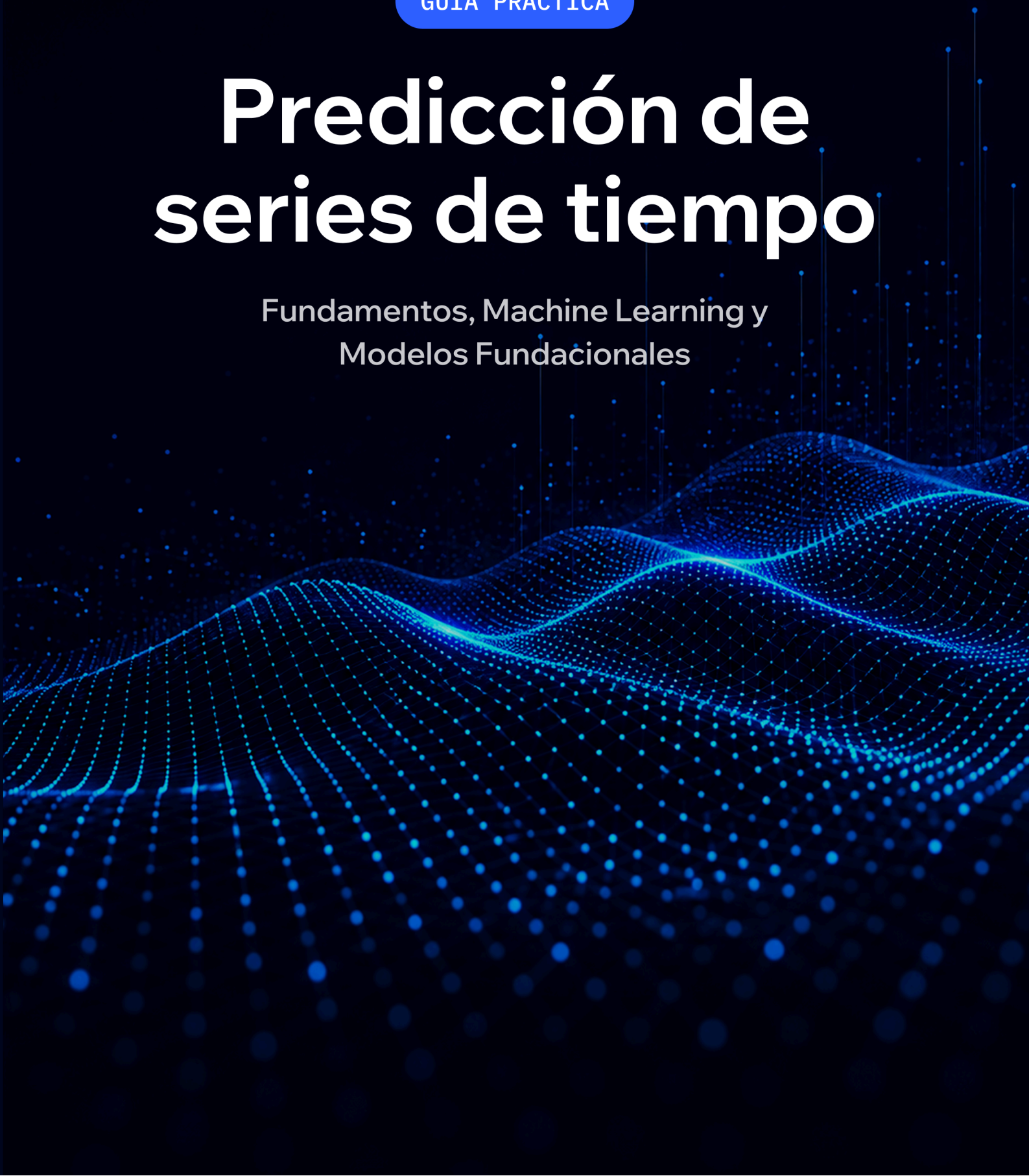
marvik.

GUÍA PRÁCTICA

Predicción de series de tiempo

Fundamentos, Machine Learning y Modelos Fundacionales



marvik.

Built for the Now

Marvik construye sistemas de IA en producción para entornos operativos complejos.

AI-native y forward-deployed desde 2018, Marvik ayuda a organizaciones líderes a ir más allá de los pilotos y las demos, convirtiendo oportunidades de IA de alto valor en sistemas diseñados alrededor de datos reales, flujos de trabajo reales y resultados de negocio medibles.

Nuestro equipo de más de 250 especialistas en IA y datos entregó más de 300 despliegues en producción para líderes tecnológicos globales, empresas Fortune 500 y plataformas que definen su categoría. Ofrecemos servicios end-to-end, desde la estrategia inicial hasta el despliegue completo.

La mayoría de las iniciativas de IA no fracasan porque el modelo sea imposible, sino porque el entorno real es más difícil que la demo: datos fragmentados, plataformas legacy, restricciones de governance y ningún camino claro hacia la adopción. Marvik está construida específicamente para cerrar esa brecha, combinando expertise técnico en sistemas agénticos, analítica predictiva, IA generativa, robótica y automatización, computer vision e ingeniería de datos con la disciplina de ingeniería necesaria para llevar a producción y hacernos cargo de lo que construimos.

Partnerships estratégicos con NVIDIA, Anthropic, OpenAI, Oracle, Microsoft, AWS, Google Cloud y ElevenLabs nos dan acceso directo a tooling avanzado e infraestructura enterprise, lo que se traduce en ciclos de despliegue más rápidos y bases listas para producción desde el día uno.

Conocé más en **www.marvik.ai**

Resumen ejecutivo

Series de tiempo: guía práctica de predicción

El forecasting funciona mejor como un portafolio de herramientas ajustado a la forma de cada problema que como un único modelo por defecto. Esta guía recorre todo el panorama, desde la estadística clásica hasta los modelos fundacionales que atravesaron dos generaciones completas entre 2023 y 2026, con un objetivo: ayudar a un equipo a elegir la herramienta correcta para el problema que tiene enfrente, y reconocer cuándo es la equivocada.

El libro abre con un mapa de ruta: una tabla y un árbol de decisión que llevan al lector desde su situación real (cuántas series, cuánta historia, variables externas o no, ceros o no, explicabilidad requerida o no) hasta el capítulo que la responde. Quien tenga poco tiempo puede empezar y terminar ahí.

Parte uno: **los fundamentos**
Descomposición, estacionariedad y diagnóstico: el trabajo que determina si se puede confiar en todo lo que se construya después. Luego las herramientas clásicas (ARIMA/SARIMA), la familia boosting (XGBoost, LightGBM, CatBoost) para problemas donde el precio, las promociones o el clima mueven el resultado, y los modelos globales, que entrenan un solo modelo sobre miles de SKUs compartiendo señal entre series. Dos capítulos cubren la disciplina que hace real todo lo anterior: backtesting en lugar de un único split afortunado, y cuantificación de la incertidumbre (intervalos clásicos, Bayesianos y conformal prediction), porque un pronóstico con intervalo es una decisión, mientras que un número solo es una apuesta.

Parte dos: **la frontera, puesta a prueba en lugar de anunciada.**
Los modelos fundacionales pronostican una serie que nunca vieron, sin entrenamiento alguno. La guía cubre las dos generaciones: primero TimeGPT, Chronos y MOIRAI, y luego Chronos-2, Moirai 2.0, TimesFM 2.5 y Sundial, que agregan soporte multivariado y covariables mientras se vuelven mucho más chicos y rápidos. En lugar de tomar esas promesas por ciertas, la guía los compara cara a cara contra sus propios baselines de SARIMA y boosting, y reporta todos los resultados, incluido el incómodo, donde el modelo de boosting quedó último. No porque los modelos de árboles sean débiles, sino porque recibió un set de features pobre. Esa es la verdadera lección del capítulo: un modelo fundacional no te pide nada, mientras que un modelo de boosting nunca es mejor que las features que construís para él.

Después argumenta el otro lado, en detalle: el caso de que en dominios con estacionalidad fuerte y bien definida un método estadístico simple puede igualar o superar a un modelo fundacional; el argumento de que estos modelos aprenden forma estadística pero no causa, lo que limita cuánto generalizan realmente; la dependencia de la calidad zero-shot respecto del dominio de pre-entrenamiento; y la explicabilidad, donde a nivel de una predicción individual siguen siendo una caja negra, lo que los descarta cuando un stakeholder necesita una razón y no un número.

La recomendación operativa que surge de todo esto es modesta: antes de invertir en feature engineering, corré un modelo fundacional como la referencia a superar. Si un modelo entrenado con tus propios datos no logra superarlo, ese trabajo nunca fue necesario. Si lo logra, el modelo fundacional ya cumplió su función: es el baseline que justifica el esfuerzo.

Resumen ejecutivo

Los capítulos finales

Abordan lo que ningún modelo resuelve automáticamente (demanda intermitente, cold start, detección de anomalías) y los errores metodológicos que, en la práctica profesional, cuestan más que los técnicos.

La conclusión, tanto para lectores técnicos como de negocio: cualquiera sea el modelo que hoy es estado del arte, será reemplazado en un año o dos. Lo que decide si se puede confiar en un pronóstico (diagnosticar la serie antes de modelarla, validar con backtesting en lugar de un único split afortunado, y adjuntar un intervalo a cada número) no depende de qué modelo esté de moda.

Contenidos

Mapa de ruta 9

1 La anatomía de una serie de tiempo 11

1.1 Introducción 11

1.2 Nuestro caso de estudio: pasajeros aéreos 11

1.3 Deconstruyendo la serie 12

1.3.1 El dilema: ¿aditivo o multiplicativo? 12

1.4 Análisis de residuos 13

1.5 Conclusión 13

2 Estacionariedad y análisis exploratorio 14

2.1 Introducción: el problema de comparar peras con manzanas 14

2.2 El diagnóstico: ¿qué nos dicen los datos? 14

2.3 El tratamiento práctico 15

2.3.1 Paso 1: de números absolutos a porcentajes 15

2.3.2 Paso 2: la tasa de crecimiento 16

2.4 La prueba formal: test de Dickey-Fuller aumentado (ADF) 17

2.4.1 ¿Cómo funciona? 17

2.4.2 Resultados en nuestros datos 17

2.5 Conclusión 18

3 El arsenal clásico: ARIMA 19

3.1 Introducción 19

3.2 Los bloques fundamentales: AR y MA 19

3.2.1 AR: el pasado predice el futuro 19

3.2.2 MA: los shocks se propagan en el tiempo 19

3.2.3 Por qué importan ambos 20

3.3 Referencia rápida de identificación 20

3.4 ARIMA: manejando la tendencia 20

3.5 SARIMA: incorporando la estacionalidad 21

3.6 ARIMAX: agregando variables externas 22

3.7 El atajo práctico: `auto_arima` 22

3.8 Verificación diagnóstica 22

3.9 Conclusión 23

4 Series de tiempo como regresión supervisada 24
4.1 Introducción: rompiendo el paradigma 24
4.2 El caso de estudio: ventas de un gran retailer 24
4.3 Feature engineering: enseñando tiempo a la máquina 24
4.3.1 1. Features de calendario 24
4.3.2 2. Lags 25
4.3.3 3. Ventanas móviles 25
4.3.4 El dataset resultante 25
4.4 La familia boosting: no solo XGBoost 25
4.5 Feature engineering avanzado 26
4.6 Estrategias de pronóstico multi-step 26
4.7 El duelo: XGBoost vs ARIMA 27
4.7.1 ¿Por qué comparar? 27
4.7.2 ¿Cómo medimos el éxito? 27
4.7.3 Resultados visuales 27
4.7.4 Análisis de resultados 28
4.7.5 ¿Qué features fueron más importantes? 28
4.7.6 Más allá del feature importance: SHAP 28
4.8 Hiperparámetros esenciales 29
4.9 Guía de selección: ¿qué modelo debo usar? 29
4.10 Conclusión 29

5 Validación y métricas de error 30
5.1 Introducción: el peligro del “Lucky Split” 30
5.2 La solución: backtesting 30
5.3 Métricas de error: ¿cómo medimos el éxito? 30
5.3.1 1. MAE (mean absolute error) 31
5.3.2 2. RMSE (root mean squared error) 31
5.3.3 3. MAPE (mean absolute percentage error) 32
5.3.4 4. WMAPE (weighted MAPE) 32
5.4 Diagnóstico de errores: de los residuos a las features 32
5.4.1 ¿Cómo leer este gráfico? 33
5.4.2 De residuos a features: capitalizando los errores 33
5.5 Conclusión 33

6 Navegando la incertidumbre 34
6.1 Introducción: el pronóstico del clima 34
6.2 Intervalos de confianza clásicos 34
6.2.1 ¿Qué es un intervalo de confianza? 34
6.2.2 ¿Cómo se calculan? 34

6.2.3 Los supuestos 35
6.2.4 La interpretación en escala logarítmica 35
6.2.5 El contrafactual: intervalos sobre datos crudos vs transformados 36
6.3 Forecast Bayesiano: la distribución completa 36
6.3.1 ¿Qué es el enfoque Bayesiano? 36
6.3.2 Resultados 37
6.3.3 Ventajas del enfoque Bayesiano 37
6.3.4 Limitaciones 37
6.4 Conformal Prediction: el enfoque moderno 38
6.4.1 La idea en 3 pasos 38
6.4.2 Ejemplo numérico 38
6.4.3 El supuesto de intercambiabilidad 39
6.4.4 Resultados: Conformal vs paramétrico 39
6.5 Comparación y guía de selección 40
6.6 Conclusión 41

7 Modelos globales 42
7.1 Introducción: el problema de los 10,000 productos 42
7.2 La solución: modelos globales 42
7.2.1 El formato de datos: Panel 42
7.3 El experimento: local vs global 43
7.3.1 Diseño del experimento 43
7.3.2 Velocidad 43
7.4 Resultados de precisión 43
7.4.1 El fenómeno del Cross-Learning 43
7.5 Cuándo global, cuándo local, cuándo híbrido 44
7.6 Evaluación más allá del MAE 45
7.7 Clustering de series antes de entrenar 45
7.8 Ensembling: la ganancia más simple 46
7.9 Retraining en producción 46
7.10 Conclusión 46

8 Modelos Fundacionales: teoría 47
8.1 Introducción: más allá del entrenamiento 47
8.2 El problema del Cold Start 47
8.2.1 ¿Cuándo ocurre? 47
8.2.2 ¿Por qué los métodos tradicionales fallan? 47
8.2.3 Estrategias modernas 48
8.3 Zero-shot forecasting: predecir sin entrenar 48
8.3.1 Nuestro experimento 48
8.4 Transformers para series de tiempo 49
8.4.1 ¿Qué es la atención? 49

8.4.2 Componentes clave de un Transformer para series de tiempo 49
8.5 Los Modelos Fundacionales 50
8.5.1 ¿Cómo se entrenan? 50
8.5.2 El ecosistema de Modelos Fundacionales 50
8.5.3 Tendencias del campo 51
8.6 Conclusión 51

