en un indicador de bajo sesgo sin necesidad que ningún árbol individual sea profundo. Este modelo añade las regulaciones L1 y L2 al objetivo de la construcción de árboles, además de que reduce la varianza submeustreando las columnas y reduce la contribución de cada árbol mediante un parámetro de tasa de aprendizaje. Estos motivos son los que hace a este modelo considerablemente más robusto que el simple gradient boosting, y es por ello por lo que se convirtió en estándar como modelo de aprendizaje automático a finales de la década de 2010.

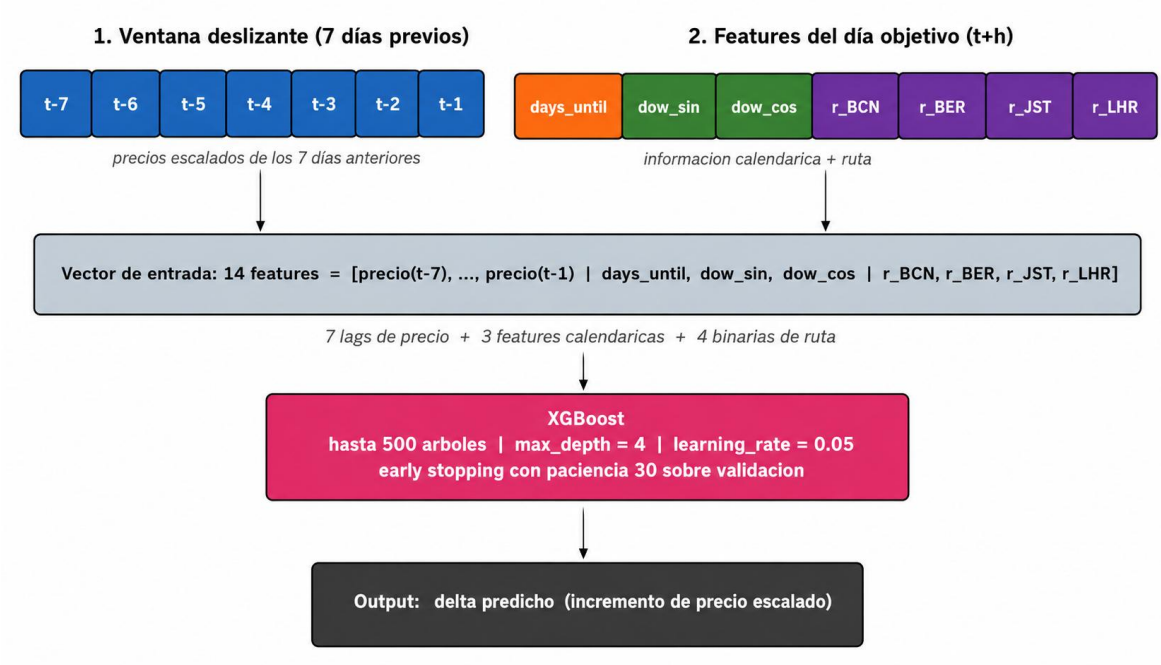

Ilustración 23 Construcción del vector de entrada al modelo XGBoost. Cada muestra de entrenamiento se obtiene aplanando la ventana de 7 días previos junto con las features calendáricas y de ruta del día objetivo.

En cuanto a su arquitectura, cabe destacar que es diferente al LSTM ya que este último consume tensor tridimensional (muestras, intervalos de tiempo, características). En cambio, XGBoost toma una matriz bidimensional en la que cada una de las filas es un vector de características plano. En nuestro caso, para cada muestra este vector contiene los siete precios escalados, y desfasados de la venta de entrada, el valor de *days_until*, la codificación cíclica del día de la semana en la fecha objetivo y la codificación de ruta *one-hot*. Mientras que XGBoost ve directamente la información del calendario para el día objetivo (*days_until* y *dow* en el día en el que se está prediciendo), en el modelo LSTM solo se ve los siete días de la ventana de entrada son acceso directo al calendario del día objetivo. Para un horizonte de un día, esa asimetría es pequeña, pero para el horizonte de 14 días, el modelo XGBoost tiene una ventaja notable. Se decidió mantener esta comparativa tal cual y documentar la limitación de la asimetría en lugar de intentar equilibrar las entradas de forma artificial.

Hiperparámetro	Valor
n_estimators (max)	500
max_depth	4
learning_rate	0,05
subsample	0,8
colsample_bytree	0,8
early_stopping_rounds	30
random_state	42

Tabla 14 Hiperparámetros de XGBoost.

El proceso de entrenamiento utiliza la detención temprana en el conjunto de validación. El número de árboles tienen un límite de 500, aunque el tamaño real para el conjunto se selecciona en función del MSE que suele estar por debajo del límite. En cuento a los valores de los hiperparámetros, no se han ajustado de manera significativa ya que no es la parte central de este proyecto (secuencial frente a no secuencial o clásico frente a ML). Son valores razonables para un conjunto de datos pequeño con alrededor de 780 muestras de entrenamiento. La implementación de XGBoost se encuentra en el *Anexo II, Fase 3B*.

5.6 MARCO DE EVALUACIÓN

5.6.1 LA PREDICCIÓN INGENUA COMO REFERENCIA

Los fundamentos teóricos de la predicción ingenua y su relación con la hipótesis de paseo aleatorio se discuten en el capítulo 2, *secciones 2.2* y *2.5*. A continuación, se explica por qué esta es la referencia principal respecto a la que se evalúan los modelos y como se calcula en la práctica dicha predicción ingenua.

Para el conjunto de datos B, la predicción ingenua se calcula en cada observación por serie, usando el precio del último día de la ventana de entrada:

$$\hat{y}_{naive}(t+h) = y(t)$$

Esto provocó varios errores en versiones anteriores, en una de ellas agrupaba las predicciones por ruta y desplazaba todo el grupo una posición para calcular el modelo ingenuo, mezclando información de vuelos distintos e incrementando el error aparente. La corrección a esos modelos que parecían artificialmente competitivos consiste calcular el modelo ingenuo estrictamente dentro de cada serie (ruta, `fecha_vuelo`), es este el procedimiento usado en todos los resultados expuestos en el *capítulo 6*.

Cabe destacar que, en lugar de evaluar frente, por ejemplo, la media intrameustrada o un modelo ingenuo estacional, se decantó por evaluar frente al modelo ingenuo. En la fijación de precios de los billetes de avión, la hipótesis nula del paseo aleatorio es la referencia teórica (discutida en el *capítulo 2*), por lo que la comparación con la previsión de persistencia es la prueba empírica de si el mercado es eficiente en el horizonte que se está midiendo. Un modelo que no supera al ingenuo no ha aprendido nada que el precio más reciente no codifique ya.

5.6.2 MÉTRICAS UTILIZADAS EN ESTA TESIS

Las cuatro métricas definidas en el *capítulo 2.5* (MAE, RMSE, MAPE y una ratio escalada con respecto al ingenuo) se calculan para cada modelo en cada horizonte. Únicamente la cuarta requiere un comentario adicional, ya que la definición utilizada aquí difiere ligeramente del MASE estándar.

La métrica que aparece a lo largo del *capítulo 6* como Ratio MAE/Naive se calcula como:

$$Ratio = \frac{MAE_{modelo}}{MAE_{naive}}$$

Donde ambos valores de MAE se calculan en el mismo horizonte sobre las mismas observaciones de prueba. Además, de que un valor inferior a 1,0 significa que el modelo supera al ingenuo.

Esta métrica se diferencia del MASE estándar que fija el denominador en el MAE ingenuo de un paso de las muestras de los datos de entrenamiento. Esto permite, dentro del mismo horizonte, contestar a preguntas como si el modelo supera al ingenuo en $h=7$. Pero implica que una relación de 1,02 en $h=14$ y una relación de 1,02 en $h=1$ no representan la misma diferencia de rendimiento absoluta. Por ello, la pregunta principal de este trabajo es si cada modelo supera al ingenuo en cada horizonte específico, no cómo crece el error absoluto con el horizonte. Para evitar confusiones con la definición de Hyndman-Koehler, la métrica se denomina Ratio MAE/Naive en todo el texto, en vez de MASE.

5.6.3 EL DISEÑO MULTIHORIZONTE

Todos los modelos se evalúan en cuatro horizontes: uno, tres, siete y catorce días por delante. Una alternativa podría ser la predicción recursiva (entrenar un modelo $h=1$ y aplicar de forma iterativa) pero con series de solo 50 o 60 puntos, el error acumulado habría sido dominado por $h=7$. Por ello, se decantó por qué cada horizonte corresponde a una ejecución de entrenamiento independiente con su propia instancia de modelo, siguiendo la estratega de previsión directa en múltiples pasos.

La motivación para probar múltiples horizontes es que la ventaja del modelo ingenuo no es constante en todos ellos. En $h=1$, copiar el precio de ayer es realmente competitivo porque los mercados rara vez se mueven mucho durante la noche. Para $h=14$, el último precio observado tiene dos semanas de antigüedad, y un modelo que haya aprendido algo sobre tendencias o estructura a largo plazo debería tener más margen para obtener mejores resultados. Si esto ocurre realmente, y si varía automáticamente según la ruta, es la pregunta que el *capítulo 6* responde en detalle.

6 ANÁLISIS DE RESULTADOS

6.1 ARIMA EN EL CONJUNTO DE DATOS A: EL EFECTO ESCALERA

Como se ha visto previamente, el conjunto de datos A consta de seis series de precios interpoladas diariamente de Madrid (BCN, VER, CDG, IST, LHR, SIN) cada una de ellas con 395 observaciones procedentes de 197-198 puntos brutos muestreados cada dos días durante una única sesión de Playwright contra el endpoint final GraphQL de Kiwi.com. El orden del modelo se determina automáticamente mediante `auto_arima`. Se opta por una división 80/20, quedando a disposición 316 días de entrenamiento y 79 días de prueba por ruta.

6.1.1 ANÁLISIS DE CARDINALIDAD Y COMPORTAMIENTO DE AUTOCORRELACIÓN

Como se estudió en la *sección 5.2.2*, cada ruta presenta entre 18 y 21 precios distintos a lo largo de unas 200 observaciones y un rango de estabilidad de unos 10 días. Por ello, el efecto del escalón (staircase) es visualmente notable en la *ilustración 24*.

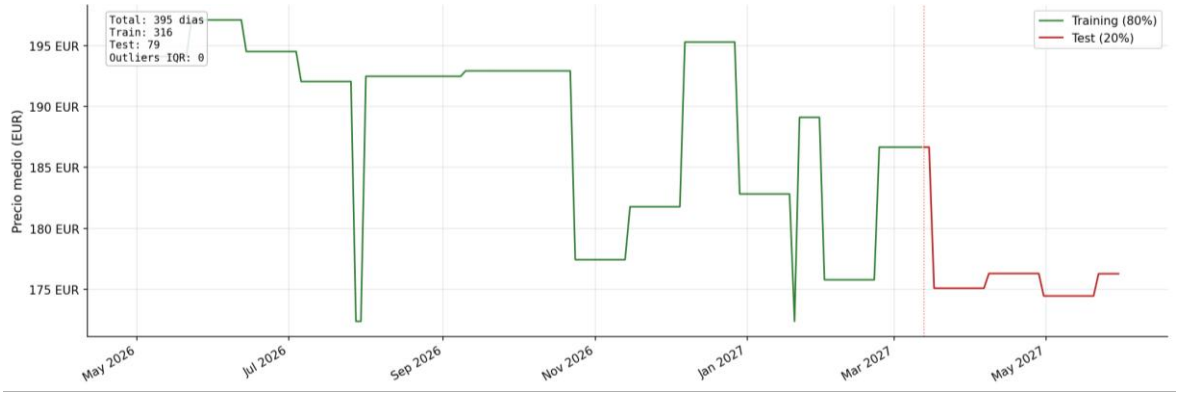

Ilustración 24. Serie original MAD-LHR.

6.1.2 ARIMA VS. *NAIVE*

Ruta	Orden ARIMA	RMSE ARIMA	RMSE Naive	Mejora	Veredicto
MAD-BCN	(5, 1, 0)	10.53	10.53	+0.0 %	No supera al naive
MAD-BER	(5, 1, 0)	4.91	4.91	+0.0 %	No supera al naive
MAD-CDG	(1, 0, 1)	4.48	4.30	−4.1 %	No supera al naive
MAD-IST	(0, 1, 1)	4.55	4.52	−0.8 %	No supera al naive
MAD-LHR	(5, 1, 0)	11.02	11.02	+0.0 %	No supera al naive
MAD-SIN	(2, 1, 0)	5.92	5.92	+0.0 %	No supera al naive

Tabla 15. Comparación ARIMA vs. naive en el conjunto de test (n=79).

ARIMA cómo se puede apreciar no supera al modelo base de referencia en ninguna de las rutas, pero esto tiene la justificación anterior descrita del efecto del escalón que apenas muestra desviaciones en los precios. Cuatro de las seis rutas (BCN, BER, LHR, SIN) coinciden con el RMSE del modelo base hasta el quinto decimal: los modelos ajustados se resumen en una previsión constante que es igual al último valor del entrenamiento (133,41, 147,42, 186,67 y 549,39 EUR respectivamente), que es lo que genera la previsión *naive.* Las otras dos rutas (CDG y IST), son los dos únicos destinos donde `auto_arima` designó con términos AR o MA, pero que incluso llegaron a dar peor resultado que el modelo de referencia. Esto sugiere, que fuera cual fuera la señal qué el modelo, no fue lo bastante fuerte

como para compensar el ruido en los datos escalonados. La *ilustración 25* muestra la previsión de MAD-BCN superpuesta a la línea base (*naive*).

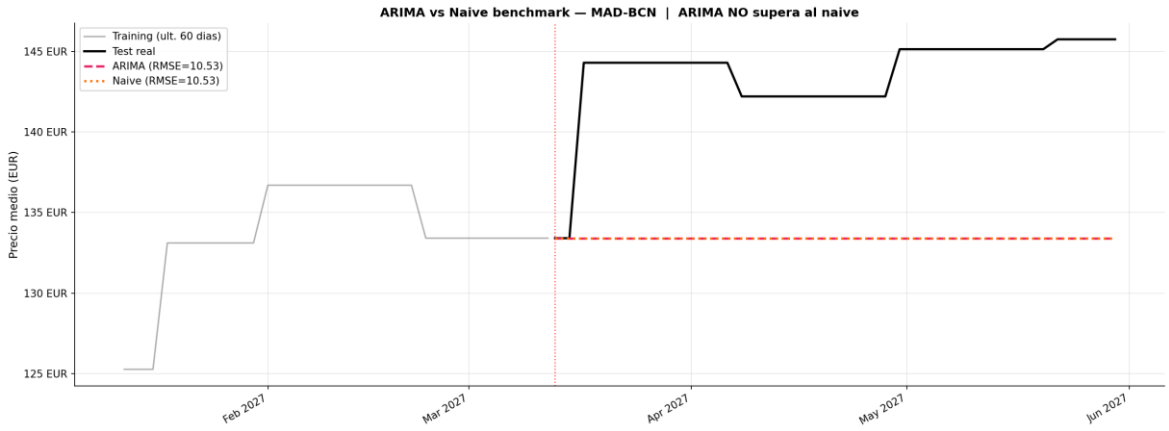

Ilustración 25. Predicción ARIMA(5,1,0) frente a la predicción ingenua en MAD-BCN (horizonte = 79 días)

Cabe destacar que los seis modelos superan la prueba de Ljung-Box (sin autocorrelación residual). Pero en cambio, ninguno supera la prueba de Shapiro-Wilk lo que concuerda con los saltos discretos en los datos escalonados. Los p valores exactos se recogen en el *Anexo II, Fase 3C*.

6.2 *ARIMA CON EL CONJUNTO DE DATOS B*

Las cuatro rutas de este conjunto de datos con origen en Mad y destino: BCN, BER, IST y LHR. Se analizan de manera diaria con 59 observaciones cada una, y una división 70/30 (41 de entrenamiento y 18 de prueba) y la selección del orden mediante `auto_arima`. Mediante la consolidación se aplicó la eliminación de valores atípicos (factor IQR 3,0) y la interpolación lineal para los días faltantes. En el apartado 6.3 se examina si dicha interpolación distorsiona los resultados.

6.2.1 MAD-BCN

Ambas estadísticas ADF=-2,50, p=0,117 y KPSS= 0,16, p=0,10 son pruebas que discrepan. Es por ello, que `auto_arima` selecciona ARIMA(2,0,0) con AIC= 107,87 y BIC=114,72.