9 Modelos Fundacionales en la práctica 52
9.1 Introducción 52
9.2 Caso 1: TimeGPT (Nixtla) 52
9.2.1 Características 52
9.2.2 Resultados 53
9.2.3 Limitaciones 53
9.3 Caso 2: Chronos (Amazon) 53
9.3.1 Características 53
9.4 Caso 3: MOIRAI (Salesforce) 54
9.4.1 Características 54
9.5 El benchmark consolidado 55
9.5.1 Interpretación de los resultados 55
9.6 La otra cara de la moneda: limitaciones y críticas 56
9.6.1 Crítica 1: no siempre superan a modelos simples 56
9.6.2 Crítica 2: el problema de la generalización 56
9.6.3 Crítica 3: zero-shot depende del dominio de pre-entrenamiento 56
9.6.4 Crítica 4: covariables y variables exógenas 56
9.7 ¿Cuándo Foundation Model, cuándo no? 57
9.8 Conclusión 57

10 Desafíos del mundo real 58
10.1 Introducción 58
10.2 Demanda intermitente: el problema de los ceros 58
10.2.1 El problema real 58
10.2.2 El contrafactual: ¿por qué fallan los modelos del Cap 3-4? 58
10.2.3 La solución: método de Croston-SBA 59
10.2.4 Nota sobre métricas 61
10.3 Cold Start: el producto fantasma 61
10.3.1 El problema 61
10.3.2 Estrategia 1: transferencia por similitud 62
10.3.3 Estrategia 2: modelo global con Cross-Learning 62
10.3.4 Estrategia 3: zero-shot con Modelos Fundacionales 62
10.3.5 ¿Cuál elegir? 63
10.4 Detección de anomalías: encontrar lo raro 64
10.4.1 ¿Por qué importa? 64

10.4.2 El contrafactual: el costo de ignorar anomalías 64
10.4.3 Tres métodos de detección 64
10.5 Errores frecuentes en la práctica del forecasting 65
10.5.1 Errores en el manejo de datos 65
10.5.2 Errores en la selección de modelos 66
10.5.3 Errores en la comunicación de resultados 66
10.6 Reflexiones finales 66
Próximos pasos 67
Bibliografía 68

CAPÍTULO 0

Mapa de ruta

Esta sección es un atajo: si ya tenés un problema concreto y querés saber por dónde arrancar, usala como referencia rápida. Si preferís recorrer la guía completa, salteala y volvé cuando la necesites.

Antes de elegir un modelo: si tus datos tienen valores faltantes, anomalías o frecuencia irregular, empezá por el Capítulo 2 (diagnóstico) y la sección de anomalías del Capítulo 10. Ningún modelo compensa datos sucios.

¿Qué problema, qué herramienta?

Situación	¿Exóg.?	Método recomendado	Capítulo
1 serie, >2 ciclos estacionales, sin variables externas	No	SARIMA	3
1 serie, con precio/promos/clima u otras variables externas	Sí	XGBoost / Boosting	4
1 serie, pocos datos (<1 ciclo estacional)	—	Modelos Fundacionales	8–9
10–50 series similares	Sí/No	XGBoost (individual) o Modelo Global	4, 7
50+ series	Sí/No	Modelo Global	7
Demanda intermitente (muchos ceros)	—	Croston	10
Producto nuevo sin historia	—	Transfer / Global / Foundation	7,8,9,10
Necesito intervalos de confianza	—	Cap 6 (aplica a cualquier modelo)	6
Detección de anomalías	—	Isolation Forest / Z-Score	10

Árbol de decisión rápido

¿Tenés variables exógenas (precio, clima, promociones, calendario)?

- **Sí** → XGBoost/Boosting (Cap 4). Si además tenés 50+ series → Modelo Global (Cap 7).
- **No** → ¿Cuántos datos tenés por serie?
  - Más de 2 ciclos estacionales → SARIMA (Cap 3) o XGBoost (Cap 4)
    Menos de 1 ciclo → Modelos Fundacionales (Cap 8–9) o Cold Start (Cap 10)

¿Cuántas series?

- **1–10:** modelo individual por serie
- **10–50:** si son similares entre sí → Global (Cap 7); si son muy distintas → individual
- **50+:** Modelo Global (Cap 7)

Casos especiales:

- ¿Muchos ceros? → Croston (Cap 10)
- ¿Necesitás incertidumbre? → Cap 6 (aplica a cualquier modelo)
- ¿Horizonte largo (>3 ciclos)? → La incertidumbre domina; priorizá Cap 6

Nota sobre explicabilidad

Si tu stakeholder necesita entender por qué el modelo predice lo que predice:

- **SARIMA:** coeficientes directamente interpretables ("el efecto del mes anterior es 0.7").
- **XGBoost:** explicable vía SHAP ("el precio fue la variable más importante").
- **Modelos Fundacionales:** caja negra; difícil de explicar a nivel individual.

Qué vas a aprender en cada capítulo

1. Anatomía de una serie de tiempo — Tendencia, estacionalidad, residuos. Cómo descomponer y entender cualquier serie.

2. Diagnóstico de la serie — Tests de estacionariedad (ADF), ACF/PACF, transformaciones. El toolkit de diagnóstico.

3. Modelos clásicos (ARIMA) — AR, MA, ARIMA, SARIMA. Cuándo y cómo usarlos para series univariadas.

4. Machine Learning (Boosting) — XGBoost, LightGBM, CatBoost. Feature engineering y variables exógenas.

5. Validación — Splits temporales, backtesting, métricas. Cómo evaluar honestamente.

6. Incertidumbre — Intervalos de predicción, conformal prediction, enfoques Bayesianos.

7. Modelos globales — Un modelo para muchas series. Cross-learning y escala.

8. Modelos Fundacionales I — Chronos, MOIRAI. Modelos pre-entrenados para forecasting zero-shot.

9. Modelos Fundacionales II — Evaluación crítica, benchmarks y limitaciones prácticas.

10. Desafíos del mundo real — Demanda intermitente, cold start, anomalías y una matriz de decisión integral.

Para una evaluación detallada de cuándo los Modelos Fundacionales aportan valor sobre las alternativas, ver el Capítulo 9. Para problemas especiales (demanda intermitente, cold start, anomalías), ver el Capítulo 10.

CAPÍTULO 1

La anatomía de una serie de tiempo

1.1 Introducción

Bienvenido al fascinante mundo de las series de tiempo. A diferencia de los datos de corte transversal, donde el orden de las observaciones no importa, en una serie de tiempo **el orden es fundamental**. El pasado dicta, en gran medida, el presente.

Una serie de tiempo es una secuencia de puntos de datos indexados en orden cronológico. Matemáticamente, la denotamos como:

$$\{Y_t\}_{t=1}^{T} = \{y_1, y_2, \ldots, y_T\}$$

1.2 Nuestro caso de estudio: pasajeros aéreos

Para este primer capítulo, analizaremos el dataset clásico de Box & Jenkins: **Monthly Airline Passenger Numbers 1949-1960**. Este conjunto de datos contiene el número de pasajeros mensuales de una aerolínea estadounidense.

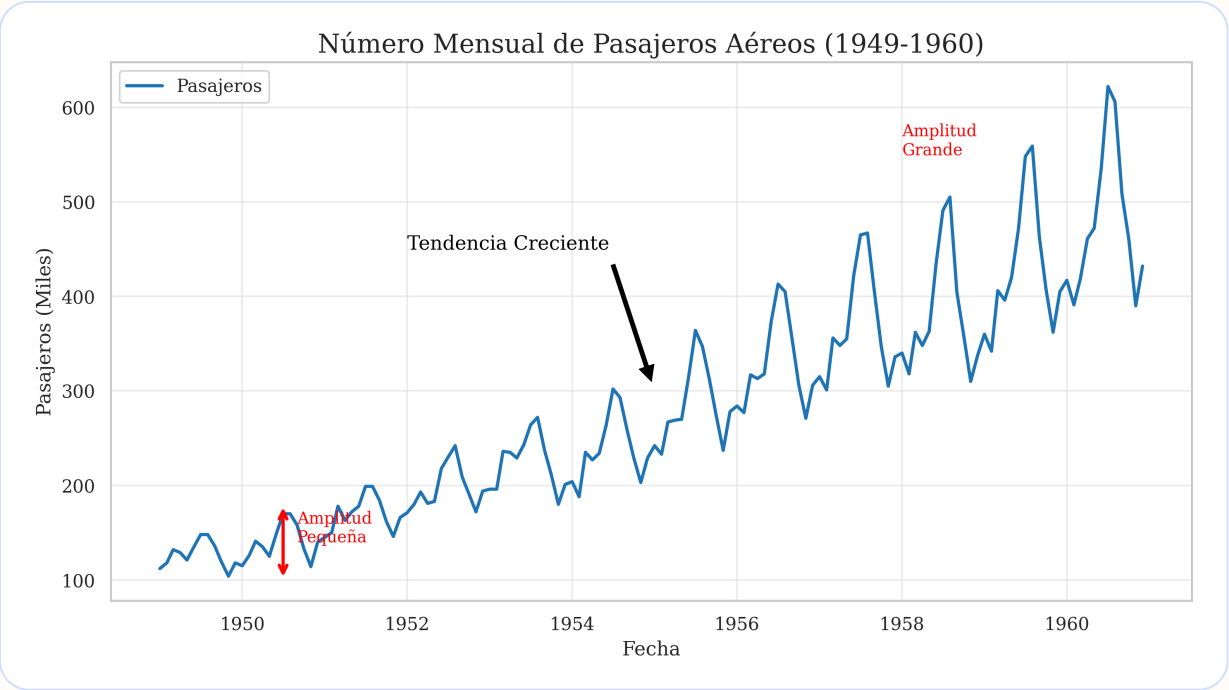


Figura 1.1: Evolución mensual de pasajeros (1949-1960)

Al observar la Figura 1.1, un economista entrenado nota inmediatamente tres cosas:

1. Tendencia positiva: El número de pasajeros crece consistentemente a lo largo de la década.

2. Estacionalidad fuerte: Hay un patrón cíclico que se repite cada 12 meses (picos en verano, valles en invierno).

3. Heterocedasticidad (varianza no constante): Note cómo las oscilaciones estacionales se vuelven más amplias a medida que sube la serie. La diferencia entre el verano y el invierno de 1960 es mucho mayor que la de 1950.

1.3 Deconstruyendo la serie

Podemos pensar en una serie de tiempo como la suma (o producto) de tres componentes invisibles:

- **Tendencia** (T_t)**:** La dirección a largo plazo.
- **Estacionalidad** (S_t)**:** El patrón repetitivo de corto plazo.
- **Residuo** (R_t)**:** El ruido aleatorio o irregular que el modelo no explica.

1.3.1 El dilema: ¿aditivo o multiplicativo?

Esta es la primera decisión crucial de modelado.

Modelo aditivo: Asume que la magnitud de la estacionalidad es constante, sin importar el nivel de la serie.

$$Y_t = T_t + S_t + R_t$$

Modelo multiplicativo: Asume que la estacionalidad crece proporcionalmente al nivel de la serie.

$$Y_t = T_t \times S_t \times R_t$$

Veamos qué sucede cuando intentamos forzar ambos modelos a nuestros datos.

Aplicamos la descomposición multiplicativa a nuestra serie de pasajeros. La Figura 1.2 muestra el resultado: cuatro paneles que revelan las "capas" ocultas de los datos.

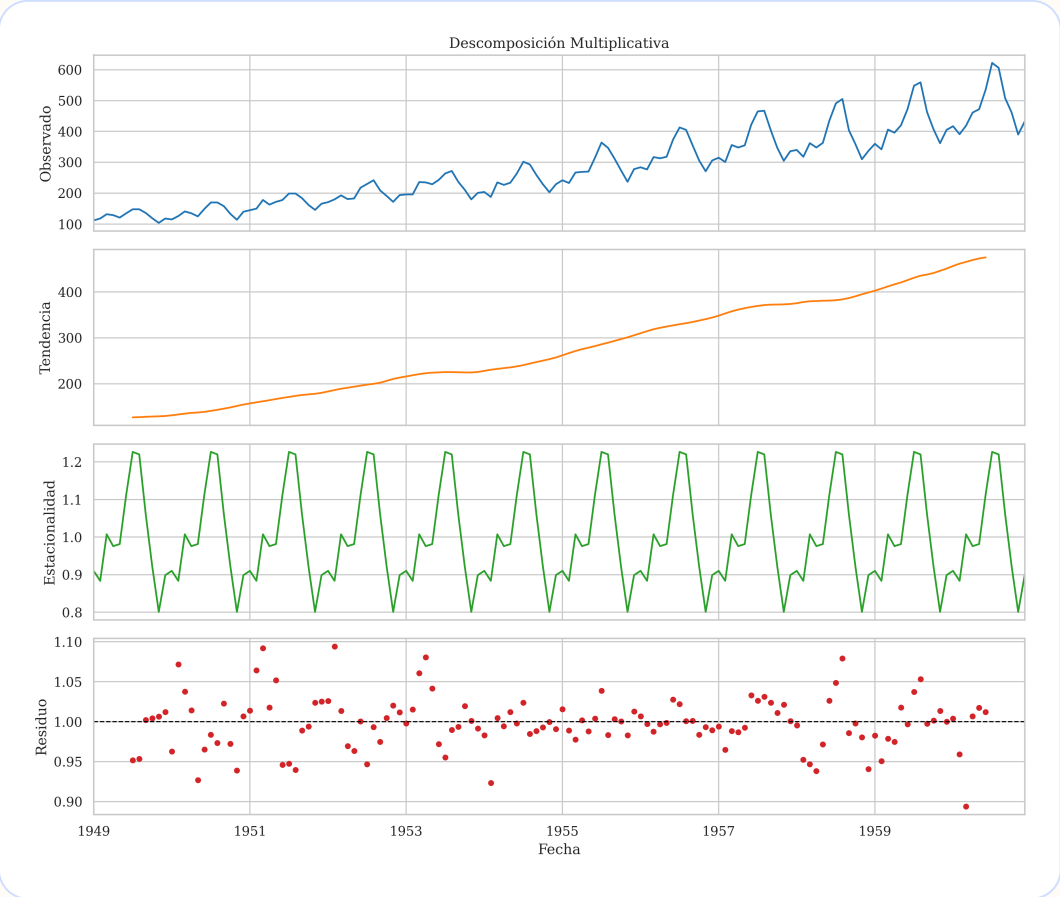


Figura 1.2: Descomposición multiplicativa. De arriba a abajo: (1) la serie original, (2) la tendencia extraída —dirección a largo plazo—, (3) el componente estacional —patrón que se repite cada 12 meses, con picos en verano y valles en invierno—, y (4) los residuos —oscilan alrededor de 1.0 con varianza estable, indicando que el modelo capturó la estructura de los datos—.

Pero, ¿cómo sabemos que la descomposición multiplicativa es la correcta y no la aditiva? La respuesta está en los residuos: lo que "sobra" después de extraer tendencia y estacionalidad. Si el modelo es bueno, los residuos deberían ser **ruido blanco** (aleatoriedad pura, sin patrones).

1.4 Análisis de residuos

Comparemos lado a lado los residuos de ambos modelos. La diferencia es reveladora.

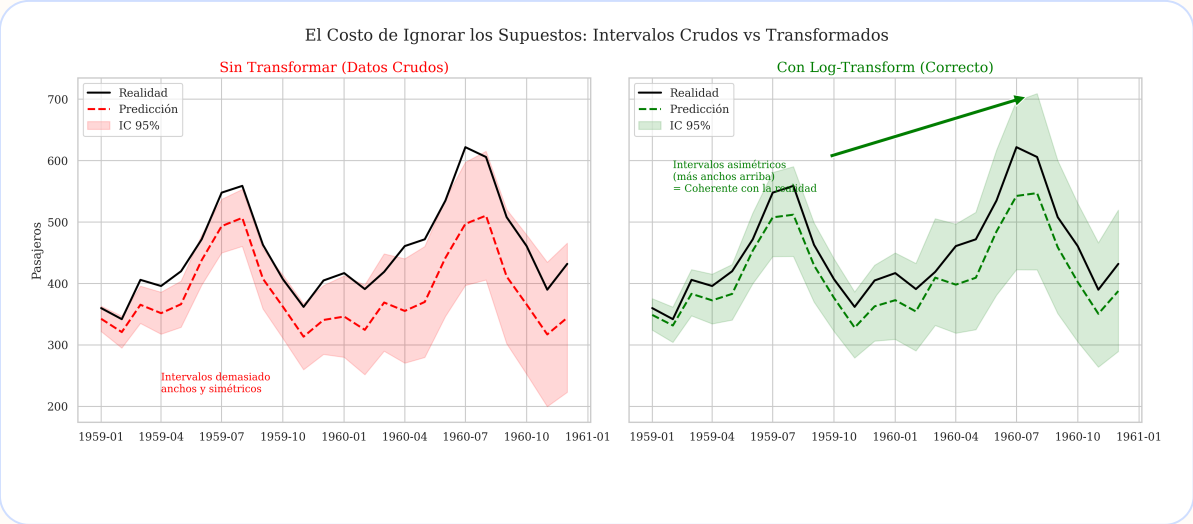


Figura 1.3: Comparación directa de residuos. **Izquierda (aditivo):** los errores forman un abanico que crece con el tiempo, de ±15 pasajeros en 1950 a ±60 en 1960, un aumento de 3x. Las flechas señalan esta expansión. **Derecha (multiplicativo):** los errores se mantienen dentro de una banda estable del ±5% durante toda la década. Las flechas confirman que la banda es la misma al inicio y al final de la serie.

En el panel izquierdo de la Figura 1.3 (modelo aditivo), observamos un **patrón de abanico**: los errores son pequeños al principio (±15 pasajeros) y enormes al final (±60 pasajeros). Esto indica que el modelo aditivo falló en capturar la estructura de los datos. ¿Por qué? Porque asumió que la estacionalidad tiene siempre la misma magnitud absoluta (ej: "en verano siempre suben 50 pasajeros"), pero en realidad la subida es proporcional al nivel ("en verano siempre suben un 30%"). A medida que la serie crece de 100 a 500 pasajeros, el error crece proporcionalmente.

En el panel derecho (modelo multiplicativo), los residuos oscilan dentro de una banda estable del ±5% durante toda la serie. Tanto en 1950 como en 1960, las flechas confirman que la varianza no cambia. Esto se parece mucho más a ruido aleatorio: el modelo multiplicativo "entendió" que la estacionalidad crece proporcionalmente al nivel.

1.5 Conclusión

Para series de tiempo económicas y de negocios (ventas, demanda, tráfico), donde el crecimiento suele ser porcentual (exponencial) y la volatilidad aumenta con el tamaño, **el modelo multiplicativo es generalmente la elección correcta**.

En el próximo capítulo, daremos el primer paso hacia la predicción: introduciremos el concepto de **Estacionariedad** , el requisito fundamental que nuestros datos deben cumplir antes de construir cualquier modelo de pronóstico en el Capítulo 3.

CAPÍTULO 2

Estacionariedad y análisis exploratorio

2.1 Introducción: el problema de comparar peras con manzanas

Imaginá que querés predecir la altura de un adolescente basándote en su altura de los años pasados. Es difícil, porque el adolescente está en pleno "estirón". Su altura media cambia cada año.

En cambio, predecir la altura de un adulto es más fácil: su altura media es constante. Si un día mide un poco más, es probablemente porque lleva zapatos de suela gruesa (ruido), pero volverá a su media.

En series de tiempo, llamamos a este estado de "adulto estable" **Estacionariedad**.

- **No estacionaria (adolescente):** La media y la varianza cambian con el tiempo. (Ej: pasajeros de avión en los 50s).
- **Estacionaria (adulto):** La media y la varianza son constantes. Los patrones estadísticos no dependen de cuándo miremos la serie.

Nuestro objetivo en este capítulo es convertir nuestra serie "adolescente" en una serie "adulta" para poder modelarla.

2.2 El diagnóstico: ¿qué nos dicen los datos?

Antes de aplicar cualquier tratamiento, necesitamos un diagnóstico. El instrumento principal es el **Correlograma (ACF)**: la Función de Autocorrelación.

La ACF nos dice cuánto se parece la serie a sí misma en el pasado. Cada barra representa la correlación entre la serie y su versión retrasada k periodos ("lag k"):

- **Lag 1:** ¿El valor de este mes se parece al del mes pasado?
- **Lag 12:** ¿El valor de este mes se parece al de hace un año?

Si la serie tiene tendencia, la ACF decae muy lentamente ("el pasado lejano aún importa mucho"). Si la serie es estacionaria, la ACF cae a cero rápidamente ("solo el pasado reciente importa").

En el panel superior (ACF), las barras azules bajan lentísimo. Esto confirma lo que vimos en el Capítulo 1: la serie tiene una tendencia fuerte. El número de pasajeros de hoy depende del de hace 1 mes, 2 meses, 5 meses, incluso 12 meses. La serie tiene "memoria infinita", lo cual es un síntoma clásico de no estacionariedad.

En el panel inferior (PACF), la **Autocorrelación Parcial** nos da información complementaria. Mientras que la ACF mide la correlación total entre Y_t e Y_{t-k} (incluyendo la influencia indirecta de todos los lags intermedios), la PACF aísla la correlación directa de cada lag, eliminando el efecto de los intermedios. En nuestro caso, la PACF muestra una barra muy alta en el lag 1 y luego cae abruptamente. Esto indica que la relación principal es con el mes inmediato anterior: una vez que conocemos Y_{t-1}, los lags 2, 3, etc. aportan poca información adicional por sí solos.

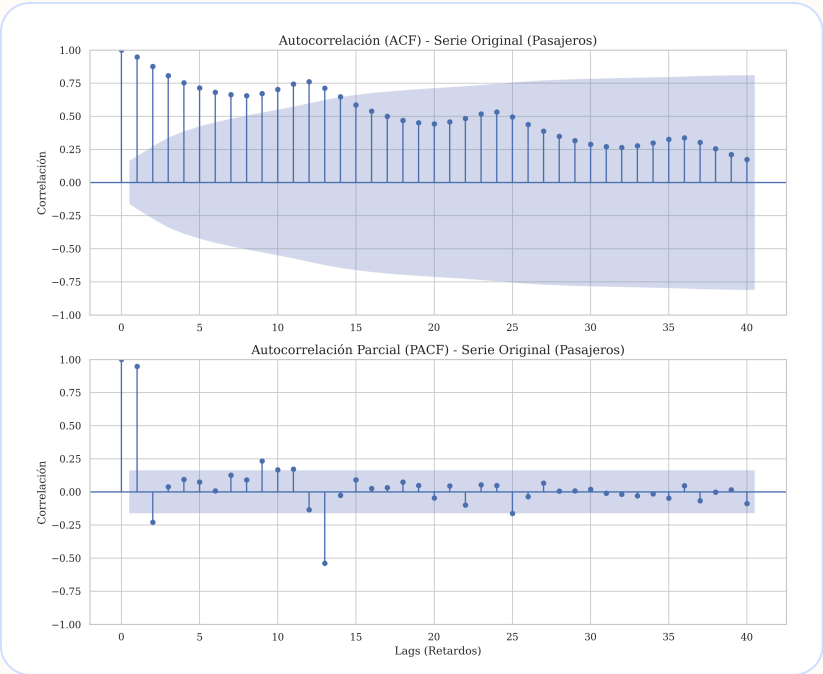


Figura 2.1: ACF de la serie original de Pasajeros. **Arriba (ACF):** Las barras decaen muy lentamente, indicando una fuerte tendencia. Las barras que sobresalen de la zona azul sombreada son "estadísticamente significativas". **Abajo (PACF):** La Autocorrelación Parcial muestra la correlación directa con cada lag, eliminando la influencia de los intermedios.

Este patrón (ACF que decae lentamente + PACF que corta en lag 1) es la huella clásica de un proceso autoregresivo de orden 1 con tendencia. También notará un pico negativo significativo alrededor del lag 13: esta es la firma estacional que abordaremos en el Paso 2.

2.3 El tratamiento práctico

2.3.1 Paso 1: de números absolutos a porcentajes

En el Capítulo 1 vimos que la varianza de la serie crecía con el nivel (heterocedasticidad). En economía, cuando algo crece exponencialmente, la herramienta estándar es el **Logaritmo Natural**.

¿Por qué funciona? Porque el logaritmo convierte un crecimiento exponencial (donde los cambios son proporcionales al nivel) en un crecimiento lineal (donde los cambios son constantes). En otras palabras, transforma una curva que se acelera en una línea más recta. Veamos con un ejemplo numérico:

- Si los pasajeros pasan de 100 a 110 (crecimiento del 10%): $\log(110) - \log(100) = 4.70 - 4.61 = 0.095$
- Si los pasajeros pasan de 400 a 440 (crecimiento del 10%): $\log(440) - \log(400) = 6.09 - 5.99 = 0.095$

En ambos casos, aunque los números absolutos son muy distintos (10 vs 40 de diferencia), el cambio logarítmico es idéntico (0.095). El logaritmo "normaliza" las diferencias: un 10% es un 10%, sin importar si partimos de 100 o de 400.

Visualmente, la serie transformada (Figura 2.2) ya no muestra el efecto "abanico" del Capítulo 1: las oscilaciones estacionales tienen amplitud pareja tanto en 1950 como en 1960. Hemos curado la heterocedasticidad.

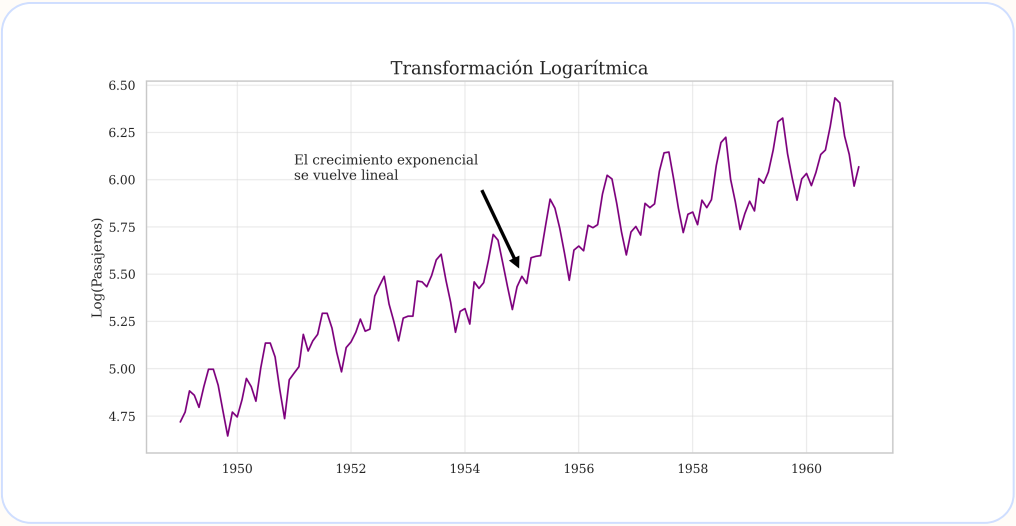


Figura 2.2: Transformación logarítmica. La curva exponencial se "aplana" en una línea más recta. Note que las oscilaciones estacionales ahora tienen amplitud más uniforme (compare con la Figura 1.1 del Capítulo 1).

2.3.2 Paso 2: la tasa de crecimiento

Hemos estabilizado la varianza, pero la serie todavía sube: aún tenemos tendencia. Para eliminarla, en vez de predecir el nivel de la serie, vamos a predecir el **cambio** entre un periodo y el anterior.

Matemáticamente:

$$\Delta \log(Y_t) = \log(Y_t) - \log(Y_{t-1}) \approx \text{Tasa de Crecimiento Porcentual}$$

Ejemplo numérico: Si $\log(Y_t) = 5.50$ y $\log(Y_{t-1}) = 5.45$, entonces $\Delta = 0.05$, que se interpreta como un crecimiento aproximado del 5%. Esto es algo que cualquier CEO entiende: "El mes pasado crecimos un 5%".

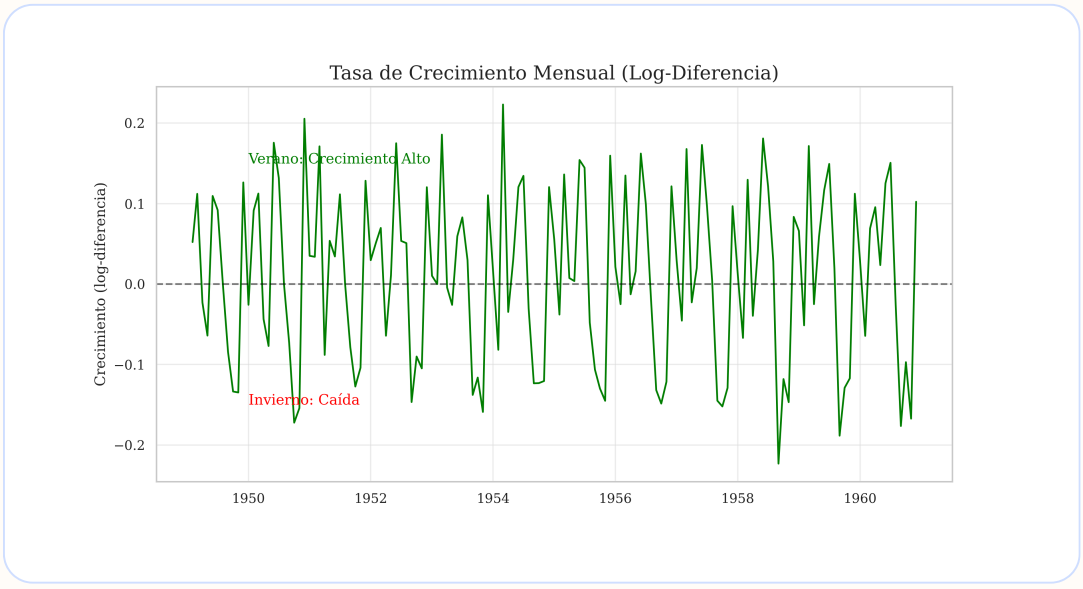


Figura 2.3: Tasa de Crecimiento Mensual (primera diferencia del logaritmo). La serie ahora oscila alrededor de cero: los picos positivos corresponden a los meses de verano (alta demanda) y los valles negativos al invierno. Ya no hay tendencia ascendente.