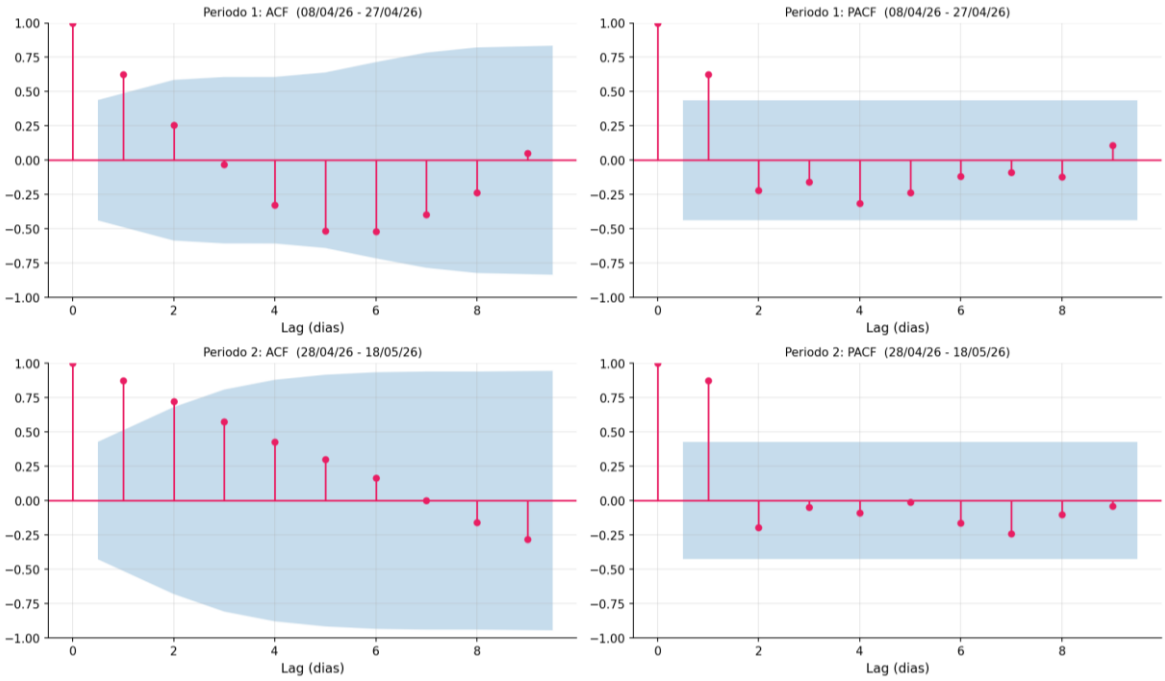

Ilustración 26. ACF y PACF de la serie MAD-BCN

Ljung-Box con retrasos de 5 y 10 da p=0,93 y p=0,996; Shapiro-Wilk da W=0,975, p=0,51. Ambas pruebas son exitosas como se puede apreciar en la ilustración X. De hecho, es la única ruta en este conjunto cuyos residuos satisfacen tanto la independencia y la normalidad.

Ilustración 27. Diagnóstico de residuos del modelo ARIMA(2,0,0) para MAD-BCN.

El RMSE en el conjunto de prueba es de 5,10 frente a 6,04 de la referencia básica, lo que supone una mejora del +15,52 % (véase *Figura 27*). La especificación AR(2) capta una ligera deriva descendente durante la ventana de prueba que la previsión base pasa por alto.

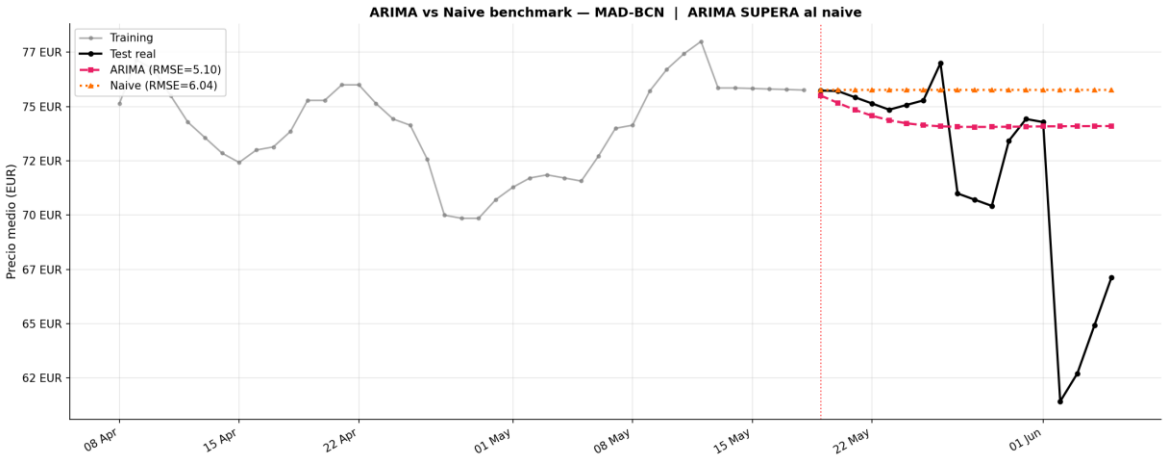

Ilustración 28. Predicción ARIMA(2,0,0) frente al benchmark base (18 días de prueba) para MAD-BCN.

Como se puede apreciar en la siguiente *ilustración 28*, esta muestra la misma previsión con intervalos de confianza del 80 % y del 95 %. La serie real permanece dentro del intervalo del 80 % durante los primeros diez días y sale por debajo de él durante la caída final del precio, cuya magnitud no anticipa la previsión puntual.

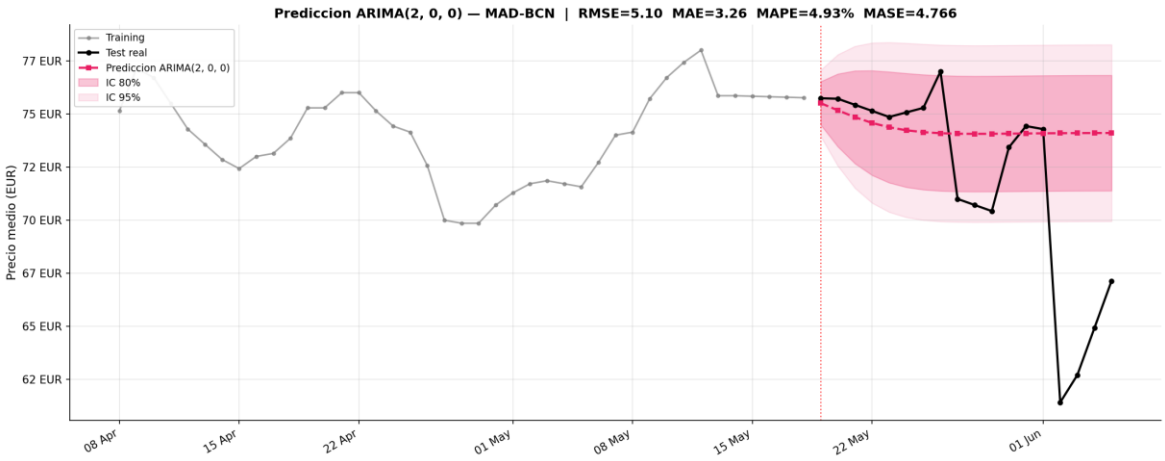

Ilustración 29. Predicción ARIMA(2,0,0) con intervalos de confianza del 80 % y del 95 % para MAD-BCN.

En conclusión, se puede afirmar que MAD-BCN es la única ruta del conjunto de datos B en la que ARIMA supera a la previsión básica, y la única con diagnósticos de residuos limpios. Estos dos hallazgos se refuerzan mutuamente: la estructura AR(2) es estadísticamente válida y económicamente útil en esta ruta.

6.2.2 MAD-BER

Estadística ADF = −2,06, p = 0,261; estadística KPSS = 0,49, p = 0,044. Ambas pruebas rechazan la estacionariedad. `Auto_arima` selecciona ARIMA(0,1,0) con AIC = 180,28 y BIC = 181,97. Esta especificación es el paseo aleatorio: una sola diferencia sin términos AR ni MA, equivalente por construcción a la predicción base en el conjunto de prueba.

Ljung-Box en los retrasos 5 y 10 da p=1,000 en ambos casos; Shapiro-Wilk da W=0,279, p<0,001 (véase *ilustración 30*). Los residuos son independientes, pero marcadamente no normales, con una distribución dominada por un pequeño número de grandes saltos que sitúan la estadística de Shapiro-Wilk muy por debajo de la normalidad.

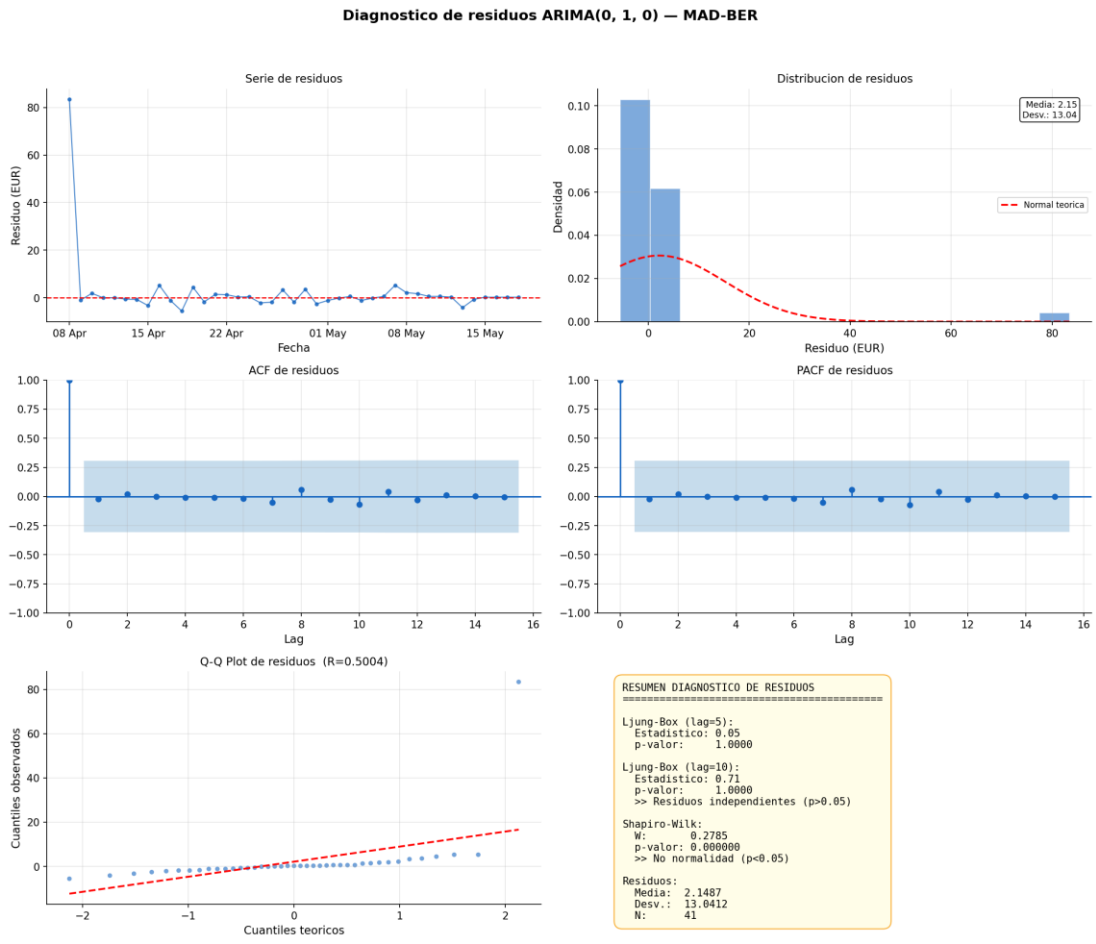

Ilustración 30. Diagnóstico de residuos del modelo ARIMA(0,1,0) para MAD-BER.

El RMSE en el conjunto de prueba es de 3,91, idéntico a la referencia *naive* de 3,91, lo que no supone una mejora sustancial (*Ilustración 31*). La predicción puntual es una línea horizontal en el último valor de entrenamiento (~87,5 EUR), mientras que la serie real asciende de forma constante hasta el rango de 90-95 EUR.

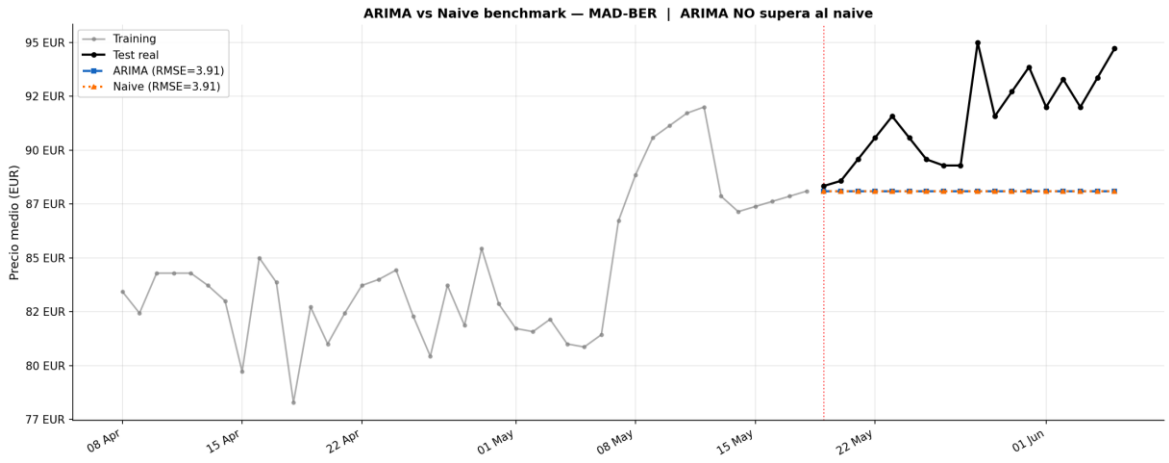

Ilustración 31. Predicción ARIMA(0,1,0) frente al benchmark base (18 días de prueba) para MAD-BER.

MAD-BER es el caso teórico de la hipótesis del paseo aleatorio: `auto_arima` selecciona por sí mismo el paseo aleatorio como el mejor modelo entre todos los candidatos, y la predicción resultante coincide punto por punto con la línea de referencia ingenua. El modelo no ha fallado frente al benchmark, sino que ha convergido con él.

6.2.3 MAD-IST

Estadística ADF = −2,01, p = 0,281; estadística KPSS = 0,20, p = 0,10. Las dos pruebas no coinciden, lo que refleja el diagnóstico dudoso de MAD-BCN. `auto_arima` selecciona ARIMA(2,0,0) con AIC = 223,22 y BIC = 230,07.

Ljung-Box en los retrasos 5 y 10 da p=0,86 y p=0,98; Shapiro-Wilk da W=0,976, p=0,52 (*Ilustración 32*). Ambas pruebas de residuos son satisfactorias, con un gráfico Q-Q que se sitúa a lo largo de la línea de referencia y un histograma que sigue de cerca la distribución normal teórica.