¡Eureka! En la Figura 2.3, la tendencia ha desaparecido. La serie sube y baja alrededor del 0%.
En verano crecemos mucho (picos positivos) y en invierno caemos (picos negativos). Pero aún vemos un patrón repetitivo: la estacionalidad.

2.4 La prueba formal: test de Dickey-Fuller aumentado (ADF)

Hasta ahora, nuestro diagnóstico ha sido visual ("parece estacionaria"). Pero en ciencia necesitamos un veredicto formal. El **Test de Dickey-Fuller Aumentado (ADF)** es el estándar para esto.

2.4.1 ¿Cómo funciona?

El test plantea dos hipótesis:

- **H_0 (Hipótesis Nula):** La serie tiene una raíz unitaria, es decir, NO es estacionaria. Hay tendencia o comportamiento no estable.
- **H_1 (Hipótesis Alternativa):** La serie SÍ es estacionaria.

El test calcula un **p-value**: un número entre 0 y 1 que indica la probabilidad de observar estos datos si H_0 fuera verdadera. La regla de decisión es:

- Si $p < 0.05$: Rechazamos H_0 → La serie es estacionaria.
- Si $p \geq 0.05$: No podemos rechazar H_0 → La serie no es estacionaria.

2.4.2 Resultados en nuestros datos

Aplicamos el test ADF a cada versión de la serie:

1. Serie original (pasajeros crudos): $p = 0.99$. Muy lejos de 0.05. La serie es claramente no estacionaria. El test confirma lo que veíamos: hay tendencia.

2. Tasa de crecimiento (Log + diferenciación): $p = 0.07$. Mejor, pero aún no pasamos el umbral de 0.05. ¿Por qué? Porque eliminamos la tendencia, pero la estacionalidad (el ciclo anual de 12 meses) sigue presente y el test la detecta como no estacionariedad residual.

3. Crecimiento ajustado estacionalmente (Log + Diff + Diff estacional): $p = 0.0002$. ¡Ahora sí! Al aplicar una segunda diferenciación (esta vez con lag 12 para eliminar el ciclo anual), la serie se vuelve completamente estacionaria.

2.5 Conclusión

Hemos aprendido que para modelar series económicas, casi nunca usamos los datos crudos. El camino de transformación es:

$$\text{Pasajeros} \xrightarrow{\text{Log}} \log(\text{Pasajeros}) \xrightarrow{\Delta_1} \text{Tasa de Crecimiento} \xrightarrow{\Delta_{12}} \text{Serie Estacionaria}$$

Cada paso resuelve un problema específico:

- **Logaritmo:** Estabiliza la varianza (heterocedasticidad).
- **Diferenciación (Δ_1):** Elimina la tendencia.

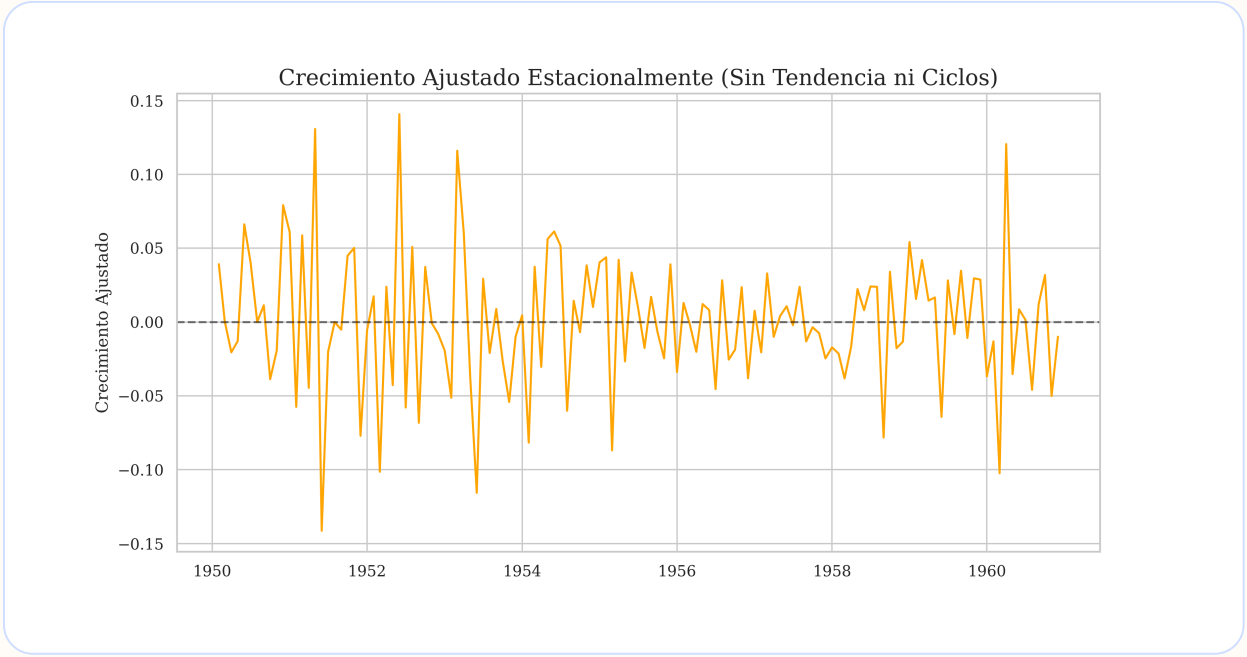


Figura 2.4: Serie final después de Log + Diferenciación + Diferenciación Estacional (lag 12). Esto es puro "ruido" estacionario: no hay tendencia, no hay ciclos predecibles. Es lo que necesitamos para alimentar un modelo ARIMA en el Capítulo 3.

- **Diferenciación Estacional (Δ_{12}):** Elimina el ciclo anual.

Ahora nuestra serie es estable y predecible. En el próximo capítulo, usaremos **ARIMA** para modelar esta serie transformada y generar predicciones del futuro.

CAPÍTULO 3

El arsenal clásico: ARIMA

3.1 Introducción

Este capítulo cubre la familia de modelos más fundamental en series de tiempo: los **modelos de Box-Jenkins**, que culminan en SARIMA. Son estructuras transparentes construidas sobre la idea de que el futuro es una combinación lineal del pasado.

Usaremos el dataset clásico AirPassengers (pasajeros mensuales de una aerolínea, 1949–1960) a lo largo del capítulo. El objetivo es práctico: entender cuándo y cómo usar estos modelos, y sus limitaciones. Para los detalles matemáticos completos (notación de operador de rezago, condiciones formales de estacionariedad/invertibilidad, ecuaciones de Yule-Walker), ver la literatura especializada (ej: Hyndman & Athanasopoulos, *Forecasting: Principles and Practice*).

3.2 Los bloques fundamentales: AR y MA

Todo modelo ARIMA se construye con dos mecanismos simples:

3.2.1 AR: el pasado predice el futuro

Un modelo **Autoregresivo** de orden p predice el valor de hoy como combinación lineal de sus propios valores pasados:

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \epsilon_t \tag{3.1}$$

Los coeficientes $\phi_1, \ldots, \phi_p$ capturan cuán fuertemente la serie "recuerda" su pasado reciente. Para que el modelo sea estable (estacionario), la influencia del pasado debe atenuarse con el tiempo en lugar de amplificarse. Veamos con AR(1): si $\phi_1 = 0.8$, un shock de 10 unidades se atenúa: 10 → 8 → 6.4 → 5.1, hasta desaparecer. Pero si $\phi_1 = 1.1$, el mismo shock se amplifica: 10 → 11 → 12.1 → 13.3, y la serie se aleja indefinidamente de su media. Para AR(1), la condición es $|\phi_1| < 1$. Para modelos de orden superior, la condición involucra las raíces del polinomio característico (en la práctica, el software lo verifica automáticamente).

Firma ACF/PACF: La PACF se corta en el rezago p; la ACF decae gradualmente.

3.2.2 MA: los shocks se propagan en el tiempo

Un modelo de **Media Móvil** de orden q predice el valor de hoy basándose en shocks aleatorios pasados:

$$X_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} \tag{3.2}$$

A diferencia del AR, la memoria de un proceso MA está estrictamente limitada: un shock en el tiempo t tiene efecto cero después de q períodos.

Firma ACF/PACF: La ACF se corta en el rezago q; la PACF decae gradualmente.

3.2.3 Por qué importan ambos

Las series reales típicamente tienen tanto persistencia (AR) como efectos de shocks (MA). Combinarlos da ARMA(p, q), que puede explicar dinámicas complejas con menos parámetros que cualquier componente por separado. Cuando tanto la ACF como la PACF decaen gradualmente sin cortes claros, probablemente se necesita un modelo ARMA. En la práctica, usá AIC/BIC o `auto_arima` para encontrar el (p, q) correcto.

3.3 Referencia rápida de identificación

Firma ACF/PACF	Modelo	Cómo determinar el orden
PACF se corta en lag p, ACF decae	AR(p)	p = lag de corte de PACF
ACF se corta en lag q, PACF decae	MA(q)	q = lag de corte de ACF
Ambas decaen gradualmente	ARMA(p, q)	Búsqueda por grilla AIC/BIC
Picos en el lag estacional s	Agregar términos estacionales $(P, D, Q)_s$	

Tabla 3.1: Identificación de modelos vía patrones ACF y PACF.

3.4 ARIMA: manejando la tendencia

AR y MA asumen que la serie es estacionaria, es decir, su media no cambia con el tiempo. Los datos del mundo real generalmente tienen tendencia.

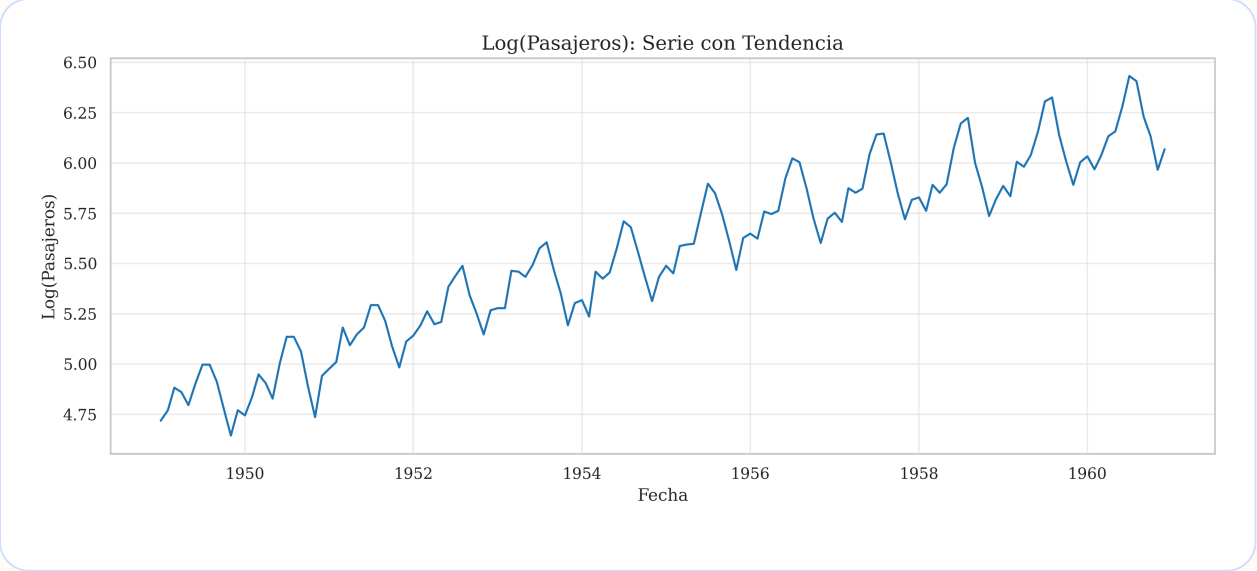


Figura 3.1: Log-pasajeros con tendencia aproximadamente lineal. Esta serie es no estacionaria: aplicar ARMA directamente fallaría.

La solución es la **diferenciación**: en lugar de modelar X_t, modelamos el cambio $\nabla X_t = X_t - X_{t-1}$. Si la tendencia es lineal, la primera diferencia será estacionaria.

Un modelo **ARIMA(p, d, q)** aplica ARMA a la serie diferenciada d veces. En la práctica, d = 1 maneja tendencias lineales y d = 2 maneja tendencias cuadráticas (raro).

3.5 SARIMA: incorporando la estacionalidad

Los pasajeros aéreos tienen picos en verano y valles en invierno, un ciclo que se repite cada s = 12 meses. ARIMA por sí solo no puede capturar esto.

La diferenciación estacional elimina el ciclo estacional:

$$\nabla_s X_t = X_t - X_{t-s} \tag{3.3}$$

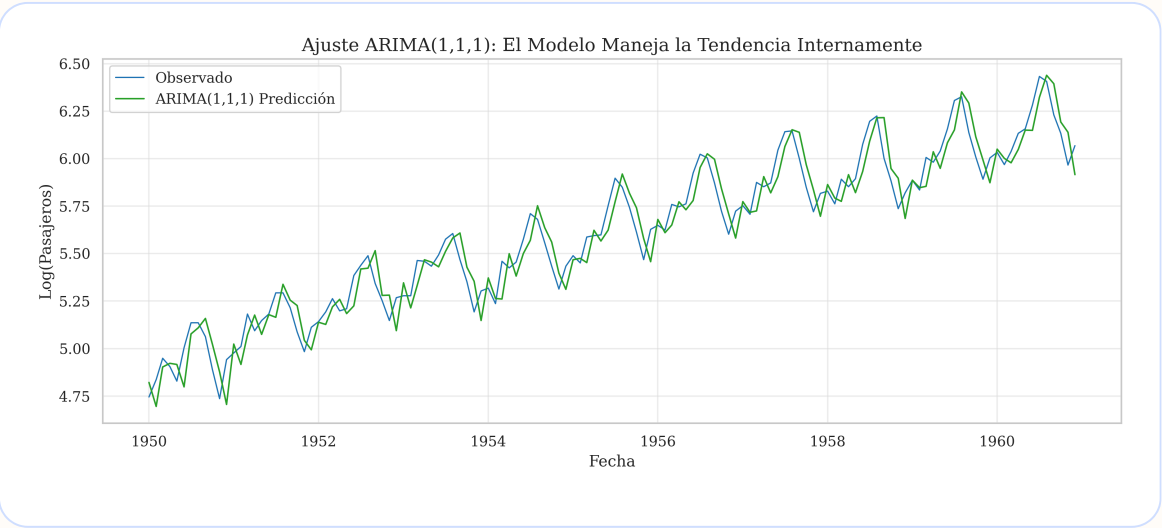


Figura 3.2: Ajuste ARIMA(1,1,1). El modelo maneja la tendencia diferenciando internamente.