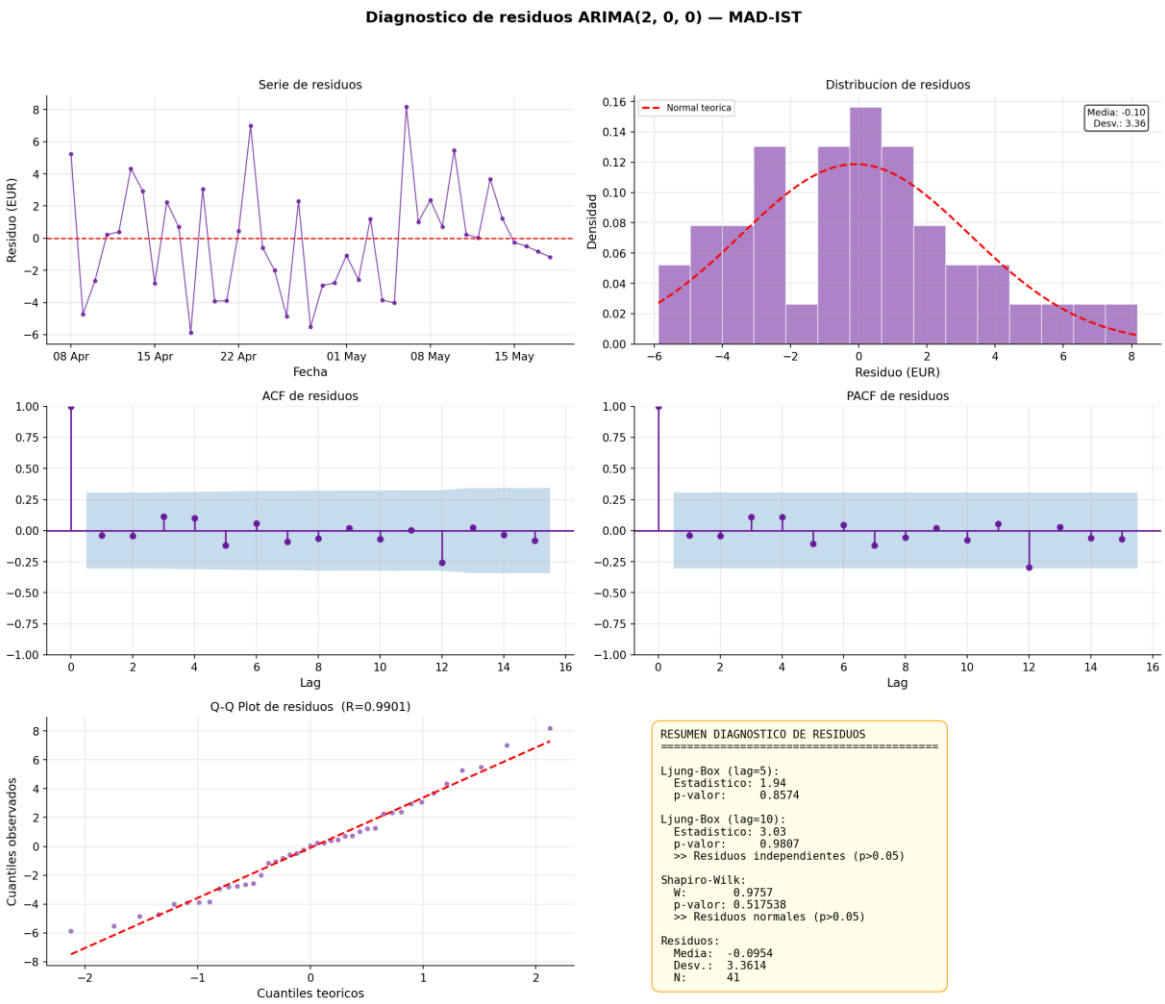

Ilustración 32. Diagnóstico de residuos del modelo ARIMA(2,0,0) para MAD-IST.

El RMSE en el conjunto de prueba es de 4,91 frente a una referencia ingenua de 4,86, lo que supone un empeoramiento del −1,03 % (Ilustración X). La previsión AR(2) se mantiene prácticamente estable en torno a los 124 EUR, mientras que la serie real oscila entre 121 y 135 EUR, alcanzando un pico pronunciado al final de la ventana de prueba.

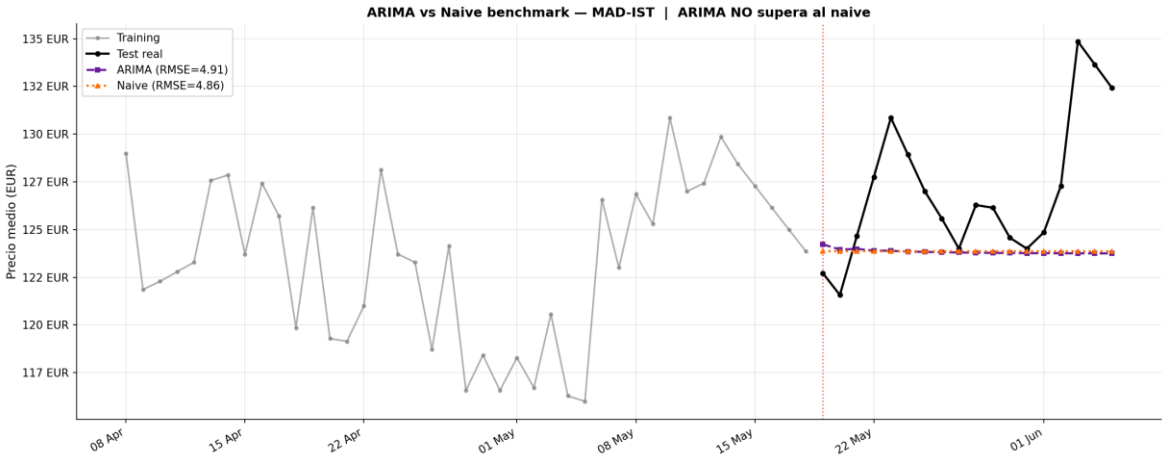

Ilustración 33. Predicción ARIMA(2,0,0) frente al benchmark base (18 días de prueba) para MAD-IST.

MAD-IST se sitúa entre los dos casos anteriores: `auto_arima` encuentra una estructura AR(2) con residuos limpios y normales, pero la estructura es demasiado débil para traducirse en una ventaja predictiva sobre la línea de base ingenua. El modelo está correctamente especificado, pero no es lo suficientemente informativo como para superar la persistencia.

6.2.4 MAD-LHR

- Estacionariedad. Estadística ADF = −1,50, p = 0,536; estadística KPSS = 0,76, p = 0,010. Ambas pruebas coinciden en indicar no estacionariedad.
- Modelo. ARIMA(0,1,1), AIC = 190,49, BIC = 195,55.
- Residuos (*Ilustración 34*). Ljung-Box con retrasos de 5 y 10 da p=1,000; Shapiro-Wilk da W=0,265, p<0,001. Independientes, pero no normales, como en MAD-BER.
- Previsión. Prueba RMSE = 13,34 frente a una referencia ingenua de 9,55, lo que supone un deterioro del −39,6 % (véase *ilustración 35*).

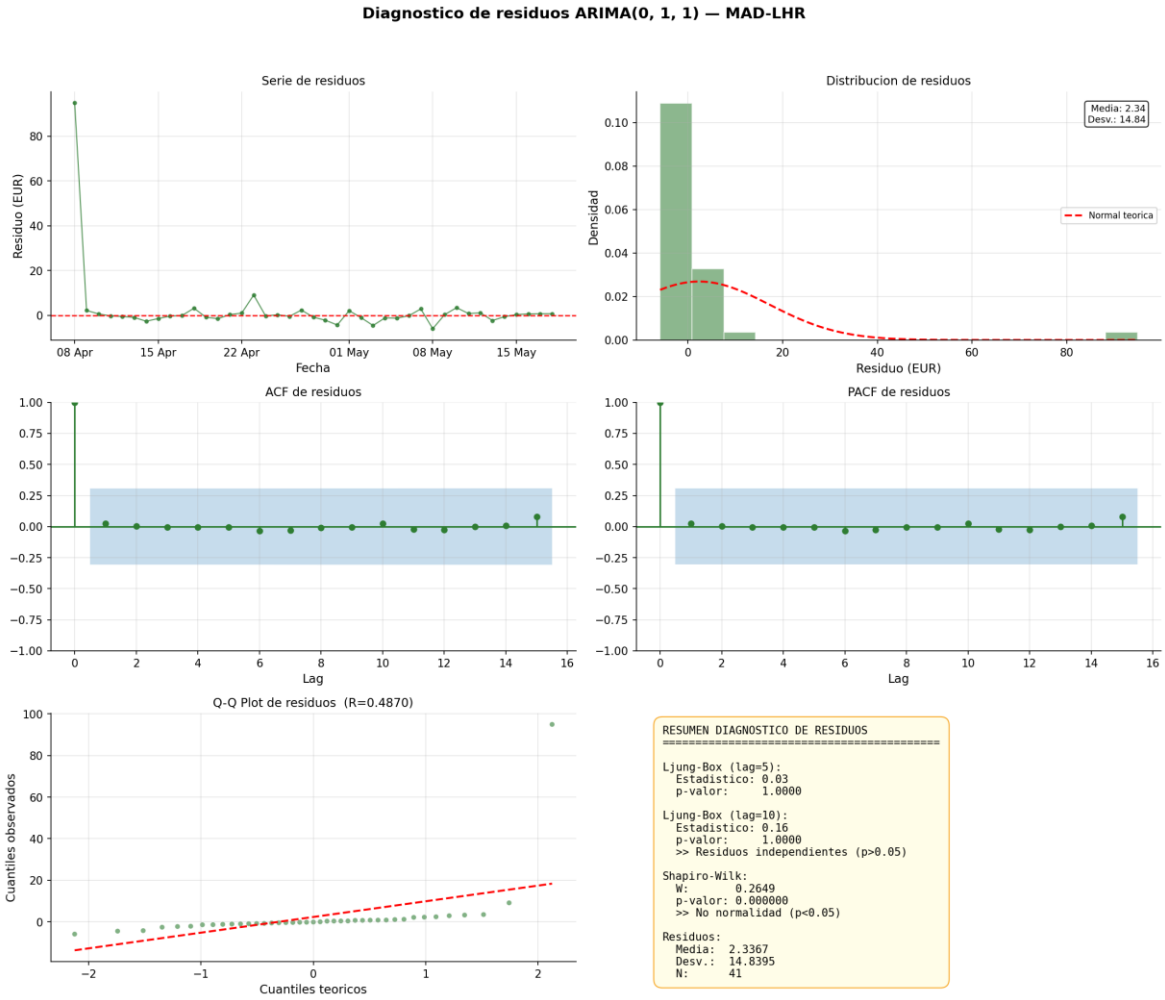

Ilustración 34. Diagnóstico de residuos del modelo ARIMA(0,1,1) para MAD-LHR.

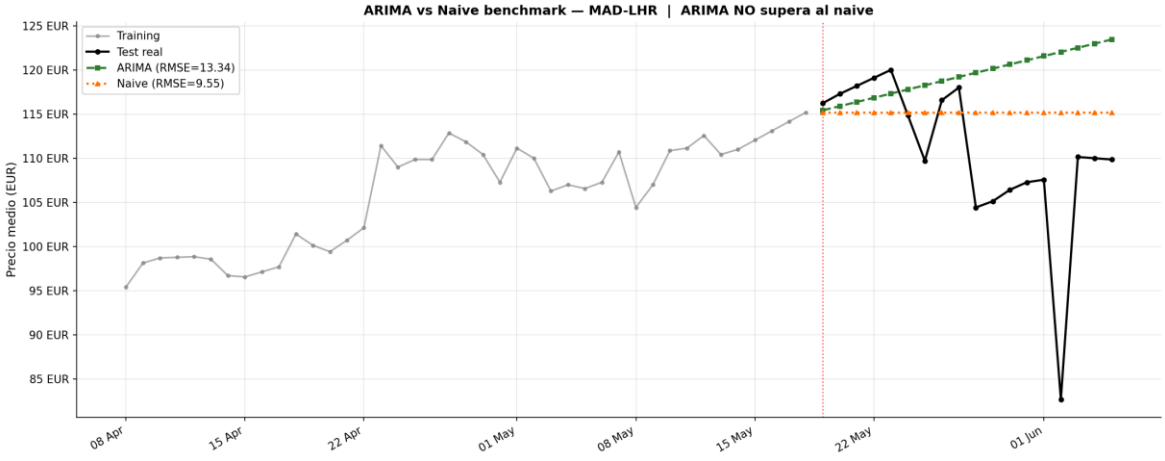

Ilustración 35. Predicción ARIMA(0,1,1) frente al benchmark base (18 días de prueba) para MAD-LHR.

MAD-LHR es el peor caso del conjunto de datos. El término MA(1) proyecta la última tendencia de entrenamiento al alza en la ventana de prueba precisamente cuando la serie real comienza un descenso sostenido, lo que convierte la apuesta direccional del modelo en una penalización del 40 % en el RMSE frente a la previsión base.

6.2.5 RESUMEN

Ruta	Orden ARIMA	AIC	RMSE ARIMA	RMSE Naive	Mejora	LB	SW	Veredicto
MAD-BCN	(2, 0, 0)	107.87	5.10	6.04	+15.5 %	✓	✓	Supera al naive
MAD-BER	(0, 1, 0)	180.28	3.91	3.91	+0.0 %	✓	✗	No supera (es el naive)
MAD-IST	(2, 0, 0)	223.22	4.91	4.86	−1.0 %	✓	✓	No supera al naive
MAD-LHR	(0, 1, 1)	190.49	13.34	9.55	−39.6 %	✓	✗	No supera al naive

Tabla 16. Resumen ARIMA + benchmark naive para las cuatro rutas de Dataset B (test = 18 días).

Tres de las cuatro rutas confirman el dominio del paseo aleatorio en el horizonte de 18 días: Berlín selecciona explícitamente el paseo aleatorio como el mejor modelo ARIMA, Estambul pierde por muy poco a pesar de unos residuos limpios, y Londres pierde por un 40 % debido a una apuesta direccional que va en contra de los datos de prueba. Barcelona es el único destino en la que la metodología de Box-Jenkins produce una señal aprovechable, con ambos diagnósticos de residuos en concordancia. Si esta señal se generaliza a arquitecturas más flexibles o se desvanece como un efecto de una serie concreta es la cuestión que aborda el *apartado 6.4* con el modelaje de LSTM y XGBoost.

6.3 SENSIBILIDAD A LA INTERPOLACIÓN LINEAL

Las cuatro series del conjunto de datos B contienen 10 días interpolados cada una (el 16,9 % de la ventana de 59 días). En esta sección se cuantifica en qué medida dicha interpolación influye en los resultados ARIMA del *apartado 6.2*, comparando dos modelos en cada ruta.

El modelo A es el ARIMA del apartado 6.2, ajustado a las series interpoladas. El modelo B es del mismo orden, ajustado mediante SARIMAX a las series sin procesar, con los huecos NaN gestionados internamente por el filtro de Kalman sin imputación. Si A y B coinciden, la interpolación es positiva; si no coinciden, está contribuyendo al resultado.

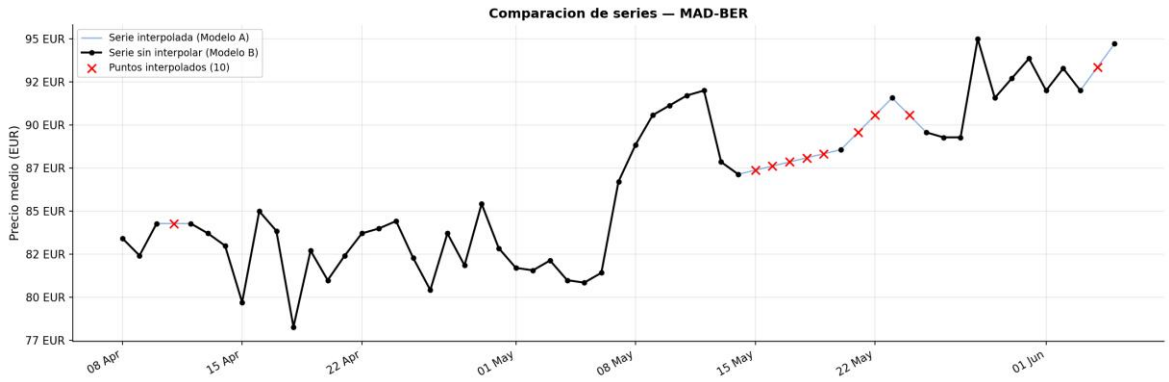

Ilustración 36. Serie MAD-BER con los 10 puntos interpolados marcados con cruces rojas. Las cruces señalan los días en que el scraper falló durante la campaña.

Ruta	Orden	AIC A	AIC B	RMSE A	RMSE B	Diff RMSE	Corr. pred.	Veredicto
MAD-BCN	(2, 0, 0)	107.87	94.66	5.10	5.51	7.9 %	0.759	ACEPTABLE
MAD-BER	(0, 1, 0)	180.28	159.55	3.91	5.09	30.1 %	—	ATENCIÓN
MAD-IST	(2, 0, 0)	223.22	190.25	4.91	5.09	3.8 %	−0.814	ACEPTABLE
MAD-LHR	(0, 1, 1)	190.49	163.09	13.34	9.31	30.2 %	—	ATENCIÓN

Tabla 17. Comparación entre el modelo ARIMA sobre la serie interpolada (A) y el mismo orden estimado por SARIMAX sobre la serie sin interpolar (B).

Dos observaciones que caben mencionar antes de analizar cada serie por separado. En primer lugar, el AIC mejora de A a B en las cuatro series: la probabilidad es mayor en la serie sin procesar que en la interpolada. Esto no es de extrañar, ya que la interpolación lineal aplana la varianza local y un modelo ARIMA tiene que compensar eso con valores suavizados artificialmente. En segundo lugar, la mejora del AIC no se traduce automáticamente en una

mejor predicción en el conjunto de prueba. Esa distinción es el tema central de lo que viene a continuación.

6.3.1 CASOS SON PROBLEMAS METODOLÓGICOS: MAD-BCN Y MAD-IST

En MAD-BCN, los dos modelos producen una trayectoria de previsión prácticamente idéntica. Ambas predicciones AR(2) descienden suavemente desde unos 75 EUR hasta unos 74 EUR a lo largo del periodo de prueba de 18 días. El RMSE difiere en un 7,9 % (5,10 frente a 5,51) y la correlación entre los dos vectores de predicción es de 0,759, la más alta de las cuatro rutas. La conclusión del apartado 6.2.1 (ARIMA supera al modelo ingenuo en un +15,5 %) se mantiene independientemente de si las lagunas se interpolan o se dejan como NaN: el modelo capta la misma deriva descendente en ambos casos.

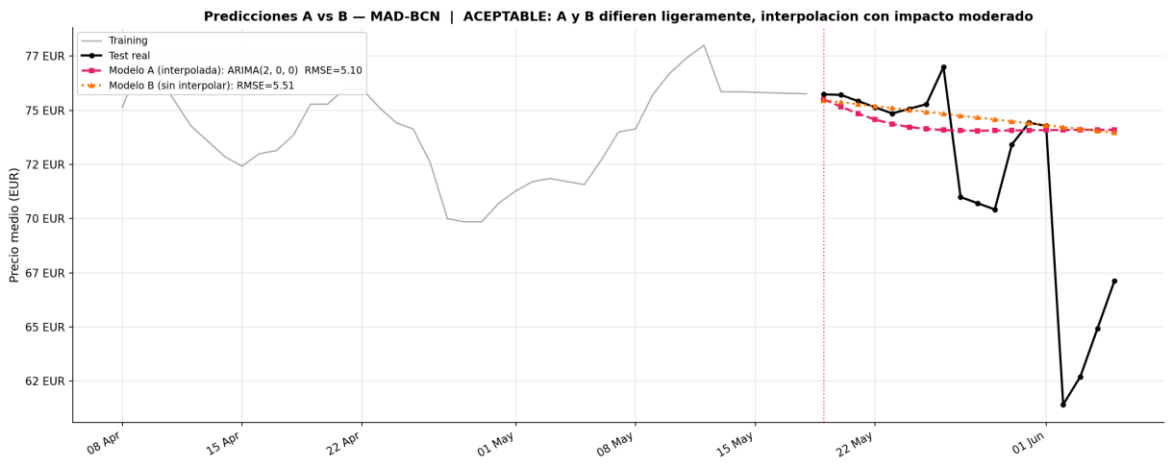

Ilustración 37. Predicciones A frente a B en MAD-BCN. Las dos curvas se mantienen prácticamente superpuestas a lo largo del horizonte de prueba.

En MAD-IST, la diferencia en el RMSE es aún menor (3,8 %, 4,91 frente a 5,09), pero la correlación entre las predicciones A y B es fuertemente negativa (−0,814). Los dos modelos giran en torno a niveles similares (~123 EUR para A, con un ligero aumento para B), pero discrepan en cuanto a la dirección de la deriva. El resultado final en el RMSE es el mismo porque la serie real oscila entre 121 y 135 EUR y se encuentra aproximadamente a igual distancia de ambas trayectorias de previsión. Ninguno de los modelos supera la referencia

ingenua, y la conclusión metodológica es la misma que en el *apartado 6.2.3*: la interpolación no es la causa del bajo rendimiento.

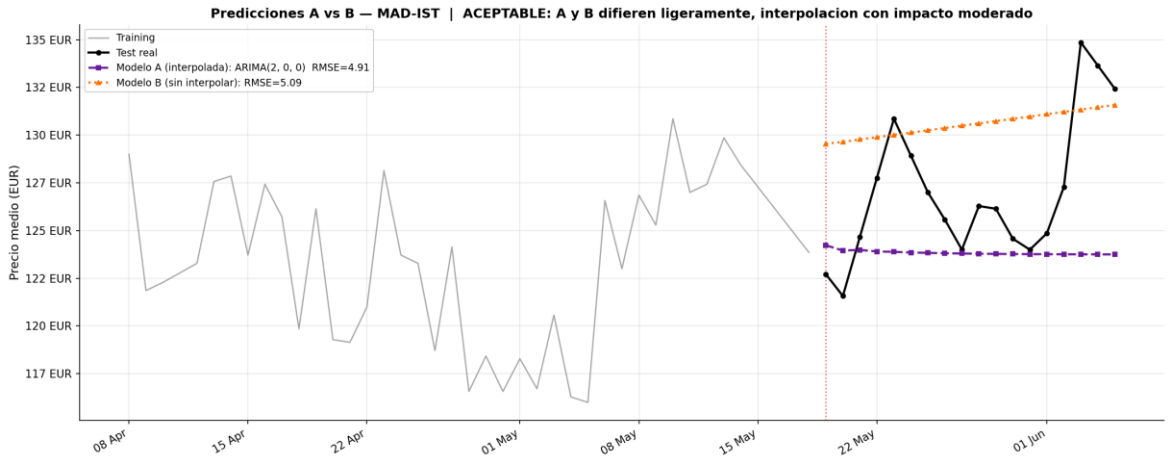

Ilustración 38. Predicciones A frente a B en MAD-IST. Ambas trayectorias se mantienen próximas al último valor de entrenamiento, pero con pendientes de signo opuesto.

Las dos trayectorias aceptables comparten una característica común: la serie se mantiene más o menos estacionaria en nivel durante la ventana de prueba, y el relleno lineal entre los valores observados no contradice el patrón general. En ese régimen, sustituir unos pocos segmentos por NaN no cambia lo que el modelo puede aprender.

6.3.2 CASOS QUE REQUIEREN ATENCIÓN: MAD-BER Y MAD-LHR

En MAD-BER, A es el paseo aleatorio (ARIMA(0,1,0)) y produce una previsión plana en el último valor de entrenamiento, ~87,5 EUR. El modelo B tiene la misma especificación sobre los datos brutos, pero el filtro de Kalman recupera una estimación del estado terminal ligeramente diferente, ~86,7 EUR. Ambas previsiones son líneas planas y ambas se desvían de la serie real, que asciende al rango de 90–95 EUR. La diferencia del 30,1 % en el RMSE se debe a la pequeña desviación entre las dos líneas planas, amplificada por el error sistemático de la prueba, y no a dinámicas divergentes. La correlación no está definida porque ambas predicciones son constantes.

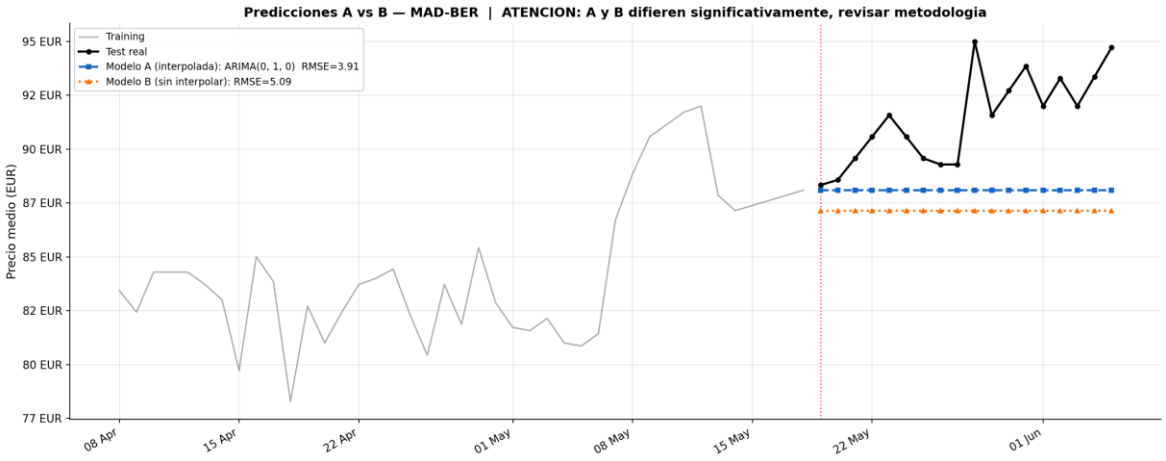

Ilustración 39. Predicciones A frente a B en MAD-BER. Ambas líneas son constantes con un desfase aproximado de 0,8 EUR. La serie de prueba asciende de manera sostenida y ningún modelo la captura.

MAD-LHR es el caso más informativo. El modelo A (ARIMA(0,1,1) sobre la serie interpolada) proyecta una tendencia al alza, alcanzando ~123 EUR al final del horizonte. El modelo B (mismo orden sobre los datos brutos) produce una previsión casi plana en ~111 EUR. La serie real entra en la prueba en ~116 EUR, se mantiene en torno a ese nivel durante unos días y luego cae por debajo de los 110 EUR. El modelo B se acerca más a ese comportamiento durante casi todo el horizonte, por lo que su RMSE (9,31) es un 30,2 % inferior al del modelo A (13,34). La correlación vuelve a ser indefinida porque B es esencialmente constante.

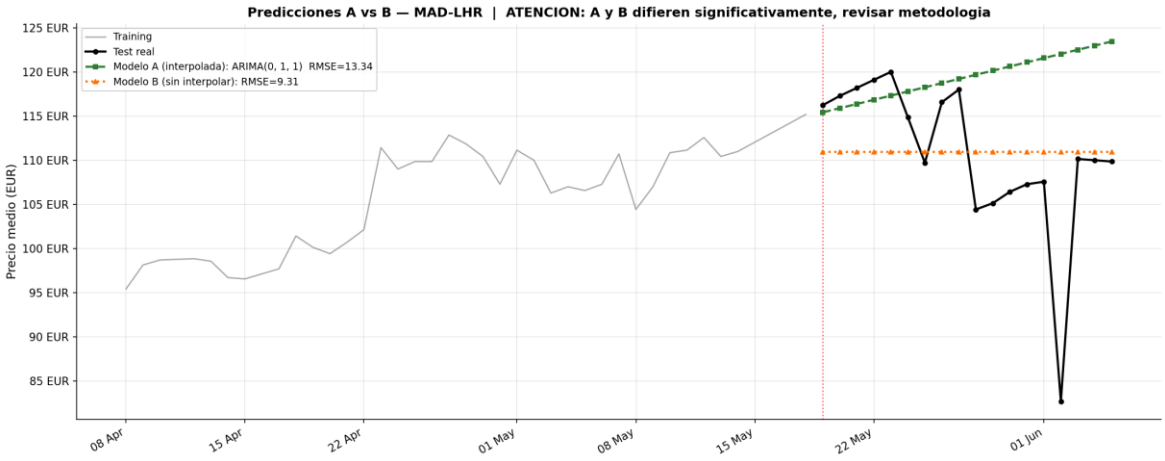

Ilustración 40. Predicciones A frente a B en MAD-LHR. La trayectoria ascendente de A se aleja de los valores reales, mientras que B se mantiene cerca del nivel inicial y termina con menor error.

El resultado de MAD-LHR es el metodológicamente relevante: la interpolación distorsionó el modelo. Los diez días interpolados incluían un tramo (15-22 de mayo) en el que el relleno lineal entre dos valores observados creó una pendiente ascendente artificial, y el término MA(1) captó esa pendiente como señal y la proyectó hacia adelante. Sin interpolación, el mismo modelo aplicado a los mismos datos solo detecta la caída abrupta que realmente se produjo y se abstiene de extrapolar una tendencia.

6.3.3 IMPLICACIONES PARA EL APARTADO 6.2

El análisis de sensibilidad no invalida el apartado 6.2, pero lo matiza. La mejora del +15,5 % de ARIMA sobre el modelo ingenuo en MAD-BCN supera la prueba sin problemas. El −1,0 % de MAD-IST también es robusto: A y B son más o menos equivalentes a la línea de base ingenua, en cualquier caso. El 0,0 % de MAD-BER es estructuralmente invariante, ya que el paseo aleatorio es el paseo aleatorio en cualquiera de las configuraciones, y lo que se pierde es la tendencia de la prueba, no la elección de la interpolación.

La ruta que requiere matices es MAD-LHR. El deterioro del −39,6 % descrito en el apartado 6.2.4 es en parte un artefacto de la interpolación: el mismo ARIMA(0,1,1) aplicado a la serie bruta produce una previsión más conservadora y una penalización menor (~13 % en lugar

de 40 %). La conclusión sustantiva se mantiene (ARIMA no supera a la interpolación ingenua en esta ruta), pero la magnitud del fallo descrito en el *apartado 6.2* está exagerada.

Estos resultados se incorporan directamente al *capítulo 7* (limitaciones y trabajo futuro). La solución estructural es una campaña de recopilación longitudinal sin lagunas; la alternativa metodológica, ajustar SARIMAX directamente sobre las series brutas, se conserva allí como un tema de trabajo futuro en lugar de incorporarse retroactivamente al apartado 6.2.

6.4 LSTM Y XGBOOST EN EL CONJUNTO DE DATOS B

Se evalúan otros dos modelos en el conjunto de datos B, junto con los resultados de ARIMA del apartado 6.2. El LSTM es una red recurrente de una sola capa con 32 unidades (≈5 500 parámetros), entrenada para cada horizonte de previsión con detención temprana. XGBoost es un modelo tabular de refuerzo de gradientes con 500 árboles y una profundidad máxima de 4, ajustado sobre características retardadas. Ambos modelos se evalúan en las mismas 28 series de fechas de vuelo individuales en cuatro horizontes (h = 1, 3, 7 y 14 días), con resultados agregados por ruta y a nivel global. La métrica de error es el MASE, normalizado de modo

que la previsión base obtenga exactamente 1: los valores inferiores a 1 indican que el modelo supera a la persistencia, mientras que los valores superiores a 1 indican que pierde. El proceso se documenta en el *Anexo II, Fase 3B*.

6.4.1 MASE GLOBAL POR HORIZONTE

Las cifras globales de MASE muestran la misma tendencia para ambos modelos, con una pequeña diferencia

en la cadencia. Tanto LSTM como XGBoost están prácticamente empatados con el modelo ingenuo en h = 1 (MASE en torno a 1,01–1,02) y convergen hacia él en h = 3, donde LSTM iguala al modelo ingenuo con una precisión de una milésima. Más allá de ese horizonte, los dos modelos divergen al alza: en h = 7 ambos obtienen puntuaciones entre 1,13 y 1,15, y en

h = 14 LSTM ha subido hasta 1,22, mientras que XGBoost se mantiene ligeramente más contenido en 1,14. El patrón cualitativo, visible en la Figura 6.17, es coherente con la hipótesis del paseo aleatorio: cuando los precios siguen un paseo casi aleatorio, la previsión ingenua es difícil de superar en horizontes cortos, y cualquier modelo que intente extrapolar más allá de eso paga una penalización estructural.

h	Modelo	MAE	RMSE	MAPE	MASE
1	Naive	4.36	9.16	4.68 %	1.000
1	LSTM	4.41	9.12	4.74 %	1.010
1	XGBoost	4.45	9.11	4.77 %	1.020
3	Naive	7.29	11.86	7.79 %	1.000
3	LSTM	7.30	11.84	7.84 %	1.000
3	XGBoost	7.40	11.87	7.94 %	1.015
7	Naive	8.94	12.78	9.71 %	1.000
7	LSTM	10.24	13.87	11.06 %	1.145
7	XGBoost	10.09	13.53	10.98 %	1.128
14	Naive	11.00	15.01	12.02 %	1.000
14	LSTM	13.42	17.53	14.67 %	1.220
14	XGBoost	12.54	16.17	13.51 %	1.140

Tabla 18. Valores de los modelos en los distintos horizontes

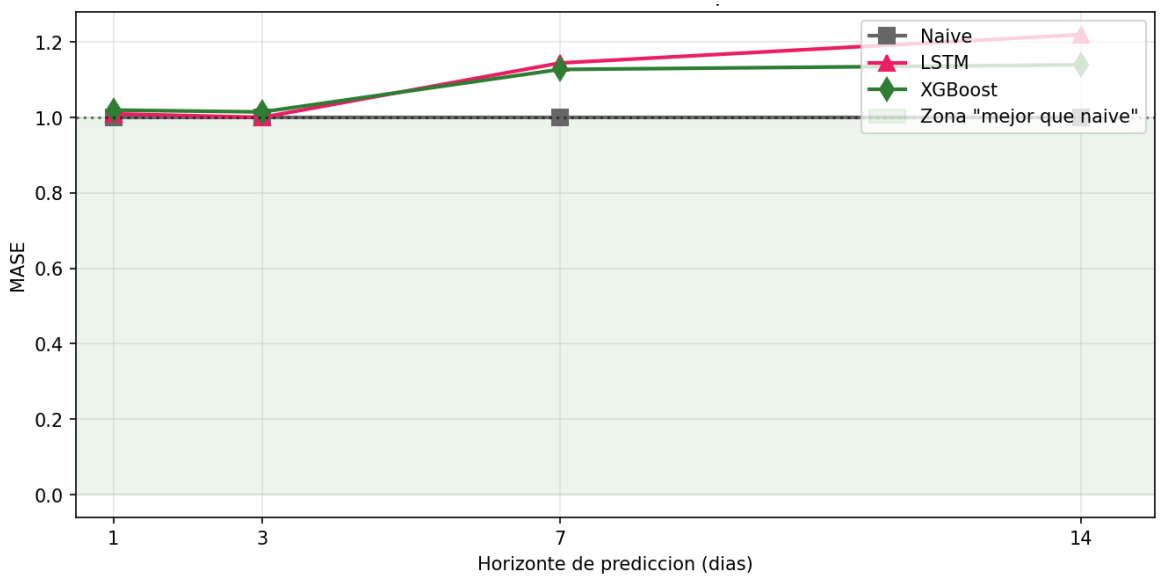

Ilustración 41. MASE global de LSTM y XGBoost frente al benchmark base (línea horizontal en MASE = 1) para los cuatro horizontes de predicción. El área sombreada en verde marca la región donde un modelo supera al base.