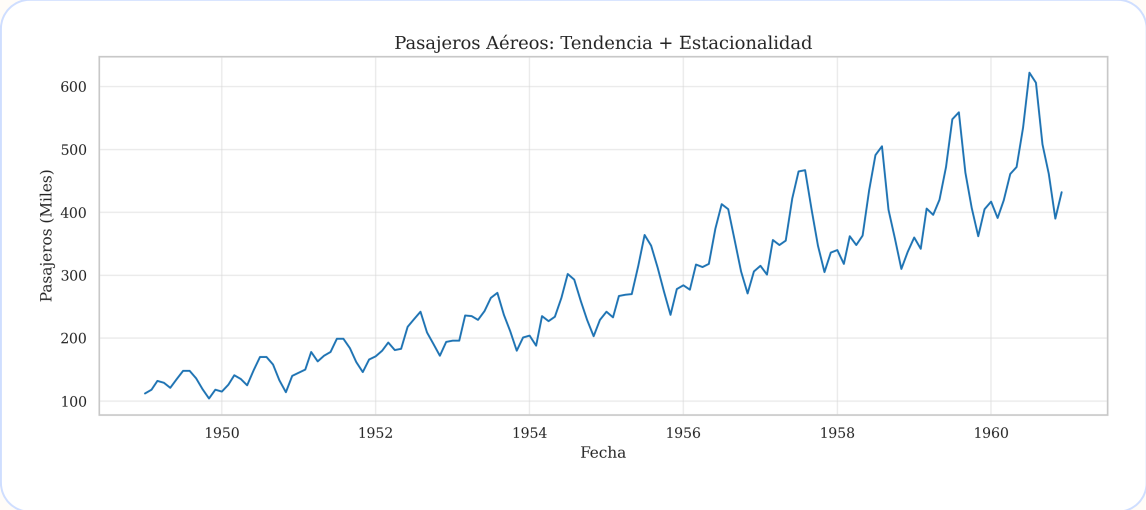


Figura 3.3: Datos con tendencia y estacionalidad. Note el patrón de onda repetitivo cada 12 meses.

Un modelo **SARIMA(p, d, q)(P, D, Q)$_s$** combina:

- **Partes no estacionales (p, d, q):** dinámica de corto plazo (rezagos 1, 2, …).
- **Partes estacionales (P, D, Q)$_s$:** patrones que se repiten cada s períodos (rezagos s, 2s, …).

Identificación estacional: Un pico en la ACF en el rezago s sugiere un término MA Estacional; un pico en la PACF en el rezago s sugiere un término AR Estacional.

3.6 ARIMAX: agregando variables externas

Hasta ahora, usamos solo el pasado de la serie. Un modelo **ARIMAX** agrega regresores externos (precio, promociones, temperatura):

$$\mathrm{Yt} = \beta 0 + \beta 1 \mathrm{Z1,t} + \cdots + \beta \mathrm{k} \mathrm{Zk,t} + \mathrm{ut}, \quad \text{donde ut} \sim \mathrm{ARIMA} \tag{3.4}$$

Advertencia clave: Para pronosticar Y_{T+h}, necesitás los valores futuros de Z_{T+h}. Si son desconocidos, hay que proyectarlos primero, lo que agrega incertidumbre. Para problemas con muchas variables exógenas, el enfoque de ML del Capítulo 4 (XGBoost/Boosting) suele ser más práctico.

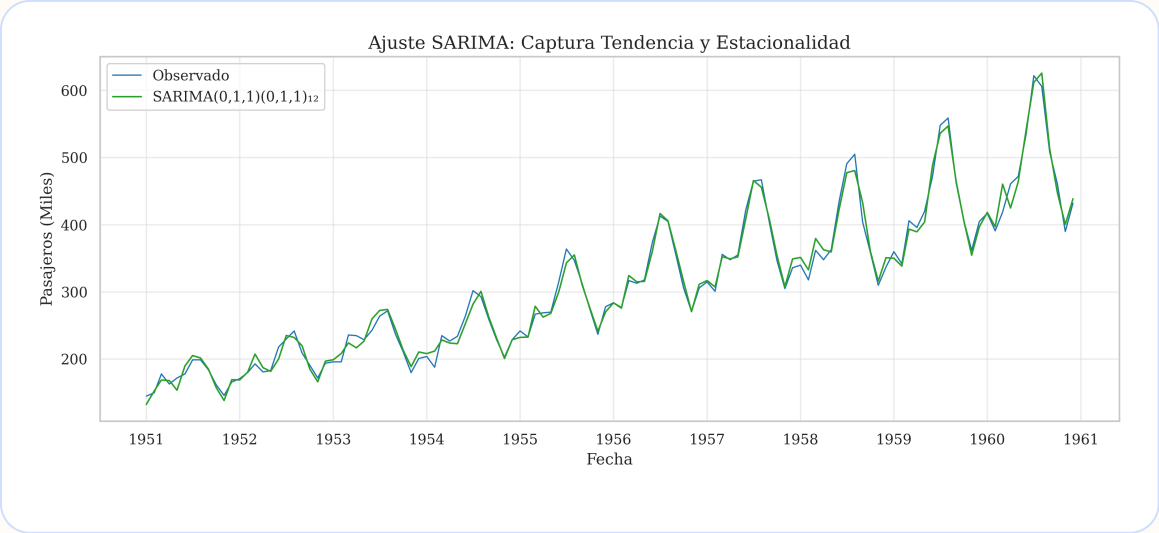


Figura 3.4: Ajuste SARIMA(0, 1, 1)(0, 1, 1)$_{12}$ (el clásico "Airline Model"). Captura tanto la tendencia ascendente como las oscilaciones estacionales.

3.7 El atajo práctico: `auto_arima`

Inspeccionar ACF/PACF manualmente para elegir (p, d, q)(P, D, Q)$_s$ funciona pero es lento y subjetivo. En la práctica, `pmdarima.auto_arima()` automatiza esto buscando sobre modelos candidatos y minimizando AIC. Es un punto de partida confiable que frecuentemente iguala o supera la identificación manual.

3.8 Verificación diagnóstica

Después de ajustar, verificá que el modelo haya extraído toda la información de los datos. Si lo hizo, los residuos deben parecer **ruido blanco**: aleatorios, no correlacionados y aproximadamente normales.

Test de Ljung-Box: Testea si los residuos tienen autocorrelación significativa. Si el p-valor es alto (> 0.05), los residuos son consistentes con ruido blanco y el modelo es válido.

Selección de modelos: Cuando múltiples modelos pasan el diagnóstico, elegí usando Criterios de Información:

$$\mathrm{AIC} = -2\log L + 2k, \qquad \mathrm{BIC} = -2\log L + k\log(T) \tag{3.5}$$

Valores más bajos indican mejor balance entre ajuste y complejidad. En caso de duda, preferí el modelo más simple (parsimonia).

3.9 Conclusión

2. MA: Captura la propagación de shocks (efectos de errores pasados).
El framework SARIMA se construye con cuatro componentes:

1. AR: Captura la memoria (persistencia de valores pasados).
3. I: Maneja la tendencia (mediante diferenciación).
4. S: Maneja la estacionalidad (mediante diferenciación estacional).
SARIMA sigue siendo un baseline fuerte contra el cual deben compararse todos los demás modelos. Sus principales ventajas son la interpretabilidad y la eficiencia en series univariadas con patrones estacionales claros.

Limitaciones prácticas:

- Los pronósticos se degradan rápidamente más allá de 2–3 ciclos estacionales.
- Manejar muchas variables exógenas es incómodo; preferí XGBoost (Capítulo 4).
- Un modelo por serie: no escala a cientos de series (ver Modelos Globales, Capítulo 7).

Para el tratamiento matemático completo de los conceptos introducidos en este capítulo, ver la literatura especializada (ej: Hyndman & Athanasopoulos, *Forecasting: Principles and Practice*).

CAPÍTULO 4

Series de tiempo como regresión supervisada

4.1 Introducción: rompiendo el paradigma

En los capítulos anteriores, tratamos a la serie de tiempo como una secuencia sagrada donde t depende de t − 1. Los modelos de Machine Learning (como XGBoost, Random Forest o Redes Neuronales) no entienden de secuencias. Para ellos, cada fila es independiente.

¿Cómo usamos entonces estos algoritmos poderosos? **Transformamos el problema.** Convertimos una serie de tiempo en una tabla de regresión clásica (X → Y).

4.2 El caso de estudio: ventas de un gran retailer

Para este reto, necesitamos datos más complejos que los pasajeros de avión. Hemos simulado las ventas diarias de un departamento de un gran retailer durante 5 años.

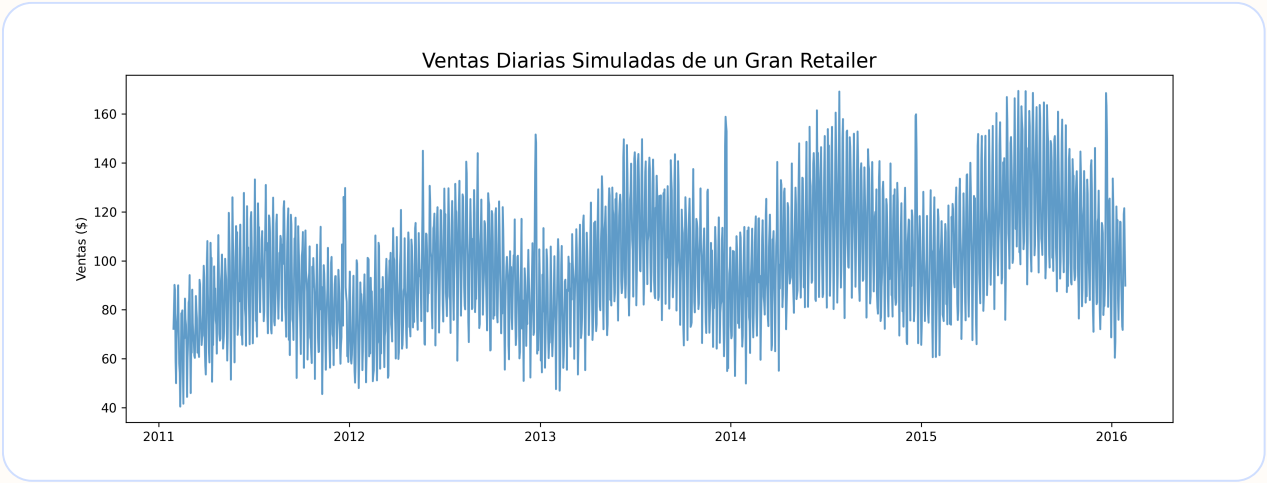


Figura 4.1: Ventas diarias simuladas de un gran retailer. Note la estacionalidad semanal (dientes de sierra), anual (ondas) y los picos de Navidad.

4.3 Feature engineering: enseñando tiempo a la máquina

Si le damos solo la fecha a XGBoost, no aprenderá nada útil. Necesitamos crear columnas ("features") que expliquen el contexto temporal.

4.3.1 1. Features de calendario

Extraemos información explícita de la fecha:

- **Día de la semana:** ¿Es lunes o sábado? (Crucial para retail).
- **Mes:** ¿Es diciembre? (Navidad).
- **Fin de semana:** ¿Es sábado o domingo?

4.3.2 2. Lags

Creamos columnas con los valores pasados:

- `lag_1` : ¿Cuánto vendí ayer?
- `lag_7` : ¿Cuánto vendí el mismo día de la semana pasada?
- `lag_28` : ¿Cuánto vendí hace 4 semanas? (captura ciclos mensuales).
- `lag_365` : ¿Cuánto vendí este día el año pasado?

4.3.3 3. Ventanas móviles

Capturan la tendencia reciente:

- `rolling_mean_7` : Promedio de ventas de la última semana.
- `rolling_std_7` : Variabilidad de ventas de la última semana.

4.3.4 El dataset resultante

Después de aplicar este proceso, cada fila de nuestro dataset ya no es solo una fecha y una venta. Es una "foto" completa del contexto temporal:

Ventas	dow	mes	lag_1	lag_7	lag_365	roll_mean_7	weekend
132.5	3	6	118.2	130.1	125.8	127.4	0
155.8	4	6	132.5	148.3	150.2	129.1	0
178.3	5	6	155.8	170.6	172.1	133.5	1
165.1	6	6	178.3	160.2	158.7	138.2	1
120.4	0	6	165.1	122.8	118.5	141.7	0

Tabla 4.1: Ejemplo del dataset con features temporales. Cada fila es una "foto" que XGBoost puede entender: sabe qué día de la semana es, cuánto se vendió ayer, hace una semana, y hace un año.

Ahora XGBoost puede aprender reglas como: *"Si es sábado (dow=5) y las ventas de ayer fueron altas (lag_1 > 150), entonces las ventas de hoy serán altas"*.

4.4 La familia boosting: no solo XGBoost

A lo largo de este capítulo usamos XGBoost, pero es un miembro de una familia más amplia. Las tres librerías principales de gradient boosting para series de tiempo son:

Un detalle notable: CatBoost usa **ordered boosting**, que respeta el orden temporal de los datos durante el entrenamiento. Esto lo hace naturalmente adecuado para series de tiempo, ya que evita una forma sutil de data leakage presente en el boosting estándar cuando se aplica a datos secuenciales.

En la práctica, empezá con LightGBM (ciclo de iteración más rápido) y cambiá a CatBoost si tenés muchas features categóricas que se beneficien del encoding nativo.

4.5 Feature engineering avanzado

Las features básicas (calendario, lags, ventanas móviles) de la sección anterior cubren la mayoría de los casos. Pero para problemas complejos, tres técnicas adicionales pueden hacer una diferencia significativa:

Modelo	Fortaleza clave	Mejor caso de uso
XGBoost	Robusto, bien documentado, gran comunidad	Default de propósito general
LightGBM	Entrenamiento más rápido, crecimiento por hojas, bajo consumo de memoria	Datasets grandes, modelos globales (Cap 7). Fue el modelo base de las soluciones ganadoras en la competición M5 de Kaggle
CatBoost	Manejo nativo de categóricas, ordered boosting	Datasets con muchas features categóricas (producto, tienda, región)

Tabla 4.2: Comparación de la familia gradient boosting. Los tres aceptan el mismo feature engineering; la diferencia está en la velocidad de entrenamiento y cómo manejan tipos de datos específicos.

Features de Fourier capturan estacionalidad compleja. Cuando una serie tiene múltiples ciclos superpuestos (diario + semanal + anual), codificarlos como pares seno/coseno a diferentes frecuencias permite al modelo aprender cada ciclo de forma independiente:

$$\sin\left(\frac{2\pi k \cdot t}{T}\right), \cos\left(\frac{2\pi k \cdot t}{T}\right) \quad \text{para } k = 1, 2, \ldots$$

Esto es particularmente útil para datos horarios con patrones diarios y semanales, donde las features de calendario solas son demasiado burdas.