En términos generales, ninguno de los dos modelos justifica su complejidad. La pregunta interesante es si la media oculta casos en los que sí superan al modelo ingenuo, que es lo que se examina en el apartado 6.4.2.

6.4.2 PATRONES POR RUTA Y CELDAS QUE SUPERAN A LA HIPÓTESIS BÁSICA

Al desglosar por ruta y horizonte se obtienen 32 celdas (4 rutas × 4 horizontes × 2 modelos). De ellas, 7 presentan un MASE < 1 (*Tabla 19*).

Modelo	Ruta	h	MASE	Mejora sobre naive
LSTM	MAD-BCN	3	0.988	+1.2 %
LSTM	MAD-BCN	7	0.954	+4.6 %
LSTM	MAD-BER	3	0.995	+0.5 %
LSTM	MAD-IST	1	0.987	+1.3 %
LSTM	MAD-IST	3	0.974	+2.6 %
XGBoost	MAD-BCN	3	1.000*	+0.0 %*
XGBoost	MAD-BCN	14	0.995	+0.5 %

Tabla 19. Celdas donde un modelo bate al benchmark naive (MASE < 1).

** XGBoost MAD-BCN h = 3 alcanza MASE = 0.9998.*

MAD-BCN acumula 4 de las 7 victorias, LSTM obtiene 5 de las 7, y h = 3 es el horizonte más productivo, con 4 victorias. La mejor célula individual es LSTM en MAD-BCN con h = 7 (MASE = 0,954).

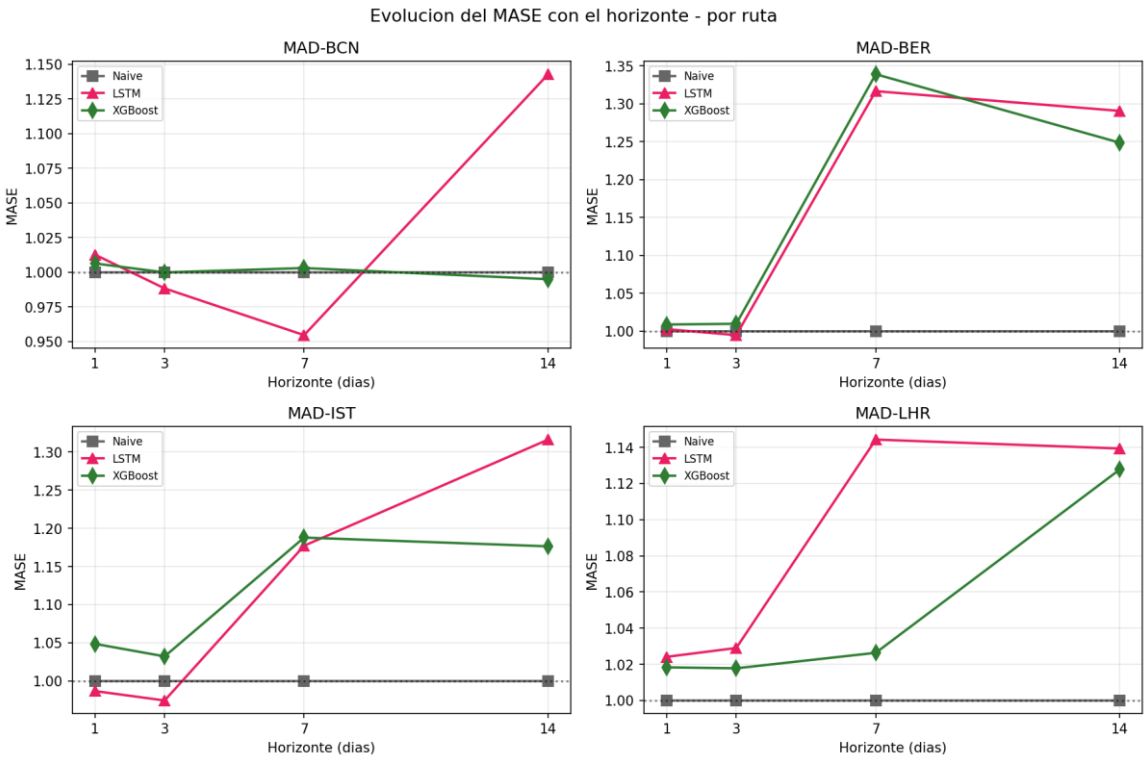

Ilustración 42. MASE de LSTM y XGBoost frente al modelo base (línea gris en 1,0) por ruta y horizonte. La línea de MAD-BCN se mantiene cerca o por debajo de 1,0; las otras tres rutas divergen claramente por encima a partir de h = 7.

El panel de MAD-BCN rompe la pauta de los otros tres, donde ambos modelos se mantienen cerca del modelo ingenuo en horizontes cortos y se elevan bruscamente por encima de él en h = 7 y h = 14. Esta es la misma ruta en la que ARIMA superó en un +15,5 % en el apartado 6.2, y la convergencia de tres métodos independientes en el mismo resultado es el resultado más sólido del capítulo.

6.4.3 DIAGNÓSTICO DEL ENTRENAMIENTO

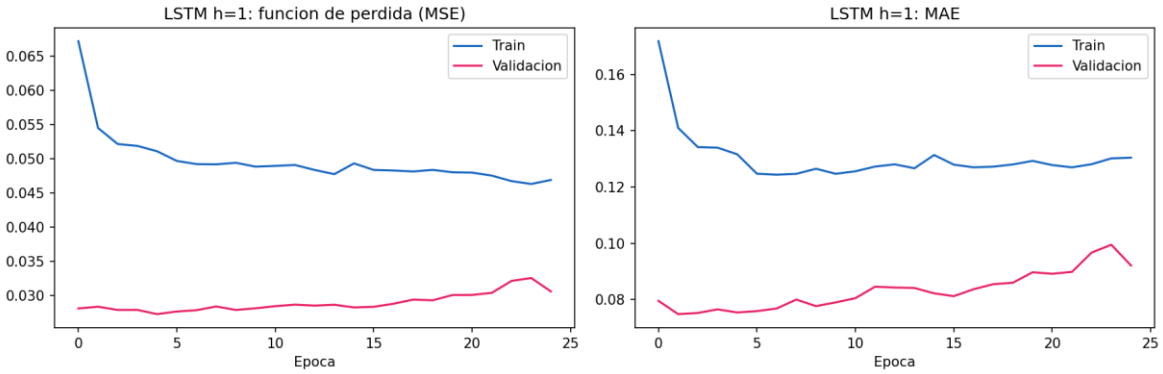

Ilustración 43. Curvas de entrenamiento del LSTM con h = 1: MSE (izquierda) y MAE (derecha) en los conjuntos de entrenamiento y validación, a lo largo de 25 epochs.

La pérdida de entrenamiento se estabiliza en torno a MSE = 0,047 tras la época 5; la pérdida de validación se desplaza al alza desde 0,027 hacia 0,032 en el mismo intervalo, lo que indica un ligero sobreajuste que la función de llamada `EarlyStopping` corta antes de que se agrave. La limitación subyacente es el tamaño de la muestra: 28 series de 41 días de entrenamiento con un *lookback* de 7 días producen solo unos pocos miles de ejemplos de entrenamiento efectivos, por debajo del régimen en el que LSTM suele dominar las líneas de base clásicas.

6.4.4 INSPECCIÓN CUALITATIVA EN H=1

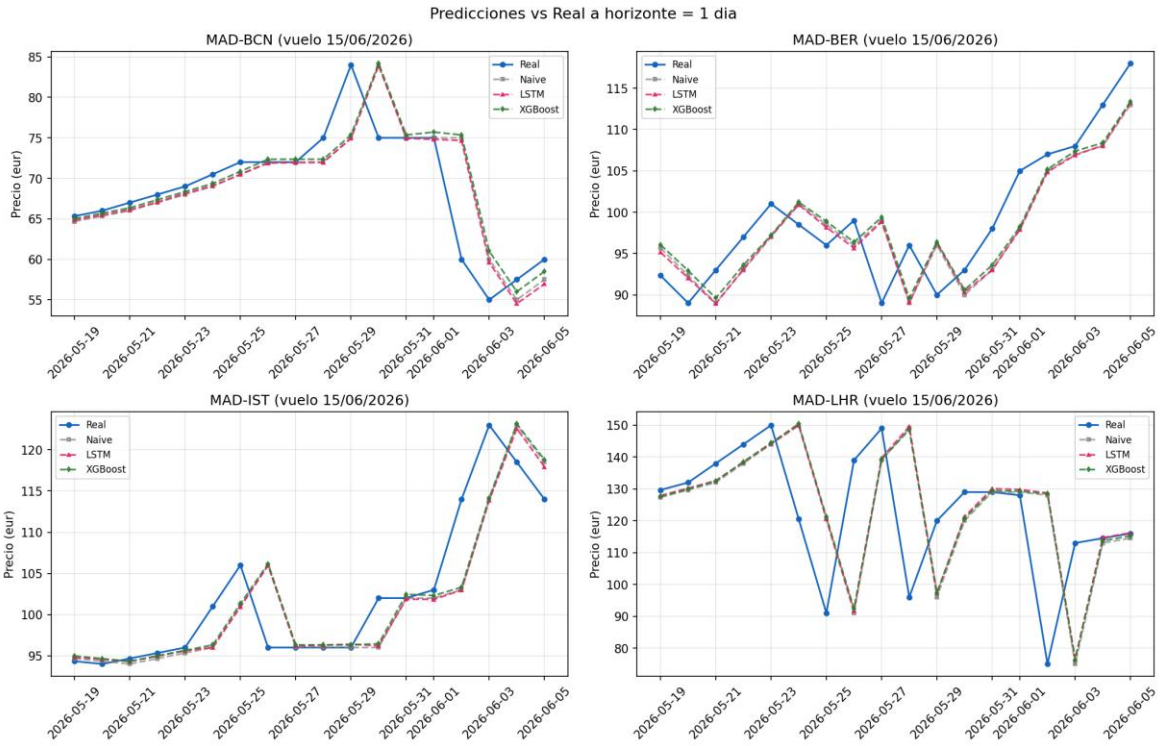

Ilustración 44. Predicciones de Naive, LSTM y XGBoost a h = 1 frente al precio real, para los cuatro vuelos del 15 de junio de 2026.

Las tres líneas de los modelos se superponen en los cuatro paneles y las tres se retrasan con respecto al precio real aproximadamente un día, lo que constituye la firma geométrica de una previsión ingenua. LSTM y XGBoost han aprendido a hacer lo que hace la predicción ingenua, con desviaciones solo marginales; en el panel MAD-LHR, altamente volátil, este retraso produce grandes errores que ningún modelo corrige.

6.4.5 RESUMEN

Ni LSTM ni XGBoost superan a la predicción ingenua a nivel global. Solo 7 de las 32 celdas por ruta y por horizonte lo superan, 4 de ellas en MAD-BCN, que así emerge de tres métodos independientes (ARIMA, LSTM, XGBoost) como la única ruta con una estructura genuinamente previsible. LSTM en h = 3 es un hallazgo secundario: indistinguible del

ingenuo a nivel global y el mejor horizonte de ML para la previsión de ventana corta en el conjunto de datos B.

6.5 EFECTO DEL DÍA DE LA SEMANA

Las 28 series individuales de fechas de vuelo se sometieron a un proceso de eliminación de tendencias en función de los días previos a la salida (metodología en el apartado 5.4), y los 1371 residuos resultantes se agruparon por día de la semana para comprobar si existía alguna regularidad calendárica en las subidas de precios.

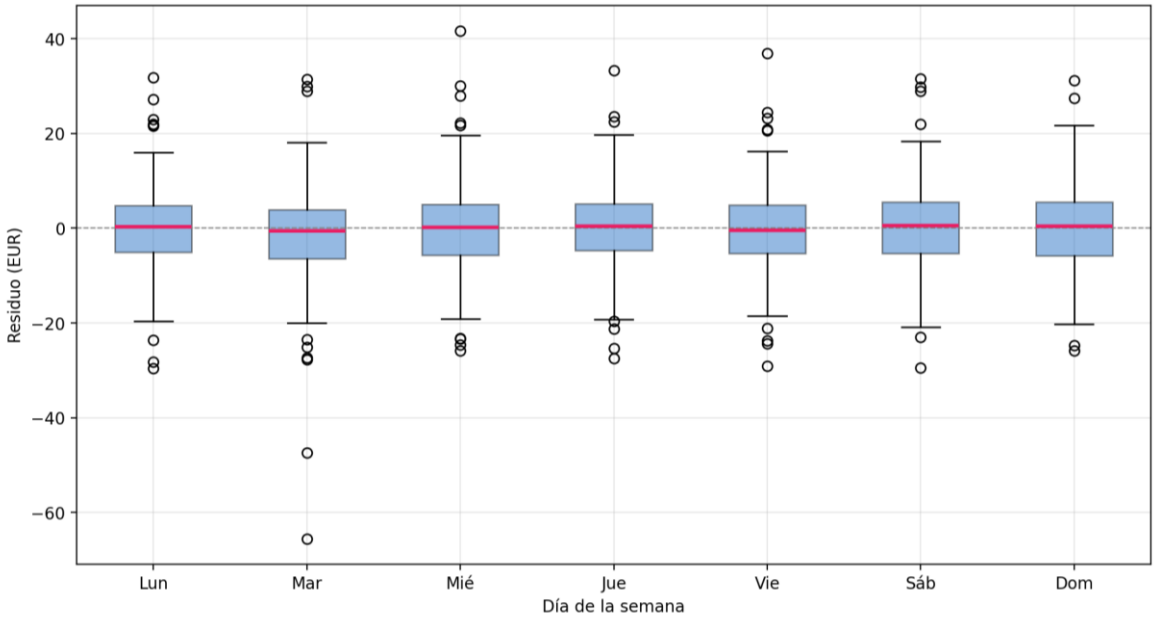

Ilustración 45. Distribución de los residuos del precio por día de la semana, sobre las 1371 observaciones reales de las 28 series del Dataset B.

Día	n	Mediana (EUR)	Media (EUR)	Desv. típica (EUR)
Lunes	196	+0.25	+0.04	9.04
Martes	195	−0.61	−1.38	11.04
Miércoles	252	+0.21	+0.21	9.31
Jueves	196	+0.42	+0.53	9.12
Viernes	196	−0.48	−0.02	9.13
Sábado	168	+0.54	+0.41	9.65
Domingo	168	+0.46	+0.24	8.96

Tabla 20.Estadísticas descriptivas de los residuos del precio por día de la semana.

Las siete medianas se sitúan dentro de un intervalo de ±0,6 EUR respecto a cero. El martes presenta la mediana más baja (−0,61) y la mayor dispersión (σ = 11,04), debido a un único residuo cercano a −65 EUR. La prueba de Kruskal-Wallis sobre los siete grupos arroja H = 3,41, p = 0,755, muy por encima de cualquier umbral convencional: no se puede rechazar la hipótesis nula de distribuciones iguales. No hay ningún efecto explotable del día de la semana en los datos, lo que concuerda con el predominio del paseo aleatorio documentado en los *apartados 6.2* y *6.4*.

6.6 MOMENTO ÓPTIMO DE COMPRA

La cuestión es si los precios son sistemáticamente más baratos cuando se compra con cierta antelación respecto a la fecha de salida, independientemente de cualquier previsión. Las 1371 observaciones reales de las 28 series del conjunto de datos B, normalizadas según la mediana de cada serie para que todas las rutas contribuyan por igual, se analizan mediante tres enfoques independientes.

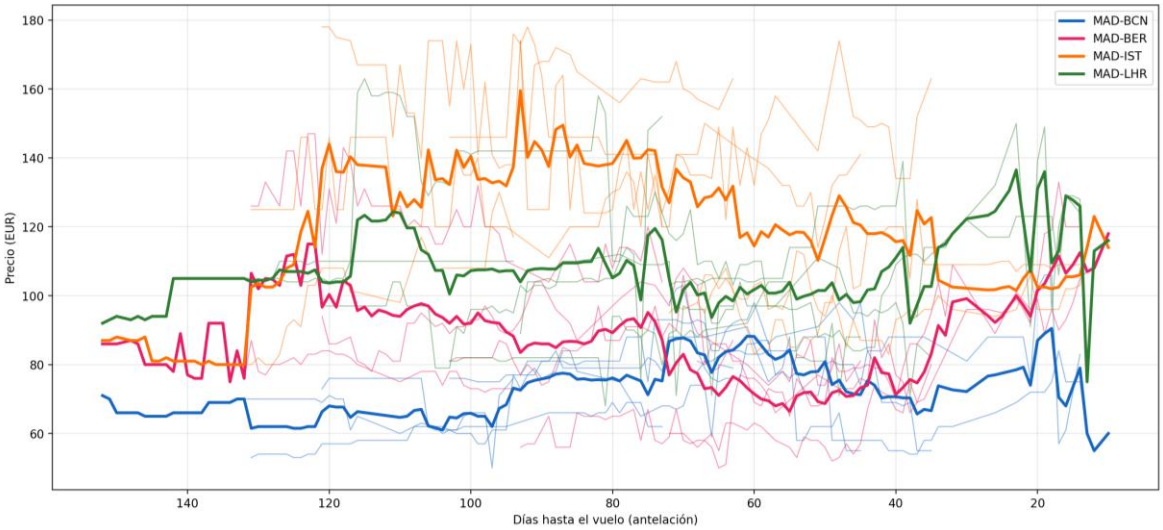

Ilustración 46. Evolución del precio en función de los días hasta el vuelo para las 28 series individuales del conjunto de datos B. Líneas finas: series individuales. Líneas gruesas: promedio por ruta.

6.6.1 ENFOQUE 1: CURVA NORMALIZADA AGREGADA

Precio normalizado mediano por intervalo de 10 días (*Figura 47*). El mínimo se sitúa en el intervalo de 46 a 60 días (mediana = 0,952, ≈5 % por debajo del precio típico de la serie). El intervalo de 0 a 10 días alcanza 1,103. Ahorro estimado al comprar con unos 50 días de antelación frente a los últimos diez días: 13,6 %.

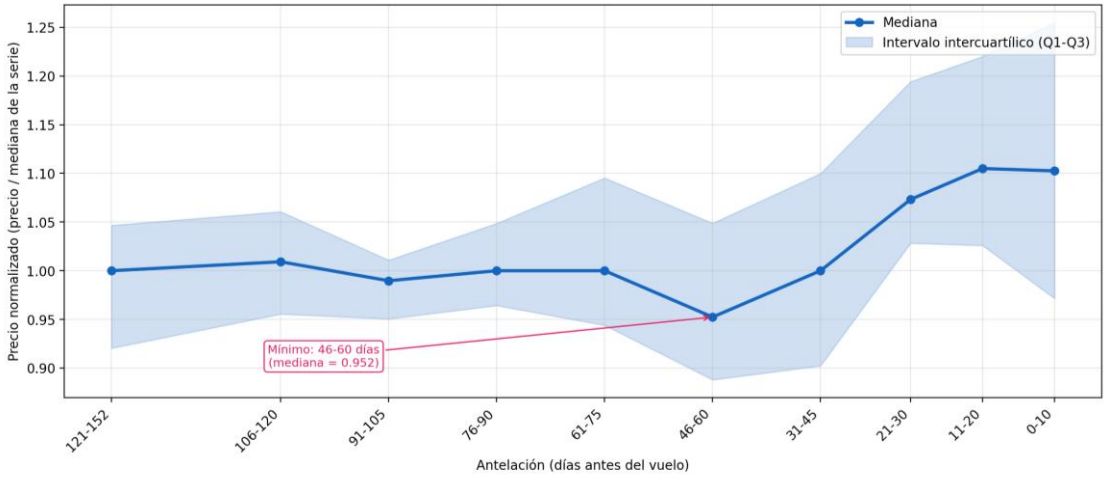

Ilustración 47. Curva agregada del precio normalizado en función de la antelación al vuelo, sobre N = 1371 observaciones. La banda sombreada representa el intervalo intercuartílico.

6.6.2 ENFOQUE 2: MÍNIMO INDIVIDUAL POR SERIE

Para cada una de las 28 series, se registra el intervalo que contiene su mínimo observado (*Figura 48*). El intervalo 46–60 es el modal, con 6 series (21 %), seguido del 76–90, con 5, y de los intervalos 31–45 y 91–105, con 4 cada uno. La distribución es dispersa: incluso el intervalo modal solo recoge una quinta parte de las series.

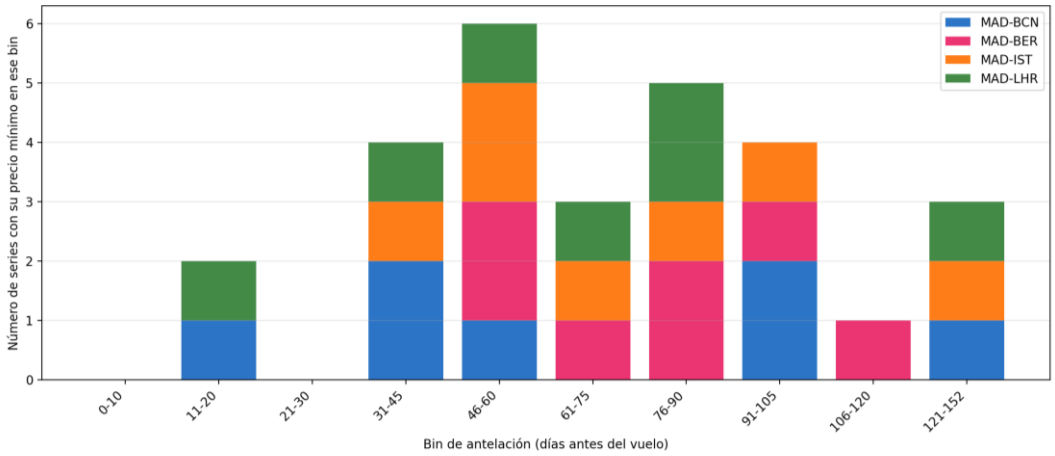

Ilustración 48. Distribución del intervalo de antelación en el que cada una de las 28 series alcanzó su precio mínimo, agrupada por ruta.

6.6.3 ENFOQUE 3: REGRESIÓN CUADRÁTICA

Ajuste cuadrático sobre las 1371 observaciones:

$$\text{precionorm} = 1.515 \cdot 10 - 5\text{d}2 - 3.025 \cdot 10 - 3\text{d} + 1.135$$

La curva es convexa con un mínimo analítico en d = 99,8 días, con un IC bootstrap del 95 % de [93,1, 112,4] días (1000 remuestreos). El R^2 es 0,044, lo que significa que la regresión explica menos del 5 % de la varianza en los precios normalizados, por lo que la ubicación precisa del mínimo no está fuertemente anclada a los datos.

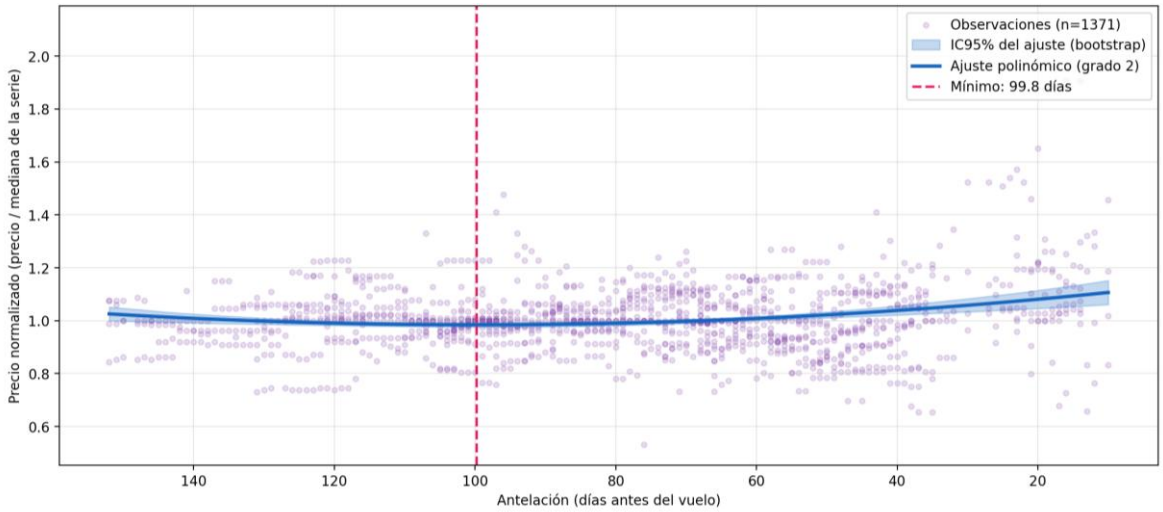

Ilustración 49. Ajuste polinómico de grado 2 sobre el precio normalizado frente a la antelación (n = 1371). La línea discontinua marca el mínimo analítico en 99,8 días.

6.6.4 SÍNTESIS

Enfoque	Método	Mínimo identificado	Indicador de confianza
1	Curva agregada normalizada	46–60 días	mediana 0.952 (−5 % vs. mediana)
2	Mínimo individual por serie	46–60 días (bin modal)	6 de 28 series (21 %)
3	Regresión polinómica grado 2	99.8 días [93.1, 112.4]	$R^2 = 0.044$

Tabla 21. Resultados de los tres enfoques sobre el efecto de la antelación.

Los enfoques 1 y 2 coinciden en 46–60 días; el enfoque 3 sitúa el mínimo en 100 días, pero su bajo R^2 debilita esa ubicación. La recomendación más sólida es la más conservadora: evitar comprar en las dos últimas semanas, donde el precio típico se sitúa en torno a un 10 % por encima de la mediana de la serie.

7 CONCLUSIONES Y TRABAJOS FUTUROS

7.1 RESUMEN

El proyecto generó dos conjuntos de datos sobre los precios de los billetes de avión con origen en Madrid mediante la interceptación de la API GraphQL de Kiwi.com, aplicó cuatro métodos de predicción (Naive, ARIMA, LSTM, XGBoost) a las series resultantes y comprobó dos posibles regularidades: el día de la semana y la antelación a la salida. El conjunto de datos A procede de una única sesión de scraping en seis rutas; el conjunto de datos B, de una campaña longitudinal de unos cincuenta días en cuatro rutas con siete fechas de vuelo cada una. El *capítulo 6* presenta los resultados numéricos; este capítulo los relaciona con los objetivos establecidos en el *capítulo 4*.

7.2 OBJETIVOS CUMPLIDOS

- El primer objetivo, la creación de un sistema de scraping web y los dos conjuntos de datos resultantes, se documenta en la *sección 5.2* y en el *Anexo II, Fase 1*. El conjunto de datos A contiene 1183 observaciones en seis rutas, y el conjunto de datos B contiene 1371 observaciones reales en 28 series de fechas de vuelo, agregadas en cuatro series a nivel de ruta para los modelos clásicos.
- El segundo objetivo, la aplicación de ARIMA a ambos conjuntos de datos bajo una metodología rigurosa, se describe en las *secciones 6.1 y 6.2*. El proceso incluye pruebas de estacionariedad duales ADF + KPSS, selección automatizada del orden mediante `auto_arima` con AIC, diagnóstico de residuos con Ljung-Box y Shapiro-Wilk, y comparación con el punto de referencia ingenuo mediante cuatro métricas de error. El análisis de sensibilidad del *apartado 6.3* añade una segunda capa de validación específica para el conjunto de datos B.
- El tercer objetivo, implementar un LSTM en el conjunto de datos B con características diseñadas y compararlo con ARIMA y la previsión ingenua, se

describe en el *apartado 6.4*. La arquitectura es un LSTM de 32 unidades con una ventana de entrada de 7 días sobre cuatro características continuas (precio, días hasta la salida, seno y coseno del día de la semana) más una codificación de ruta one-hot. Se añadió XGBoost como modelo tabular complementario. Aunque no formaba parte del objetivo original, triangula el resultado del LSTM y refuerza el marco de comparación.

- El cuarto objetivo, el análisis práctico del momento de la compra, se desarrolla en el *apartado 6.6*. Tres enfoques independientes convergen parcialmente: los métodos descriptivos señalan que el intervalo más barato es de 46 a 60 días, mientras que la regresión polinómica sitúa el mínimo más lejos, en torno a los 100 días, con R2 = 0,044. La recomendación sólida en los tres enfoques es evitar las dos últimas semanas antes de la salida, en las que los precios se sitúan aproximadamente un 10 % por encima del precio típico de la serie.
- El quinto objetivo, documentar las consecuencias metodológicas de las dos estrategias de scraping, se aborda en los *apartados 6.1 y 6.3*. El primero cuantifica el artefacto de cardinalidad del scraping de una sola sesión (la denominada «escalera»), y el segundo aísla el papel de la interpolación lineal en el conjunto de datos B comparando el modelo ARIMA en la serie interpolada con el modelo SARIMAX con NaNs filtrados por Kalman. Ambos análisis se integran en el capítulo de resultados y reaparecen como limitaciones en el *apartado 7.4*.

7.3 PRINCIPALES CONTRIBUCIONES

- Tres contribuciones de este trabajo se sitúan, por encima del alcance habitual de los trabajos de grado en este ámbito. La primera es la confirmación empírica de la hipótesis del paseo aleatorio en los precios de los vuelos a corto plazo en tres de las cuatro rutas del conjunto de datos B, respaldada por cuatro ángulos independientes: el bajo rendimiento de ARIMA, la convergencia de LSTM y XGBoost hacia la previsión ingenua en horizontes cortos, y la incapacidad de detectar un efecto del

día de la semana. El patrón concuerda con Fama (1965) y explica el modesto rendimiento de ARIMA descrito en la bibliografía revisada en el *capítulo 2*.

- La segunda es la documentación y cuantificación del artefacto de cardinalidad que produce el scraping de API en una sola sesión: aproximadamente veinte precios únicos en unas doscientas observaciones, con mesetas medias cercanas a los diez días, atribuibles a la caché de sesión de *Kiwi.com*. La bibliografía revisada no documentó este patrón con una precisión comparable.
- El tercero es la identificación de MAD-BCN como una excepción consistente en tres métodos de predicción independientes (ARIMA, LSTM, XGBoost), todos los cuales superan la predicción ingenua en esta ruta en algún horizonte. La triple convergencia no extiende el resultado más allá de la ventana de prueba, pero descarta el efecto de un único modelo como explicación. Una contribución práctica menor proviene del *apartado 6.6*, donde tres enfoques estadísticos coinciden en que la variable de antelación tiene un efecto pequeño pero persistente sobre el precio, con la regla operativa de evitar las dos últimas semanas antes de la salida.

7.4 LIMITACIONES

Dado que el conjunto de datos es breve y cada serie del Conjunto de datos B contiene 59 días, de los cuales solo 18 pertenecen a la ventana de prueba, lo que limita tanto la estadística de las pruebas de diagnóstico como el régimen en el que cabría esperar que la arquitectura LSTM dominara las líneas de base clásicas. El scraping abarca una única plataforma (Kiwi.com) y un único intervalo de tiempo (primavera de 2026), por lo que no se abordan los efectos estacionales ni las diferencias de precios entre plataformas. El ARIMA univariante fue la única especificación clásica probada, dejando el ARIMAX con regresores exógenos como una cuestión abierta. La escalera del Conjunto de datos A, una vez comprendida, limita la comparabilidad entre los dos conjuntos de datos a un papel ilustrativo en lugar de una comparación totalmente equivalente.

7.5 TRABAJOS FUTUROS

- Una campaña de recopilación de datos más prolongada, de seis a doce meses, ampliaría el horizonte de la prueba y permitiría una lectura estacional de la misma serie, algo que la ventana actual de la primavera de 2026 no puede proporcionar.
- Repetir la campaña en plataformas adicionales como *Skyscanner* o *Google Flights* permitiría comprobar si el hallazgo del paseo aleatorio se extiende más allá de *Kiwi.com* o es específico de su estructura de precios.
- En cuanto al modelado, una especificación ARIMAX con antelación y día de la semana como regresores exógenos es el siguiente paso natural, ya que los pequeños efectos identificados en los *apartados 6.6 y 6.5* no son aprovechados actualmente por los modelos univariantes de este trabajo. También merece la pena explorar las arquitecturas basadas en Transformer una vez que una campaña más amplia proporcione los datos que necesitan, en línea con la bibliografía más reciente revisada en el *apartado 2.7*.
- La excepción MAD-BCN merece ser replicada en un conjunto de datos independiente para comprobar si el resultado se generaliza más allá de la ventana de prueba de este trabajo.
- Por último, la extensión más útil desde un punto de vista práctico sería una aplicación orientada al cliente que exponga las previsiones y el análisis del momento de compra como una herramienta visual: una interfaz fácil de usar en la que un viajero pudiera introducir la ruta y la fecha y ver, junto al precio actual, la recomendación del modelo y la curva de antelación para ese destino. Los componentes técnicos ya están desarrollados; su integración en un producto utilizable convertiría un ejercicio analítico en algo que un pasajero pudiera realmente utilizar para tomar una decisión de compra.