Target encoding para variables categóricas reemplaza una categoría (ej: "Tienda A") con la media histórica del target para esa categoría. Esto le da al modelo un "baseline" numérico para cada grupo. **Precaución:** siempre calculá el encoding usando solo datos pasados para evitar leakage.

Features de interacción combinan dos features en una: `Lunes_Electrónica` captura el hecho de que los patrones de venta del lunes difieren entre categorías de producto. Estas interacciones son cosas que los modelos de árboles pueden descubrir solos, pero proporcionarlas explícitamente acelera el aprendizaje y mejora la precisión en datasets más chicos.

Un insight reciente de la literatura: **el feature engineering importa más que la arquitectura del modelo**. Un LightGBM bien engineerado supera consistentemente a un modelo de deep learning con features pobres, incluso para demanda intermitente.

4.6 Estrategias de pronóstico multi-step

Cuando necesitás pronosticar $h > 1$ pasos adelante, enfrentás una decisión de diseño:

- **Recursivo:** Entrenás un modelo para h = 1. Para predecir h = 2, alimentás la predicción de h = 1 como lag. Simple, pero los errores se acumulan con cada paso.
- **Directo:** Entrenás h modelos separados, uno por horizonte. Sin acumulación de errores, pero más modelos que mantener y sin compartir información entre horizontes.
- **Híbrido (Rectify):** Combiná ambos: usá las predicciones recursivas como features para un modelo directo. Investigación reciente muestra que las estrategias híbridas superan a cualquiera de las dos solas en la mayoría de los casos.

Librerías como `skforecast` soportan las tres estrategias out of the box, facilitando experimentar y elegir la mejor para tus datos.

4.7 El duelo: XGBoost vs ARIMA

4.7.1 ¿Por qué comparar?

En los Capítulos 2 y 3, invertimos mucho esfuerzo en transformar la serie (logaritmos, diferenciación) y configurar SARIMA. ¿Valió la pena toda esa teoría, o un algoritmo de ML "bruto" con buenas features puede hacer lo mismo (o mejor)?

Esta pregunta es fundamental en la práctica: en una empresa, el tiempo del Data Scientist es limitado. Si XGBoost con features manuales logra resultados comparables a SARIMA sin necesidad de entender estacionariedad, muchos equipos preferirán la ruta de ML.

Entrenamos dos modelos sobre los mismos datos:

1. **XGBoost:** Con todas las features de la Tabla 4.1.
2. **SARIMA:** Configurado para capturar el ciclo semanal (7 días), extendiendo la lógica estacional del Capítulo 3 a datos diarios.

4.7.2 ¿Cómo medimos el éxito?

Para declarar un ganador, necesitamos un árbitro imparcial. Usaremos el **MAE (Mean Absolute Error)**:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Es simplemente el promedio de nuestros errores en valor absoluto. Si vendo 100 y predije 110, mi error es 10. El MAE nos dice, en promedio, cuántas unidades nos equivocamos cada día. Es fácil de interpretar para cualquier gerente.

4.7.3 Resultados visuales

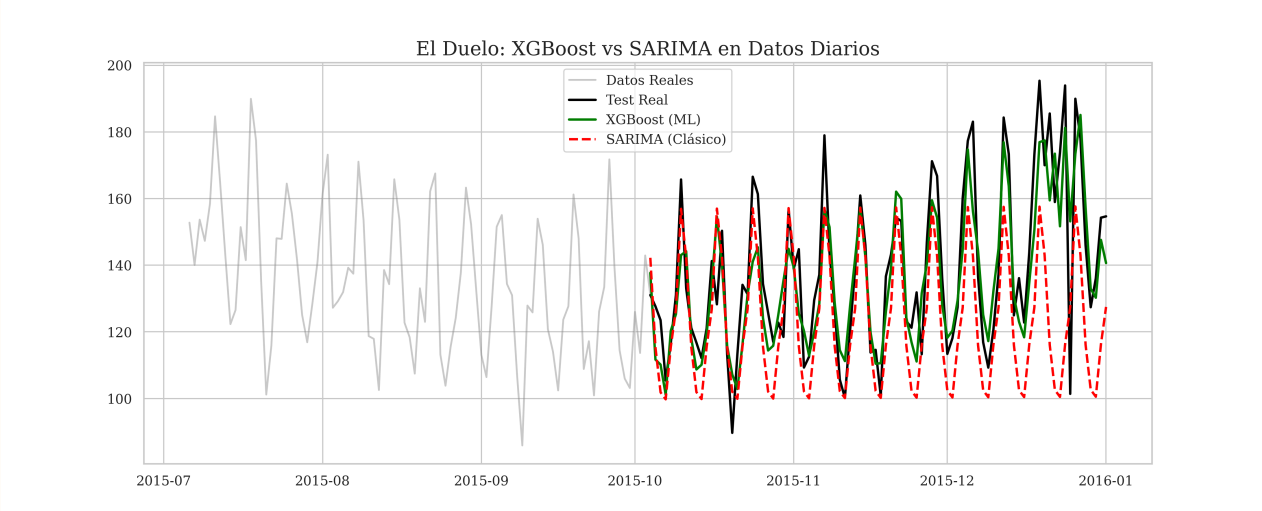


Figura 4.2: Comparación de pronósticos en los últimos 3 meses. La línea negra es la realidad, la verde es XGBoost y la roja es SARIMA. Preste especial atención a la zona de finales de Diciembre (Navidad).

4.7.4 Análisis de resultados

Observemos la Figura 4.2 con detalle:

- **XGBoost (verde):** Se ajusta muy bien a los datos reales a lo largo de todo el periodo de test. Captura el pico de compras pre-navideñas (21-24 de Diciembre) y la caída del día 25 (tienda cerrada). ¿Por qué? Principalmente gracias a `lag_365`, que le permite "recordar" el patrón navideño del año anterior, combinado con features de calendario como `month` y `day_of_year` que aportan contexto adicional.
- **SARIMA (rojo):** Captura correctamente el ritmo semanal (sube y baja cada 7 días), pero **falla completamente en Navidad**. Sigue repitiendo el patrón promedio semanal sin entender que Diciembre es especial. El modelo clásico es "ciego" a eventos especiales a menos que usemos técnicas avanzadas como Regresión Dinámica.

Métricas de error (MAE):

- XGBoost: **10.29** (en promedio, se equivoca por $10.29 al día)
- SARIMA: **18.19** (casi el doble de error)

XGBoost reduce el error a la mitad. La diferencia no es marginal: en un contexto de retail con miles de productos, esto se traduce en millones de dólares menos en inventario desperdiciado.

4.7.5 ¿Qué features fueron más importantes?

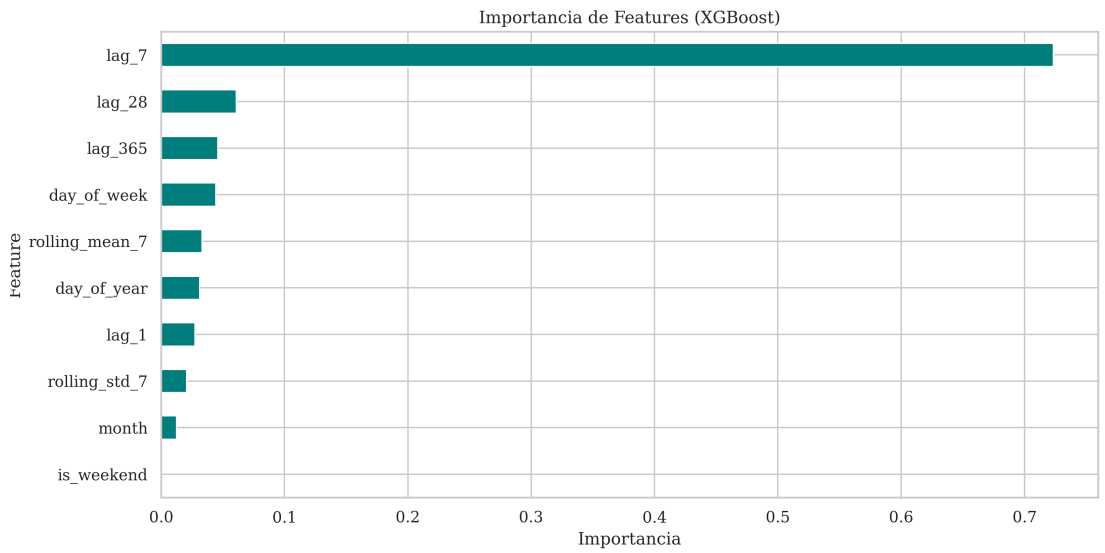


Figura 4.3: Importancia de cada feature para XGBoost. El lag semanal (lag_7) domina abrumadoramente, seguido de lag_28 y lag_365. El ciclo semanal es la señal más fuerte en estos datos de retail.

4.7.6 Más allá del feature importance: SHAP

El feature importance incorporado (Figura 4.3) nos dice qué features importan globalmente. Pero **SHAP (SHapley Additive exPlanations)** va más allá: explica cómo cada feature contribuyó a una predicción específica. Por ejemplo, SHAP puede mostrar que para el 24 de Diciembre, `lag_365` empujó la predicción 40 unidades hacia arriba porque la Nochebuena del año pasado también fue alta. Este nivel de explicabilidad es esencial cuando un stakeholder pregunta "¿por qué el modelo predijo esto?"

4.8 Hiperparámetros esenciales

Tres hiperparámetros dominan el rendimiento de cualquier modelo de boosting:

- `learning_rate` (0.01–0.3): Cuánto corrige cada árbol al anterior. Valores más bajos aprenden más lento pero generalizan mejor.
- `n_estimators` (100–3000): Número de árboles. Más árboles con un learning rate bajo es generalmente mejor, pero aumenta el tiempo de entrenamiento.
- `max_depth` (3–8): Profundidad de cada árbol. Árboles más profundos capturan interacciones más complejas pero arriesgan sobreajuste.

Regla general: Empezá con `learning_rate=0.05`, `max_depth=5` y `n_estimators=1000` con early stopping. El mecanismo de early stopping monitorea el error de validación y deja de agregar árboles cuando el rendimiento se estanca, previniendo el sobreajuste automáticamente. Para tunear hiperparámetros de forma rigurosa, ver las técnicas de validación del Capítulo 5.

4.9 Guía de selección: ¿qué modelo debo usar?

Esta es la pregunta del millón. Aquí tenés una guía práctica para decidir.

Característica	Usar ARIMA / ETS	Usar ML (XGBoost)
Frecuencia	Baja (mensual, trimestral)	Alta (diaria, horaria)
Estacionalidad	Simple (solo anual)	Compleja (múltiples ciclos)
Variables externas	Difícil de incluir	Muy fácil (precio, clima)
Cantidad de series	Pocas (< 100)	Muchas (> 1000)
Horizonte	Corto plazo	Corto y mediano plazo
Explicabilidad	Alta (coeficientes)	Media (feature importance)

Tabla 4.3: Guía para elegir entre Estadística Clásica y Machine Learning.

4.10 Conclusión

Hemos demostrado que para datos de alta frecuencia (diarios) con múltiples patrones (semanal + anual) y eventos especiales (Navidad), Machine Learning supera a los métodos clásicos.

Pero la clave no es el algoritmo en sí, sino el **Feature Engineering**: la capacidad de traducir nuestro conocimiento del negocio (calendarios, lags, ventanas móviles) en columnas que el modelo pueda entender. Sin buenas features, XGBoost será tan ciego como ARIMA ante la Navidad.

En el próximo capítulo, aprenderemos a validar correctamente estos modelos con **Backtesting**, para asegurarnos de que no nos estamos engañando.

CAPÍTULO 5

Validación y métricas de error

5.1 Introducción: el peligro del “Lucky Split”

Imaginá que sos estudiante y el profesor te dice: “El examen final será sobre el tema 4”. Estudiás solo el tema 4 y sacás un 10. ¿Sabés de la materia? Probablemente no. Tuviste suerte.

En series de tiempo, hacer una sola división Train/Test (como hicimos en los capítulos anteriores) es arriesgado.

- Si tu Test cae en un mes tranquilo (ej: julio), tu modelo parecerá un genio.
- Si tu Test cae en una crisis (ej: pandemia o Navidad), tu modelo parecerá un inútil.

En nuestro experimento, vimos que el error (MAE) cambiaba drásticamente:

- Evaluando en julio: MAE = **8.89**
- Evaluando en diciembre: MAE = **18.42**

Más del doble de error solo por cambiar la fecha de evaluación.

5.2 La solución: backtesting

Para evitar engañarnos a nosotros mismos, debemos evaluar el modelo múltiples veces en diferentes momentos del pasado. Usamos una técnica de **Ventana Expansiva** :

1. Entrenamos con datos hasta marzo → Predecimos abril.
2. Entrenamos con datos hasta abril → Predecimos mayo.
3. Entrenamos con datos hasta mayo → Predecimos junio.

Al final, promediamos los errores de los 5 folds. Este promedio es una estimación mucho más honesta de cómo se comportará el modelo en el futuro real.

5.3 Métricas de error: ¿cómo medimos el éxito?

Ya tenemos las predicciones, pero ¿cómo medimos qué tan “cerca” estuvimos? Existen varias métricas, cada una con sus ventajas y limitaciones. Veámoslas con un ejemplo concreto.

Supongamos que en 5 días las ventas reales fueron [100, 150, 80, 0, 200] y nuestro modelo predijo [110, 140, 95, 10, 180].

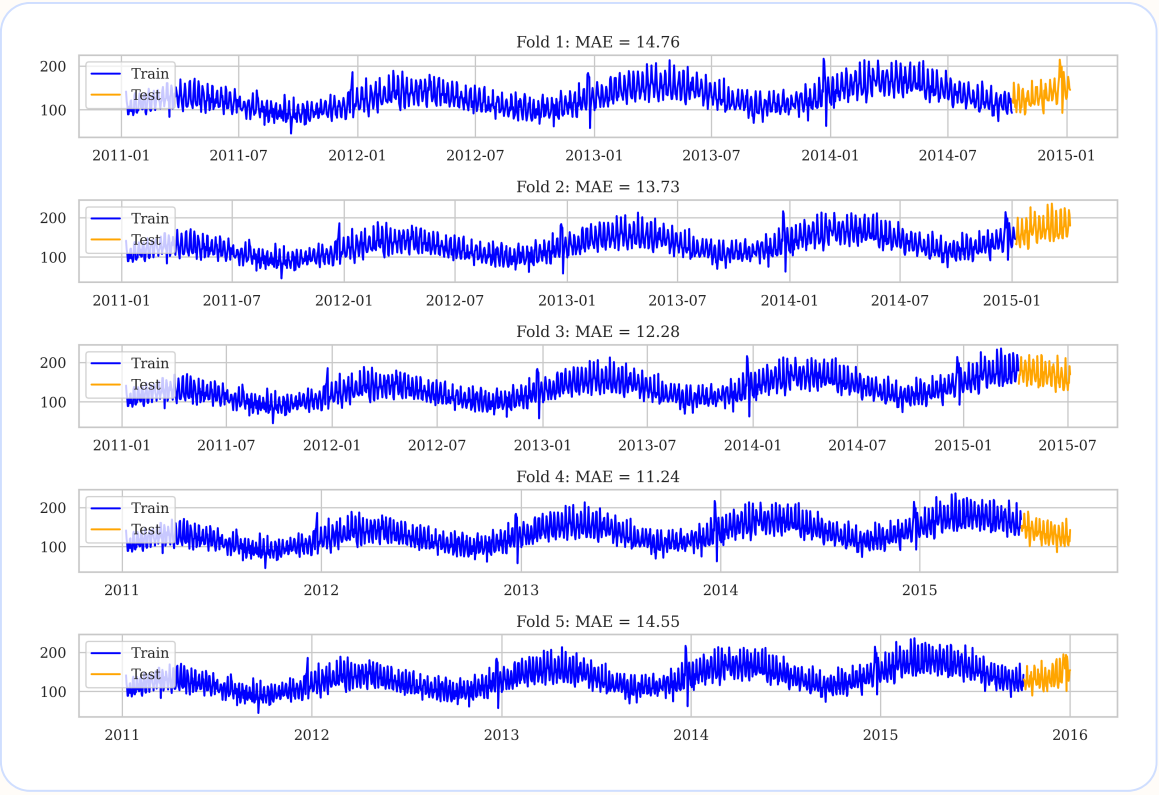


Figura 5.1: Backtesting con 5 folds. Cada panel es una evaluación independiente. El modelo debe obtener un buen resultado en todos, no solo en uno.

5.3.1 1. MAE (mean absolute error)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Calculemos paso a paso con nuestro ejemplo:

$$\text{MAE} = \frac{|100-110| + |150-140| + |80-95| + |0-10| + |200-180|}{5} = \frac{10+10+15+10+20}{5} = \frac{65}{5} = 13$$

Interpretación: "En promedio, nos equivocamos por $13 cada día". Es la métrica más intuitiva: cualquier gerente la entiende sin necesidad de explicar fórmulas. Es robusta porque no penaliza desproporcionadamente los errores grandes.

5.3.2 2. RMSE (root mean squared error)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Con el mismo ejemplo:

$$\text{RMSE} = \sqrt{\frac{10^2+10^2+15^2+10^2+20^2}{5}} = \sqrt{\frac{100+100+225+100+400}{5}} = \sqrt{185} = 13.6$$

Interpretación: Note que el RMSE (13.6) es mayor que el MAE (13.0). Siempre lo será, porque el RMSE penaliza los errores grandes al cuadrado. El error de 20 del día 5 "pesa" más en el RMSE que en el MAE. Usá RMSE cuando un error grande es catastrófico (ej: si predecir de menos significa parar una línea de producción).

5.3.3 3. MAPE (mean absolute percentage error)

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

Con nuestro ejemplo:

$$\text{MAPE} = \frac{\frac{10}{100} + \frac{10}{150} + \frac{15}{80} + \frac{10}{0} + \frac{20}{200}}{5} \times 100$$

Problema: El día 4 la venta real fue 0. La división $\frac{10}{0} = \infty$. El MAPE explota a infinito. Esto ocurre constantemente en retail con productos de baja rotación (como veremos en el Capítulo 10). Por esta razón, **nunca uses MAPE si tu serie tiene ceros**.

5.3.4 4. WMAPE (weighted MAPE)

$$\text{WMAPE} = \frac{\sum |y_i - \hat{y}_i|}{\sum y_i} \times 100$$

Con nuestro ejemplo (excluyendo el problema del cero):

$$\text{WMAPE} = \frac{10 + 10 + 15 + 10 + 20}{100 + 150 + 80 + 0 + 200} \times 100 = \frac{65}{530} \times 100 = 12.3\%$$

Interpretación: "Nuestro error total equivale al 12.3% de las ventas totales". La ventaja del WMAPE es que pondera por volumen: un error de 10 en un producto que vende 200 "pesa" menos que un error de 10 en un producto que vende 20. Es la favorita del sector Retail.

5.4 Diagnóstico de errores: de los residuos a las features

No solo importa cuánto nos equivocamos, sino cuándo y por qué. Los residuos (errores) del modelo cuentan una historia.

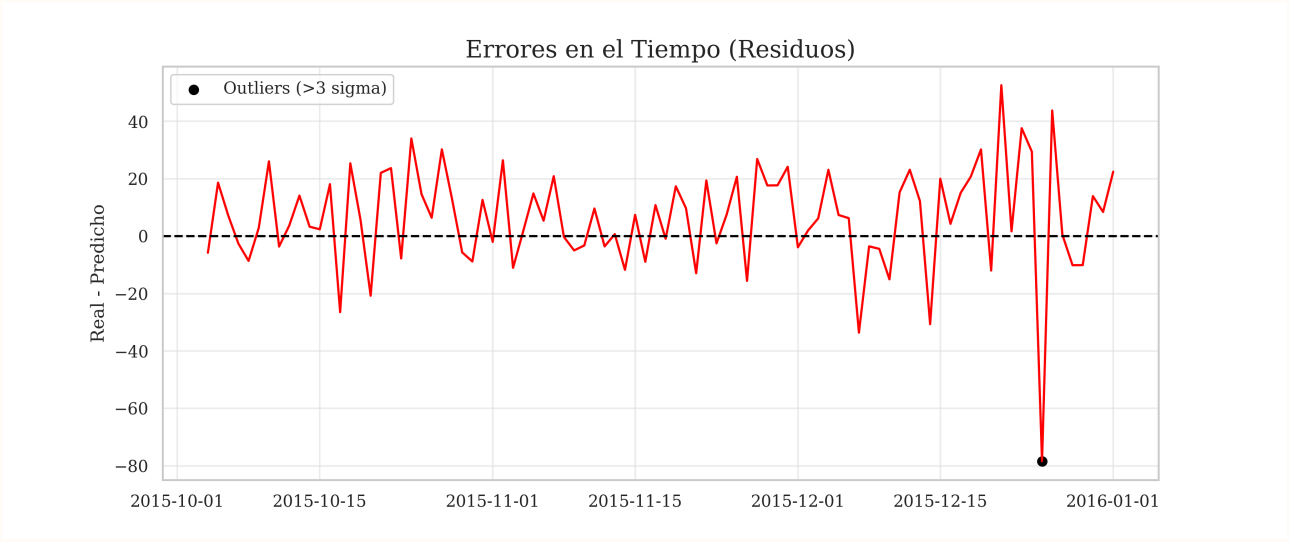


Figura 5.2: Residuos en el tiempo. La línea roja oscila mayormente alrededor de cero, pero note el sesgo positivo creciente hacia diciembre: el modelo sub-predice sistemáticamente la temporada navideña. El punto negro es un "Outlier": un error inusualmente grande que supera 3σ.

5.4.1 ¿Cómo leer este gráfico?

- **Si los residuos oscilan alrededor de cero sin patrones:** El modelo está capturando bien la señal. Los errores son "ruido" aleatorio.
- **Si hay patrones visibles** (ej: errores siempre positivos los viernes): El modelo está fallando sistemáticamente en algo. Hay información que no capturó.
- **Si hay puntos negros (outliers):** Son días donde el error fue anormalmente grande. Típicamente corresponden a eventos que el modelo no conoce.

5.4.2 De residuos a features: capitalizando los errores

Los outliers no son solo "ruido molesto". Son una oportunidad para mejorar el modelo.

El proceso es iterativo:

1. **Identificar los outliers:** ¿Qué días tuvimos errores enormes?
2. **Investigar la causa:** ¿Hubo una promoción? ¿Un feriado? ¿Una huelga de transporte? ¿Mal clima?
3. **Crear una feature:** Si descubrimos que los errores grandes coinciden con promociones, creamos una columna `is_promotion` (0 o 1) y la agregamos al modelo del Capítulo 4.
4. **Re-entrenar y re-evaluar:** El modelo ahora "sabe" que existen las promociones. Los outliers deberían reducirse.

Este ciclo de error → investigación → feature → mejora es el corazón del trabajo diario de un Data Scientist en forecasting. Un modelo nunca está "terminado"; siempre se puede mejorar incorporando más conocimiento del negocio.

5.5 Conclusión

Ahora tenemos un modelo validado robustamente (Backtesting) y sabemos exactamente cuánto confiar en él (Métricas).

Resultados finales de nuestro Backtesting:

- MAE promedio: **13.31**
- MAPE promedio: **8.87%**

Estamos listos para desplegar este modelo. Pero una predicción puntual sola puede ser peligrosamente engañosa. En el próximo capítulo, aprenderemos a cuantificar la incertidumbre de nuestras predicciones con **Intervalos de Confianza, Forecast Bayesiano y Conformal Prediction**.

CAPÍTULO 6

Navegando la incertidumbre

6.1 Introducción: el pronóstico del clima

Cuando un meteorólogo sale en televisión, no dice “Mañana hará 25 grados”. Dice: “Mañana hará entre 23 y 27 grados, con un 80% de probabilidad”.

¿Por qué? Porque sabe que predecir el futuro exacto es imposible. Lo honesto es comunicar cuánta incertidumbre tiene su predicción.

En los capítulos anteriores, hemos construido modelos que nos dan un número puntual: “Las ventas de enero serán 450”. Pero en la vida real, un gerente de operaciones necesita saber: ¿Podrían ser 400? ¿Podrían ser 550? ¿Cuánto stock debo pedir para no quedarme sin producto ni sobrecomprar?

En este capítulo, exploraremos tres formas de cuantificar esa incertidumbre.

6.2 Intervalos de confianza clásicos

6.2.1 ¿Qué es un intervalo de confianza?

Un intervalo de confianza (IC) es un rango alrededor de la predicción puntual que indica dónde esperamos que caiga el valor real con cierta probabilidad. Por ejemplo:

- **IC al 80%:** Esperamos que el valor real caiga dentro de este rango el 80% de las veces.
- **IC al 95%:** Un rango más amplio, con mayor “seguridad”.

La relación entre confianza y ancho es directa: cuanta más seguridad quieras, más ancho será el intervalo. Un IC al 99% es tan amplio que casi siempre acierta, pero es poco útil en la práctica (“las ventas serán entre 0 y 1 millón”).

6.2.2 ¿Cómo se calculan?

Para un modelo ARIMA, el intervalo de confianza al 100(1 − α)% se calcula como:

$$\hat{Y}_{t+h} \pm z_{\alpha/2} \cdot \sigma_h$$

donde:

- $\hat{Y}_{t+h}$ es la predicción puntual a horizonte h.
- $z_{\alpha/2}$ es el cuantil de la distribución Normal estándar (1.96 para 95%, 1.28 para 80%).
- σ_h es la desviación estándar del error de predicción a horizonte h.

Ejemplo numérico: Si nuestro modelo predice 450 pasajeros para el próximo mes ($\hat{Y} = 450$), con una desviación estándar del error de $\sigma = 30$, el IC al 95% sería:

$$450 \pm 1.96 \times 30 = 450 \pm 58.8 = [391.2, 508.8]$$

Es decir, esperamos entre 391 y 509 pasajeros con un 95% de confianza.

6.2.3 Los supuestos

Esta fórmula elegante tiene condiciones que deben cumplirse:

1. **Normalidad:** Los errores del modelo deben seguir una distribución Normal (campana de Gauss). Si los errores son asimétricos o tienen colas pesadas, los intervalos serán incorrectos.

2. **Homocedasticidad:** La varianza de los errores debe ser constante en el tiempo. Si los errores son más grandes en ciertos periodos (como vimos en el Capítulo 1 con la heterocedasticidad), los intervalos no se adaptan.

3. **Independencia:** Los errores no deben estar correlacionados entre sí. Si el error de hoy predice el de mañana, los intervalos subestiman la incertidumbre.

Cada modelo calcula sus intervalos de forma diferente. ARIMA asume errores normales. XGBoost no genera intervalos nativamente (necesita técnicas adicionales como Conformal Prediction, que veremos más adelante). Los modelos bayesianos generan distribuciones completas sin asumir normalidad.

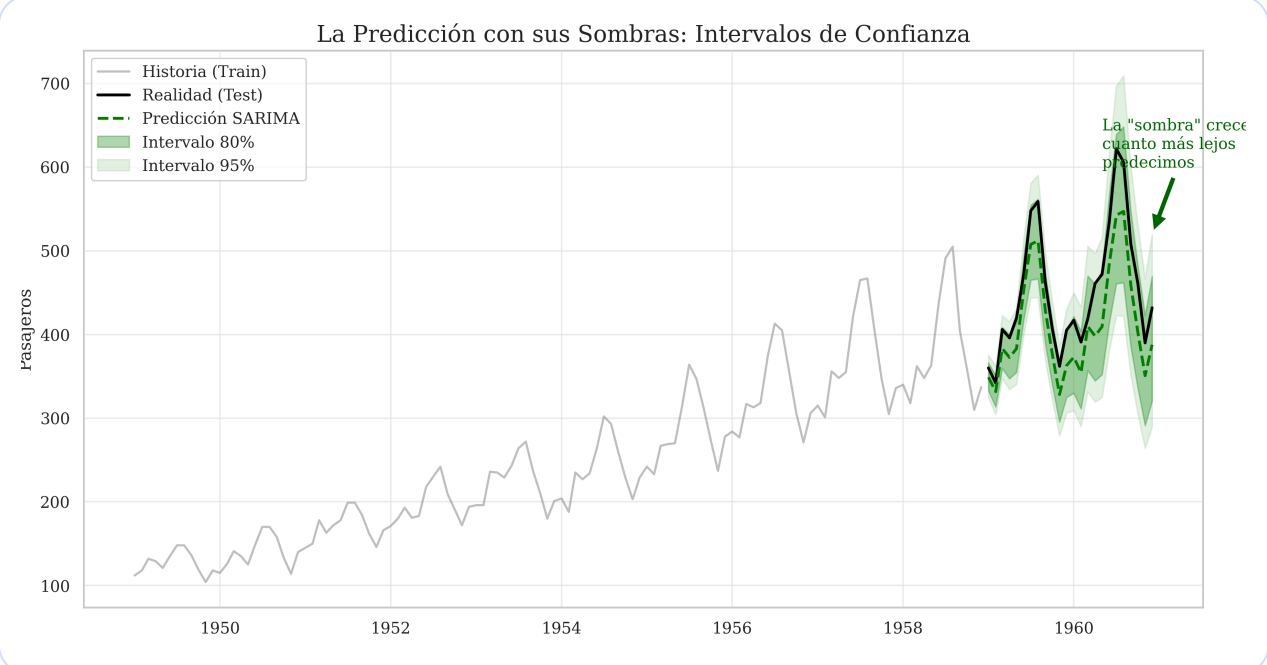


Figura 6.1: Predicción SARIMA con intervalos de confianza al 80% y 95%. Note cómo las bandas se ensanchan cuanto más lejos predecimos: estamos más seguros sobre el mes que viene que sobre el próximo año.