8 BIBLIOGRAFÍA

(Chen & Guestrin, 2016)
Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*(6), 716-723. https://doi.org/10.1109/TAC.1974.1100705

Antonio, N., Almeida, A. de, & Nunes, L. (2017). Predicting hotel booking cancellations to decrease uncertainty and increase revenue. *Tourism & Management Studies*, *13*(2), 25-39. https://doi.org/10.18089/tms.2017.13203

Belobaba, P., Odoni, A., & Barnhart, C. (2015). *The global airline industry*. John Wiley & Sons. https://books.google.es/books?hl=es&lr=&id=HwBhBgAAQBAJ&oi=fnd&pg=PA99&dq=Belobaba,+P.,+Odoni,+A.,+%26+Barnhart,+C.+(2015).+The+Global+Airline+Industry+(2nd+ed.).+Wiley.&ots=c3jEtrHqhc&sig=bRbHH3o8WuiqYkWB4JA6gIIp3M4

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *26*(2), 211-243.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.

Box, G., & Jenkins, G. M. (1976). Analysis: Forecasting and control. *San francisco*, *10*. https://books.google.es/books?hl=es&lr=&id=dFKUIS_puRcC&oi=fnd&pg=PA161&dq=Box,+G.+E.+P.,+%26+Jenkins,+G.+M.+(1970).+Time+Series+Analysis:+Forecasting+and+Control.+Holden-Day.&ots=4v4Ubuz3ja&sig=nHSQtaO_irR5wKVe8loWU0--RCk

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794. https://doi.org/10.1145/2939672.2939785

CLEVELAND, R. (1990). STL: A seasonal-trend decomposition procedure based on loess. *J Off Stat*, *6*, 3-73.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, *74*(366a), 427-431. https://doi.org/10.1080/01621459.1979.10482531

Etzioni, O., Tuchinda, R., Knoblock, C. A., & Yates, A. (2003). To buy or not to buy: Mining airfare data to minimize ticket purchase price. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 119-128. https://doi.org/10.1145/956750.956767

Fama, E. F. (1965). The behavior of stock-market prices. *The journal of Business*, *38*(1), 34-105.

Findley, D. F., Monsell, B. C., Bell, W. R., Otto, M. C., & Chen, B.-C. (1998). New Capabilities and Methods of the X-12-ARIMA Seasonal-Adjustment Program. *Journal of Business & Economic Statistics*, *16*(2), 127-152. https://doi.org/10.1080/07350015.1998.10524743

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189-1232.

GraphQL Foundation. (2021). *GraphQL Specification*. https://spec.graphql.org/October2021/?utm_source=chatgpt.com

Hartig, O., & Pérez, J. (2018). Semantics and Complexity of GraphQL. *Proceedings of the 2018 World Wide Web Conference, WWW '18*, 1155-1164. https://doi.org/10.1145/3178876.3186014

hiQ Labs, Inc. v. LinkedIn Corp. (9th Cir. 2019).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software*, *27*, 1-22. https://doi.org/10.18637/jss.v027.i03

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International journal of forecasting*, *22*(4), 679-688.

International Journal of Forecasting. (1995). North-Holland.

Janssen, T., Dijkstra, T., Abbas, S., & van Riel, A. C. (2014). A linear quantile mixed regression model for prediction of airline ticket prices. *Radboud University*, *3*. http://www.cs.ru.nl/bachelorscripties/2014/Tim_Janssen___4150880___A_Linear_Quantile_Mixed_Regression_Model_for_Prediction_of_Airline_Ticket_Prices.pdf

Jarque, C. M., & Bera, A. K. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, *6*(3), 255-259. https://doi.org/10.1016/0165-1765(80)90024-5

Khder, M. A. (2021). Web scraping or web crawling: State of art, techniques, approaches and application. *International Journal of Advances in Soft Computing & Its Applications*, *13*(3). http://www.i-csrs.org/Volumes/ijasca/2021.3.11.pdf

Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization* (arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980

Lantseva, A., Mukhina, K., Nikishova, A., Ivanov, S., & Knyazkov, K. (2015). Data-driven modeling of airlines pricing. *Procedia Computer Science*, *66*, 267-276.

LJUNG, G. M., & BOX, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, *65*(2), 297-303. https://doi.org/10.1093/biomet/65.2.297

Mitchell, R. (2018). *Web Scraping with Python, 2nd Edition*. https://www.oreilly.com/library/view/web-scraping-with/9781491985564/

Mumbower, S., Garrow, L. A., & Newman, J. P. (2015). Investigating airline customers' premium coach seat purchases and implications for optimal pricing strategies. *Transportation Research Part A: Policy and Practice*, *73*, 53-69.

Newman, J. P., Ferguson, M. E., Garrow, L. A., & Jacobs, T. L. (2014). Estimation of Choice-Based Models Using Sales Data from a Single Firm. *Manufacturing & Service Operations Management*, *16*(2), 184-197. https://doi.org/10.1287/msom.2014.0475

Páez, A., & Boisjoly, G. (2022). Exploratory Data Analysis. En A. Páez & G. Boisjoly (Eds.), *Discrete Choice Analysis with R* (pp. 25-64). Springer International Publishing. https://doi.org/10.1007/978-3-031-20719-8_2

Ren, R., Yang, Y., & Yuan, S. (2014). Prediction of airline ticket price. *University of Stanford*, 1-5.

Santana, E., & Mastelini, S. (2017). Deep regressor stacking for air ticket prices prediction. *Simpósio Brasileiro de Sistemas de Informação (SBSI)*, 25-31. https://sol.sbc.org.br/index.php/sbsi/article/view/6022

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 461-464.

SHAPIRO, S. S., & WILK, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, *52*(3-4), 591-611. https://doi.org/10.1093/biomet/52.3-4.591

Shevelenkov, A. (2024). *Expanding end-to-end test coverage using Playwright framework for Sievo Oy's Materials Forecasting product*. http://www.theseus.fi/handle/10024/872865

Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*. https://www.academia.edu/download/46980886/KPSS.pdf

Shumway, R. H., & Stoffer, D. S. (Eds.). (2006). State-Space Models. En *Time Series Analysis and Its Applications: With R Examples* (pp. 324-411). Springer. https://doi.org/10.1007/0-387-36276-2_6

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

Talluri, K. T., & Van Ryzin, G. J. (2006a). *The theory and practice of revenue management* (Vol. 68). Springer Science & Business Media. https://books.google.es/books?hl=es&lr=&id=GhdDbEM-_5oC&oi=fnd&pg=PR5&dq=Talluri,+K.+T.,+%26+van+Ryzin,+G.+J.+(2004).+The+Theory+and+Practice+of+Revenue+Management.+Springer.&ots=wK7KXEw-HZ&sig=wKAsTPK9A78iSw5Hm1xDqnjAhvA

Talluri, K. T., & Van Ryzin, G. J. (2006b). *The theory and practice of revenue management* (Vol. 68). Springer Science & Business Media. https://books.google.es/books?hl=es&lr=&id=GhdDbEM-_5oC&oi=fnd&pg=PR5&dq=Talluri,+K.+T.,+and+van+Ryzin,+G.+J.+(2004).+The+Theory+and+Practice+of+Revenue+Management.+Springer.&ots=wK7LTFy-MW&sig=bpPDANmvRDPO8s29h-07CIT3_YU

Tziridis, K., Kalampokas, T., Papakostas, G. A., & Diamantaras, K. I. (2017). Airfare prices prediction using machine learning techniques. *2017 25th European Signal Processing Conference (EUSIPCO)*, 1036-1039. https://ieeexplore.ieee.org/abstract/document/8081365/?casa_token=IrYn2LVkIwEAAAAA:zOYGx9UlvGNotY1gH6jOcsrgK_imQFE0i_ZyzMmaACnxQKx1Pvsm6tv7aDZJMO4e6LkdcIa8VQ

Wang, T., Pouyanfar, S., Tian, H., Tao, Y., Alonso, M., Luis, S., & Chen, S.-C. (2019). *A Framework for Airfare Price Prediction: A Machine Learning Approach*. 200-207. https://doi.org/10.1109/IRI.2019.00041

Wohlfarth, T., Clémençon, S., Roueff, F., & Casellato, X. (2011). A data-mining approach to travel price forecasting. *2011 10th International Conference on Machine Learning and*

Applications and Workshops, *1*, 84-89. https://ieeexplore.ieee.org/abstract/document/6146948/

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

El presente trabajo de fin de grado analiza la previsión a corto plazo de los precios de los billetes de avión desde Madrid y la relación entre el momento de la compra y el precio pagado. Su alineación con los Objetivos de Desarrollo Sostenible se articula principalmente a través de la dimensión de la información al consumidor, con conexiones secundarias con la asequibilidad de la movilidad y con el impacto climático de la aviación.

Dimensión	ODS	Función	Meta específica a la que se dirige
Economía	ODS 12 — Consumo y producción responsables	Primaria	12.8 (información para estilos de vida sostenibles)
Sociedad	ODS 11 — Ciudades y comunidades sostenibles	Secundaria	11.2 (acceso a un transporte seguro y asequible)
Biosfera	ODS 13 — Acción por el clima	Secundaria	13.3 (educación y sensibilización sobre el cambio climático)

Tabla 22. ODS identificados para este trabajo, por dimensión de sostenibilidad.

La meta 12.8 insta a garantizar que las personas dispongan de la información necesaria para llevar estilos de vida sostenibles. La fijación de precios de las aerolíneas se rige por sistemas de gestión de ingresos cuya lógica resulta opaca para el comprador, y las herramientas comerciales que intentan salvar esa brecha se basan en algoritmos propios cuya precisión no puede verificarse externamente. Este TFG contribuye a la meta 12.8 al proporcionar un análisis reproducible del comportamiento de los precios de los billetes de avión antes de la salida y una recomendación para los compradores derivada del *apartado 6.6*: evitar los catorce días previos a la salida supone un ahorro medio de alrededor del 10 % del precio del

billete (entre 7 y 11 euros por billete en las rutas europeas estudiadas), ya que ese periodo conlleva un recargo sobre el precio habitual de la serie.

La contribución a la meta 11.2 funciona a través del mismo mecanismo de asimetría de la información: reducir la brecha entre el viajero y la compañía aérea hace que la infraestructura de transporte existente sea más accesible en términos económicos, especialmente para los viajeros con un presupuesto limitado que pagan el coste relativo más alto cuando se ven obligados a comprar a última hora. La conexión con la meta 13.3 requiere un reconocimiento explícito de sus límites. La aviación es uno de los sectores de transporte con mayor intensidad de carbono por pasajero-kilómetro, y este TFG no aborda el coste climático de los vuelos, no estima la huella de carbono de las rutas y no propone un mecanismo para reducir la demanda. Su relación con el ODS 13 se limita a la dimensión educativa de la meta 13.3: al documentar abiertamente el comportamiento de los precios de los billetes de avión y garantizar que la metodología y los datos sean reproducibles, este trabajo contribuye a que el público comprenda la estructura económica del transporte aéreo, un requisito previo para cualquier debate informado sobre su coste medioambiental. Las futuras ampliaciones esbozadas en el *capítulo 7* se beneficiarían de la incorporación de una dimensión explícita de equivalentes de carbono, además de la basada en los precios.

ANEXO II

Este anexo presenta los fragmentos más representativos del Código Python desarrollado para el proyecto. Se incluyen las funciones nucleares de cada fase del pipeline, seleccionadas por su relevancia técnica.

El proyecto consta de 8 módulos Python y un archivo HTML interactivo. La tabla siguiente resume la función de cada script y la figura del capítulo 5 muestra las dependencias entre ellos.

Script	**Funcion**	**Entrada / Salida**
scraping_diario_vuelos_fijos.py	Scraping diario del Dataset B: intercepta la API GraphQL de Kiwi.com con Playwright y registra el precio mínimo de cada vuelo.	Kiwi.com (web) → SCRAPING_DIARIO/ (.txt por ruta y fecha)
consolidar_dataset_B.py	Consolida los .txt del scraping en CSVs limpios: reindexado diario, interpolación lineal, detección de outliers IQR 3.0.	SCRAPING_DIARIO/ → dataset_b_clean/ (consolidado + series agregadas + individuales)
arima_dataset_A.py	Pipeline ARIMA completo para las 6 rutas del Dataset A: estacionariedad (ADF/KPSS), auto_arima, residuos, predicción con IC y benchmark naive.	data_clean/ → ARIMA_dataset_A/ (figuras + tablas)
arima_dataset_B.py	Pipeline ARIMA análogo para las 4 series agregadas del Dataset B (split 70/30).	series_agregadas/ → ARIMA_dataset_B/
sensibilidad_dataset_B.py	Análisis de sensibilidad: ARIMA sobre serie interpolada vs.	dataset_b_clean/ → SENSIBILIDAD_dataset_B/

	SARIMAX sobre serie con huecos, para validar que la interpolación no distorsiona.	
pipeline_LSTM_XGBoost.py	Entrena y evalua Naive, LSTM y XGBoost a horizontes de 1, 3, 7 y 14 días sobre las 28 series individuales del Dataset B.	SCRAPING_DIARIO/ → LSTM_XGBoost/ (modelos .keras + métricas)
cap6_complementos.py	Genera los complementos del capitulo 6: día optimo de compra, 28 series superpuestas, tabla maestra y análisis de antelación.	Resultados previos → CAP6_complementos/
mapa_interactivo.py	Genera un mapa interactivo HTML con las 6 rutas sobre arcos geodésicos, usando Folium (Leaflet).	(datos hardcoded) → mapa_rutas_interactivo.html

A continuación, se presentan los fragmentos de código seleccionados, organizados por fase del pipeline.

8.1 FASE 1: EXTRACCIÓN DE DATOS (SCRAPING)

El scraper utiliza Playwright para lanzar un navegador Chromium en modo headless e interceptar las respuestas de la API GraphQL interna de Kiwi.com (SearchOneWayItinerariesQuery). Esta técnica captura los datos JSON antes de su renderizado en la interfaz web, evitando el parsing de HTML y obteniendo campos estructurados. Las dos funciones siguientes constituyen el núcleo del proceso de extracción.

```
def scrape_route_date(page, origin, dest, flight_date, cookies_accepted):
    """Navega a Kiwi.com, intercepta SearchOneWayItinerariesQuery
    y extrae el precio minimo del vuelo."""
    url = (f"https://www.kiwi.com/es/search/results/"
           f"{KIWI_SLUGS[origin]}/{KIWI_SLUGS[dest]}/{flight_date}/no-return/")
```

```
    captured = {}

    def on_response(resp):
        if (resp.status == 200
            and "SearchOneWayItinerariesQuery" in resp.url
            and "json" in resp.headers.get("content-type", "")
            and "data" not in captured):
            captured["data"] = resp.json()

    page.on("response", on_response)
    page.goto(url, wait_until="domcontentloaded", timeout=60_000)
    page.wait_for_timeout(WAIT_AFTER_LOAD * 1_000)
    page.remove_listener("response", on_response)

    if "data" in captured:
        return _parse_search_response(captured["data"])
    return None


def _parse_search_response(data):
    """Extrae (precio_min, aerolinea, escalas) de la respuesta GraphQL."""
    itineraries = data["data"]["onewayItineraries"]["itineraries"]
    best_price, best_airline, best_stops = None, "NA", "NA"

    for it in itineraries:
        amount = float(it["price"]["amount"])
        if amount <= 0:
            continue
        segs = (it.get("sector") or {}).get("sectorSegments") or []
        carriers = {s.get("segment",{}).get("carrier",{}).get("code","")
                    for s in segs} - {""}
        stops = max(len(segs) - 1, 0)
        if best_price is None or amount < best_price:
            best_price   = amount
            best_airline = " / ".join(sorted(carriers)) or "NA"
            best_stops   = "directo" if stops == 0 else f"{stops} escala(s)"

    return (best_price, best_airline, best_stops) if best_price else None
```

8.2 FASE 2: CONSOLIDACIÓN Y LIMPIEZA

La función siguiente construye las 4 series temporales agregadas del Dataset B (una por ruta). Para cada ruta, calcula la media diaria del precio sobre las 7 fechas de vuelo monitorizadas, reindexa la serie a frecuencia diaria continua con interpolación lineal para los días sin observación, y aplica un filtro IQR con factor 3.0 para detectar outliers extremos sin eliminar variabilidad legitima.

```
def construir_series_agregadas(df):
    """Para cada ruta, calcula la media diaria del precio sobre las
    7 fechas de vuelo. Resultado: 4 series temporales."""
    for ruta in ROUTES:
        sub = df[df["ruta"] == ruta]
        agg = (sub.groupby("dia_scraping")["precio"].mean()
               .reset_index()
               .rename(columns={"precio": "precio_medio"})
               .sort_values("dia_scraping"))

        # Reindex diario + interpolacion lineal
        agg = agg.set_index("dia_scraping")
        rango = pd.date_range(agg.index.min(), agg.index.max(), freq="D")
        agg = agg.reindex(rango)
        agg["precio_medio"] = agg["precio_medio"].interpolate(method="linear")

        # Outliers IQR factor 3.0
        Q1 = agg["precio_medio"].quantile(0.25)
        Q3 = agg["precio_medio"].quantile(0.75)
        IQR = Q3 - Q1
        mask_out = ((agg["precio_medio"] < Q1 - 3.0 * IQR) |
                    (agg["precio_medio"] > Q3 + 3.0 * IQR))
        agg.loc[mask_out, "precio_medio"] = np.nan
        agg["precio_medio"] = agg["precio_medio"].interpolate(method="linear")

        agg.reset_index().to_csv(AGG_DIR / f"{ruta}_ts.csv", index=False)
```

8.3 FASE 3A: MODELADO ARIMA

El pipeline ARIMA se aplica de forma idéntica a las 6 rutas del Dataset A (split 80/20) y a las 4 series agregadas del Dataset B (split 70/30). Se muestra a continuación el núcleo común: la selección automática de orden con `auto_arima`, la predicción out-of-sample con intervalos de confianza, y el benchmark naive.

```
def paso3_auto_arima(train, route, out_dir):
    """auto_arima con los parametros del TFG."""
    model = auto_arima(
        train["precio_medio"].values,
        start_p=0, max_p=5, start_q=0, max_q=2,
        d=None, max_d=2,
        seasonal=False, stepwise=True,
        information_criterion="aic",
    )
```

```
    return model

def paso5_prediccion(model, train, test):
    """Prediccion out-of-sample y comparacion con naive."""
    real  = test["precio_medio"].values
    n     = len(test)
    forecast, conf_80 = model.predict(n_periods=n, return_conf_int=True,
alpha=0.20)
    _, conf_95        = model.predict(n_periods=n, return_conf_int=True,
alpha=0.05)

    rmse = np.sqrt(mean_squared_error(real, forecast))
    mae  = mean_absolute_error(real, forecast)
    mase = mae / mean_absolute_error(real[1:], real[:-1])
    return {"RMSE": rmse, "MAE": mae, "MASE": mase, "forecast": forecast}

def paso6_naive(train, test):
    """Benchmark naive: y_hat(t+1) = y(t)."""
    naive = np.full(len(test), train["precio_medio"].iloc[-1])
    rmse  = np.sqrt(mean_squared_error(test["precio_medio"].values, naive))
    return {"RMSE_naive": rmse, "forecast_naive": naive}
```

8.4 FASE 3B: LSTM Y XGBOOST

El pipeline de machine learning entrena ambos modelos sobre las 28 series individuales del Dataset B con un split temporal 60/10/30 (train/validation/test). El LSTM utiliza una arquitectura compacta de ~5 500 parámetros, adecuada para series cortas. Los datos de entrada se organizan en ventanas deslizantes de 7 días, y el target es el incremento de precio (no el nivel absoluto), lo que facilita la comparación con la predicción naive. El ensamble de 5 semillas mitiga la varianza de inicialización. El XGBoost opera sobre la misma ventana aplanada a vector tabular.

```
# ── Arquitectura LSTM ──────────────────────────────────────
def construir_lstm(ventana, n_features):
    """LSTM compacto (~5 500 parametros), apropiado para series cortas."""
    modelo = Sequential([
        LSTM(32, input_shape=(ventana, n_features)),
        Dropout(0.2),
        Dense(8, activation='relu'),
        Dense(1)
    ])
    modelo.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return modelo

# ── Construccion de ventanas deslizantes ──────────────────
```

```
def construir_datos(df_full, features_cont, features_cat, ventana, horizonte,
                    prop_test, prop_val):
    """Para cada serie: escala con MinMaxScaler (fit solo en train),
    construye ventanas de 7 dias, target = incremento de precio."""
    for sid, g in df_full.groupby('series_id'):
        n = len(g)
        fin_train = int(n * (1 - prop_test) * (1 - prop_val))
        scaler = MinMaxScaler()
        scaler.fit(g.iloc[:fin_train][features_cont].values)

        escalado_cont = scaler.transform(g[features_cont].values)
        escalado_cat  = g[features_cat].values.astype(float)
        escalado = np.concatenate([escalado_cont, escalado_cat], axis=1)

        for i in range(ventana, n - horizonte + 1):
            target_idx = i + horizonte - 1
            X = escalado[i - ventana:i]
            y = escalado[target_idx, 0] - escalado[i - 1, 0]
            # Asignar a train/val/test segun posicion del target
            ...
    return tensores_lstm, matrices_xgboost, metadatos

# ── XGBoost ──────────────────────────────────────────────────
def construir_xgboost():
    return xgb.XGBRegressor(
        n_estimators=500, max_depth=4, learning_rate=0.05,
        subsample=0.8, colsample_bytree=0.8,
        early_stopping_rounds=30, random_state=42
    )
```

8.5 FASE 3C: DIAGNOSTICO DE RESIDUOS ARIMA

Tras ajustar cada modelo ARIMA, se valida la hipótesis de ruido blanco sobre los residuos mediante el test de Ljung-Box (autocorrelación) y el test de Shapiro-Wilk (normalidad). La función genera además la figura de 6 paneles (serie de residuos, histograma, ACF, PACF, Q-Q plot y resumen numérico) que se incluye en el *capítulo 6* para cada ruta.

```
def paso4_residuos(model, train, route, out_dir):
    """Diagnostico completo de residuos: Ljung-Box, Shapiro-Wilk y 6 paneles."""
    resid = model.resid()
    n_lags_acf = min(30, len(resid) // 2 - 1)

    # Test de Ljung-Box (autocorrelacion residual)
    lb = acorr_ljungbox(resid, lags=[10, 20], return_df=True)
    lb10_pval = lb.iloc[0]["lb_pvalue"]
    lb20_pval = lb.iloc[1]["lb_pvalue"]
```

```
    lb_ok = lb10_pval > 0.05 and lb20_pval > 0.05

    # Test de Shapiro-Wilk (normalidad)
    sw_stat, sw_pval = st.shapiro(resid)
    sw_normal = sw_pval > 0.05

    # Figura 3x2: serie, histograma, ACF, PACF, Q-Q, resumen
    fig, axes = plt.subplots(3, 2, figsize=(15, 13))
    axes[0, 0].plot(train.index[:len(resid)], resid, lw=0.8)
    axes[0, 0].axhline(0, color="red", ls="--")
    axes[0, 1].hist(resid, bins=35, density=True)
    plot_acf(resid, lags=n_lags_acf, ax=axes[1, 0])
    plot_pacf(resid, lags=n_lags_acf, ax=axes[1, 1], method="ywm")
    osm, osr = st.probplot(resid, dist="norm")[0]
    axes[2, 0].scatter(osm, osr, s=12, alpha=0.6)

    fig.savefig(out_dir / "fig_residuos.png", dpi=150)
    return {"lb_ok": lb_ok, "sw_normal": sw_normal}
```

8.6 FASE 3D: ANÁLISIS DE SENSIBILIDAD (INTERPOLACIÓN VS. HUECOS)

Para validar que la interpolación lineal del 18% de la serie no distorsiona los resultados del ARIMA, se comparan dos enfoques sobre los mismos datos: el Modelo A ajusta ARIMA sobre la serie ya interpolada, y el Modelo B ajusta SARIMAX con los mismos ordenes (p, d, q) sobre la serie con NaN, utilizando el filtro de Kalman integrado en statsmodels para tratar los valores faltantes. Si ambos producen métricas similares, la interpolación es benigna.

```
def ajustar_sarimax_con_huecos(train, order):
    """SARIMAX con filtro de Kalman para tratar NaN en la serie."""
    model = SARIMAX(
        train["precio_medio"],
        order=order,
        enforce_stationarity=False,
        enforce_invertibility=False,
    )
    return model.fit(disp=False, maxiter=200)

def procesar_ruta(route, out_dir):
    """Compara ARIMA interpolado (A) vs SARIMAX con huecos (B)."""
    ts_interp = cargar_serie_interpolada(route)   # 49 puntos, sin NaN
    ts_nan    = cargar_serie_sin_interpolar(route) # misma longitud, con NaN
```

```
    # Split temporal 70/30 (mismas fechas para ambos modelos)
    split_idx = int(len(ts_interp) * 0.70)
    train_interp = ts_interp.iloc[:split_idx]
    test_interp  = ts_interp.iloc[split_idx:]
    train_nan    = ts_nan.iloc[:split_idx]

    # Modelo A: auto_arima sobre serie interpolada
    model_a = auto_arima(train_interp["precio_medio"].values,
                         start_p=0, max_p=5, start_q=0, max_q=2,
                         d=None, seasonal=False, stepwise=True)
    forecast_a = model_a.predict(n_periods=len(test_interp))
    rmse_a = np.sqrt(mean_squared_error(test_interp["precio_medio"], forecast_a))

    # Modelo B: SARIMAX con mismos ordenes, sobre serie con NaN
    fit_b = ajustar_sarimax_con_huecos(train_nan, model_a.order)
    forecast_b = fit_b.forecast(steps=len(test_interp))
    rmse_b = np.sqrt(mean_squared_error(test_interp["precio_medio"], forecast_b))

    # Comparar: si |RMSE_A - RMSE_B| < umbral, interpolacion benigna
    return {"RMSE_A": rmse_a, "RMSE_B": rmse_b, "order": model_a.order}
```

8.7 FASE 4A: ANÁLISIS DEL DÍA DE LA SEMANA Y ANTELACIÓN DE COMPRA

El análisis del día optimo de compra elimina la tendencia lineal por días de antelación mediante regresión (linregress) y agrupa los residuos por día de la semana. El test de Kruskal-Wallis evalúa si existen diferencias significativas entre días. Para el análisis de antelación, las 1371 observaciones reales se normalizan por la mediana de su serie (para que todas las rutas contribuyan por igual) y se agrupan en bins de 10 o 15 días.

```
def analisis_dia_semana(series):
    """Elimina tendencia por antelacion y agrupa residuos por dia de la
semana."""
    filas = []
    for ruta, fecha_vuelo, df in series:
        reales = df[df["aerolinea"].notna()].copy()
        x = reales["dias_antelacion"].to_numpy(dtype=float)
        y = reales["precio"].to_numpy(dtype=float)
        slope, intercept, *_ = st.linregress(x, y)
        residuo = y - (intercept + slope * x)
        for i, (_, row) in enumerate(reales.iterrows()):
            dow = row["dia_scraping"].weekday()
            filas.append({
                "ruta": ruta, "dia_semana": DIAS_SEMANA_ES[dow],
                "residuo": float(residuo[i]),
```

```
            })
    df_res = pd.DataFrame(filas)

    # Test de Kruskal-Wallis
    grupos = [df_res.loc[df_res["dia_semana"] == d, "residuo"].values
              for d in DIAS_SEMANA_ES]
    h_stat, p_value = st.kruskal(*grupos)
    return h_stat, p_value  # p = 0.755 -> no hay efecto del dia


def analisis_antelacion(series):
    """Normaliza precios por mediana de cada serie y agrupa por bins
    de antelacion para detectar el momento optimo de compra."""
    filas = []
    for ruta, fecha_vuelo, df in series:
        reales = df[df["aerolinea"].notna()].copy()
        mediana_serie = reales["precio"].median()
        for _, row in reales.iterrows():
            filas.append({
                "ruta": ruta, "dias_antelacion": float(row["dias_antelacion"]),
                "precio_norm": float(row["precio"]) / mediana_serie,
            })
    df_obs = pd.DataFrame(filas)
    df_obs["bin"] = pd.cut(df_obs["dias_antelacion"],
                           bins=[0,10,15,20,25,30,45,60,75,90,105,120],
                           labels=["0-10","11-15","16-20","21-25","26-30",
                                   "31-45","46-60","61-75","76-90","91-105","106-
120"])
    resumen = df_obs.groupby("bin", observed=False)["precio_norm"].agg(
        mediana="median", Q1=lambda x: np.percentile(x, 25),
        Q3=lambda x: np.percentile(x, 75), n="count")
    return resumen  # minimo en bin 46-60 (mediana = 0.952)
```

8.8 FASE 4B: MAPA INTERACTIVO DE RUTAS

El siguiente script genera un mapa interactivo en formato HTML utilizando Folium (wrapper de Leaflet.js). Las rutas se dibujan como arcos de circulo máximo (geodésicas) calculados con trigonometría esférica. Las rutas del Dataset A exclusivamente (MAD-CDG, MAD-SIN) se representan con línea discontinua para distinguirlas de las 4 rutas presentes en ambos datasets. El archivo HTML resultante se puede abrir directamente en cualquier navegador.

```
"""Mapa interactivo con las 6 rutas del TFG - Folium (Leaflet)"""
import folium, numpy as np

airports = {
    'MAD': {'name':'Madrid-Barajas',    'lat':40.4722,'lon':-3.5611},
```

```
    'BCN': {'name':'Barcelona-El Prat',  'lat':41.2974,'lon':2.0833},
    'BER': {'name':'Berlin-Brandenburg', 'lat':52.3667,'lon':13.5033},
    'IST': {'name':'Istanbul',           'lat':41.2753,'lon':28.7519},
    'LHR': {'name':'London Heathrow',    'lat':51.4700,'lon':-0.4543},
    'CDG': {'name':'Paris-CDG',          'lat':49.0097,'lon':2.5479},
    'SIN': {'name':'Singapore Changi',   'lat':1.3502, 'lon':103.9940},
}

routes = [
    {'name':'MAD-BCN','color':'#1565C0','dist':483,  'ds':'A+B'},
    {'name':'MAD-LHR','color':'#2E7D32','dist':1261, 'ds':'A+B'},
    {'name':'MAD-BER','color':'#E91E63','dist':1869, 'ds':'A+B'},
    {'name':'MAD-IST','color':'#FF6F00','dist':3210, 'ds':'A+B'},
    {'name':'MAD-CDG','color':'#6A1B9A','dist':1065, 'ds':'solo A'},
    {'name':'MAD-SIN','color':'#78909C','dist':10564,'ds':'solo A'},
]

def gc_points(lat1, lon1, lat2, lon2, n=80):
    """Puntos sobre arco de circulo maximo (geodesica)."""
    lat1, lon1, lat2, lon2 = map(np.radians, [lat1, lon1, lat2, lon2])
    d = np.arccos(np.clip(
        np.sin(lat1)*np.sin(lat2)
        + np.cos(lat1)*np.cos(lat2)*np.cos(lon2-lon1), -1, 1))
    pts = []
    for f in np.linspace(0, 1, n):
        A = np.sin((1-f)*d)/np.sin(d)
        B = np.sin(f*d)/np.sin(d)
        x = A*np.cos(lat1)*np.cos(lon1) + B*np.cos(lat2)*np.cos(lon2)
        y = A*np.cos(lat1)*np.sin(lon1) + B*np.cos(lat2)*np.sin(lon2)
        z = A*np.sin(lat1) + B*np.sin(lat2)
        pts.append([np.degrees(np.arctan2(z, np.sqrt(x**2+y**2))),
                    np.degrees(np.arctan2(y, x))])
    return pts

m = folium.Map(location=[35, 45], zoom_start=3, tiles='CartoDB positron')

for r in routes:
    orig, dest = r['name'].split('-')
    a1, a2 = airports[orig], airports[dest]
    arc = gc_points(a1['lat'], a1['lon'], a2['lat'], a2['lon'])
    dash = '5' if r['ds'] == 'solo A' else None
    folium.PolyLine(arc, color=r['color'], weight=3.5,
                    dash_array=dash, popup=r['name']).add_to(m)

for code, info in airports.items():
    folium.CircleMarker(
        [info['lat'], info['lon']], radius=6,
        color='#333', fill=True, fill_color='white',
        popup=f"{code} - {info['name']}"
    ).add_to(m)

m.save("mapa_rutas_interactivo.html")
```