6.2.4 La interpretación en escala logarítmica

Esto tiene una consecuencia importante: los intervalos se vuelven **asimétricos**. Si en escala log el intervalo es [5.8, 6.2], al transformar de vuelta obtenemos $[e^{5.8}, e^{6.2}] = [330, 493]$. El límite superior está más lejos de la predicción que el inferior. Esto es coherente con la realidad: una aerolínea puede crecer mucho en un buen año, pero difícilmente caerá a cero.

En el Capítulo 2, transformamos los datos con logaritmos. Esto significa que el modelo SARIMA trabaja internamente en escala logarítmica. Para obtener los intervalos en pasajeros reales, aplicamos la función inversa (e^x) a los límites del intervalo.

6.2.5 El contrafactual: intervalos sobre datos crudos vs transformados

¿Qué pasa si ignoramos los supuestos? Comparemos los intervalos de un SARIMA sobre datos sin transformar vs datos log-transformados:

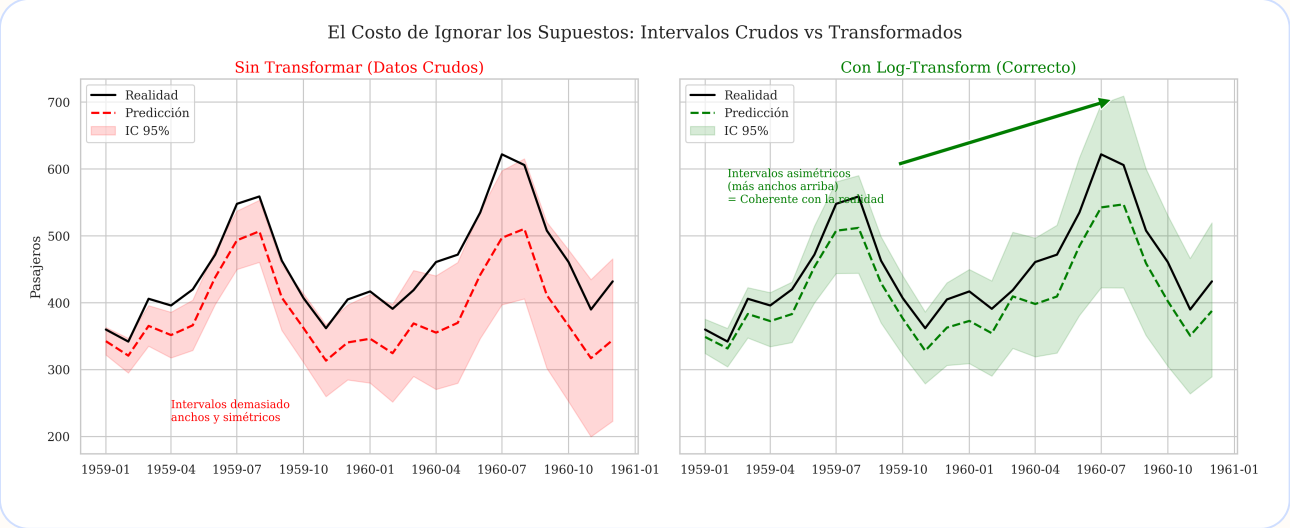


Figura 6.2: Izquierda: Sin transformar. Los intervalos son simétricos y excesivamente anchos, sin capturar el patrón real de los datos. **Derecha:** Con log-transform. Los intervalos son asimétricos y coherentes con la realidad.

Los resultados son reveladores:

- **Datos crudos:** El IC al 95% solo contiene el 54% de los valores reales. Los intervalos son simétricos y mal calibrados: la varianza no constante hace que sean demasiado anchos en algunos tramos y demasiado estrechos en otros.
- **Log-transform:** El IC al 95% contiene el 100% de los valores reales. Al transformar, los supuestos se cumplen mejor y los intervalos son fiables.

Lección: Los intervalos clásicos son tan buenos como los supuestos que los sostienen. Sin la transformación logarítmica del Capítulo 2, los intervalos dan información incorrecta.

6.3 Forecast Bayesiano: la distribución completa

6.3.1 ¿Qué es el enfoque Bayesiano?

Imaginá que querés predecir las ventas del próximo mes.

- **El enfoque clásico:** "Las ventas serán 450" (un solo número, con un intervalo alrededor).
- **El enfoque Bayesiano:** "Hay un 60% de probabilidad de que estén entre 420-480, un 25% de que estén entre 380-420, y un 15% de que superen 480" (una distribución completa de probabilidad).

El enfoque bayesiano no genera un solo pronóstico, sino **cientos de escenarios posibles** del futuro. Cada escenario representa un "mundo posible" donde los parámetros del modelo son ligeramente diferentes, reflejando nuestra incertidumbre sobre ellos.

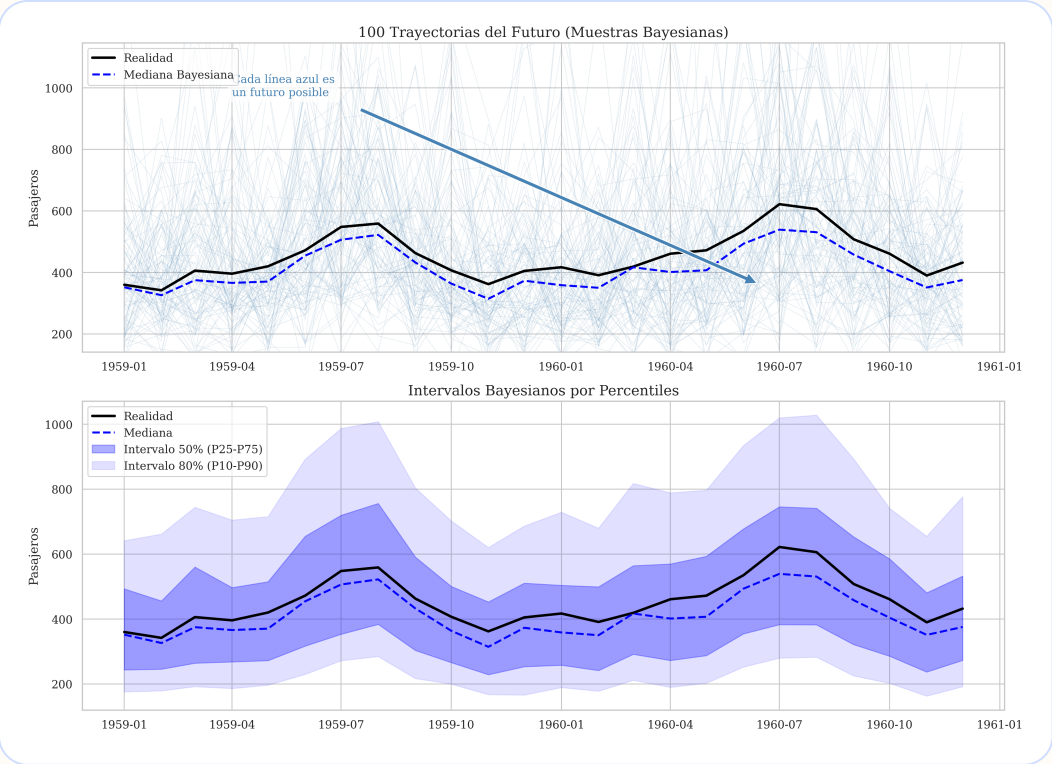


Figura 6.3: Arriba: 100 trayectorias posibles del futuro (líneas azules). La dispersión de estas líneas visualiza la incertidumbre. **Abajo:** Los intervalos por percentiles derivados de las 500 muestras totales.

6.3.2 Resultados

En la Figura 6.3, el panel superior muestra 100 trayectorias superpuestas. Note cómo forman un "abanico" que se ensancha hacia el futuro: hay más consenso sobre los primeros meses que sobre los últimos. El panel inferior resume todas las muestras en intervalos por percentiles (P25-P75 para el 50%, P10-P90 para el 80%).

6.3.3 Ventajas del enfoque Bayesiano

- **Distribución completa:** No solo tenés la media y un intervalo; tenés toda la forma de la incertidumbre. Esto es valioso cuando las decisiones dependen de escenarios extremos (ej: "¿Cuál es la probabilidad de que las ventas caigan por debajo de 300?").
- **Incorpora conocimiento previo (Priors):** Podés decirle al modelo "Las ventas nunca bajan de 100" y él lo tiene en cuenta. Esto es útil cuando tenés poca data pero mucho conocimiento del dominio.
- **Intervalos más flexibles:** No asume normalidad obligatoriamente. La forma de los intervalos se adapta a la distribución real de los errores.

6.3.4 Limitaciones

- **Costo computacional:** Generar cientos o miles de muestras es más lento que calcular un intervalo analítico. Para series de alta frecuencia (minutales) con miles de productos, puede ser prohibitivo.
  Elección de Priors: Si tus "creencias iniciales" son incorrectas (ej: asumís que las ventas nunca superan 500 y resulta que sí), el modelo puede dar resultados sesgados. En la práctica, se suelen usar priors "débiles" (poco informativos) para dejar que los datos hablen.
  Cuándo NO usarlo: Si tenés una sola serie mensual con muchos datos históricos y los supuestos de normalidad se cumplen, un SARIMA clásico con intervalos analíticos será más rápido y igual de preciso.

6.4 Conformal Prediction: el enfoque moderno

¿Y si pudiéramos generar intervalos de predicción que funcionen con **cualquier modelo** (XGBoost, Redes Neuronales, lo que sea) sin asumir absolutamente nada sobre la distribución de los errores?

Eso es exactamente lo que hace **Conformal Prediction** .

6.4.1 La idea en 3 pasos

1. Dividir datos en 3: Train (para entrenar el modelo), Calibración (para medir errores) y Test (para evaluar).

2. Calcular errores absolutos en Calibración: Predecimos en el set de Calibración y medimos $|y_i - \hat{y}_i|$ para cada punto.

3. Tomar el cuantil: Si queremos cobertura del 90%, tomamos el percentil 90 de esos errores absolutos. Ese número es el "ancho" q de la banda.

La predicción final para un nuevo punto es:

$$[\hat{Y} - q, \hat{Y} + q]$$

donde $\hat{Y}$ es la predicción puntual del modelo y q es el cuantil calculado en el paso 3.

6.4.2 Ejemplo numérico

Supongamos que nuestro XGBoost predijo 10 puntos en el set de Calibración, y los errores absolutos fueron:

$$|e| = [5, 8, 12, 3, 7, 15, 6, 9, 4, 11]$$

Ordenamos de menor a mayor: [3, 4, 5, 6, 7, 8, 9, 11, 12, 15]. Para cobertura del 90%, tomamos el percentil 90, que es el valor en la posición $\lceil 0.90 \times 10 \rceil = 9$: **q = 12**.

Ahora, si el modelo predice $\hat{Y} = 150$ para un nuevo punto, el intervalo conformal al 90% es:

$$[150 - 12, 150 + 12] = [138, 162]$$

No asumimos nada sobre la distribución de los errores. Simplemente usamos los errores reales observados.

6.4.3 El supuesto de intercambiabilidad

El único supuesto de Conformal Prediction es que los datos de calibración y los datos futuros provienen del mismo proceso. En la práctica, esto significa:

- **Funciona bien cuando:** Las condiciones del negocio durante la calibración son similares a las del futuro. Por ejemplo, si calibrás con datos de 2023 y predecís 2024, y no hay cambios estructurales (nuevas regulaciones, pandemias, etc.), los errores de calibración son representativos.
- **Falla cuando:** Hay un cambio estructural entre la calibración y el futuro. Por ejemplo, si calibrás con datos pre-pandemia (2019) y predecís durante la pandemia (2020), los errores de calibración subestimarán dramáticamente los errores reales, y los intervalos serán demasiado estrechos.

En resumen: Conformal funciona si "el futuro se parece al pasado reciente". Si hay un quiebre, ningún método de incertidumbre (ni clásico, ni bayesiano) será fiable.

6.4.4 Resultados: Conformal vs paramétrico

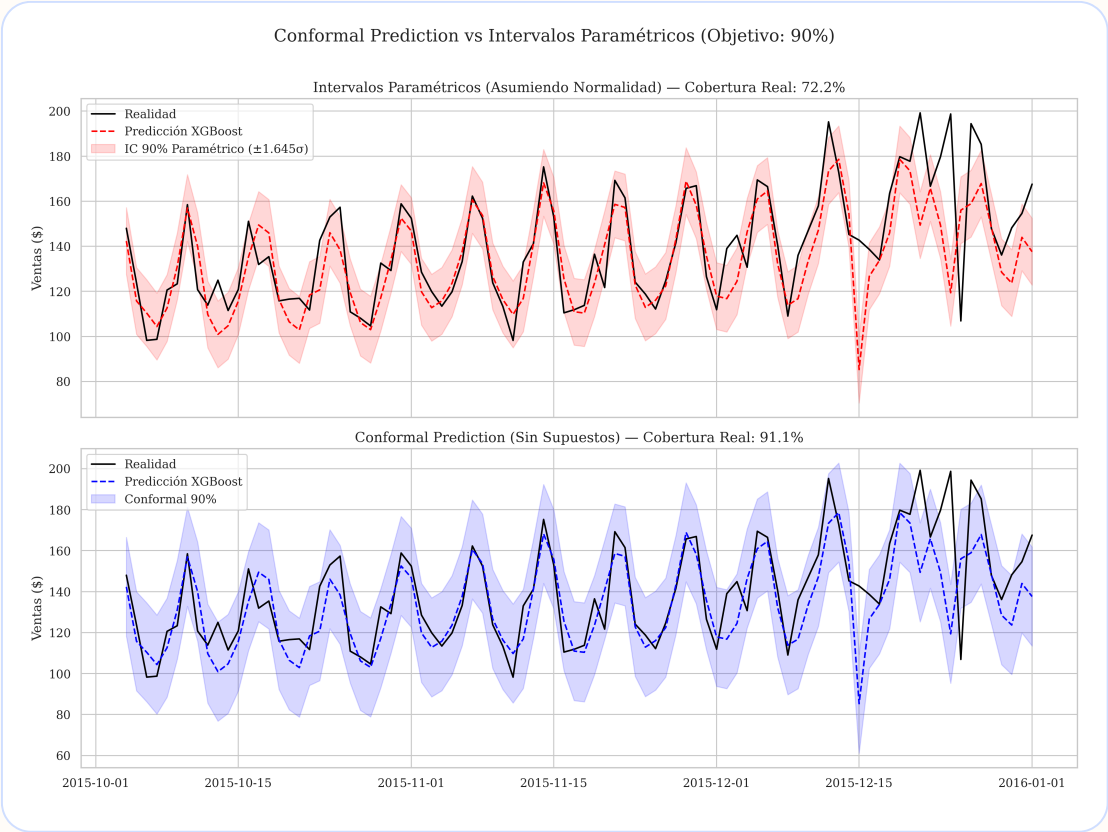
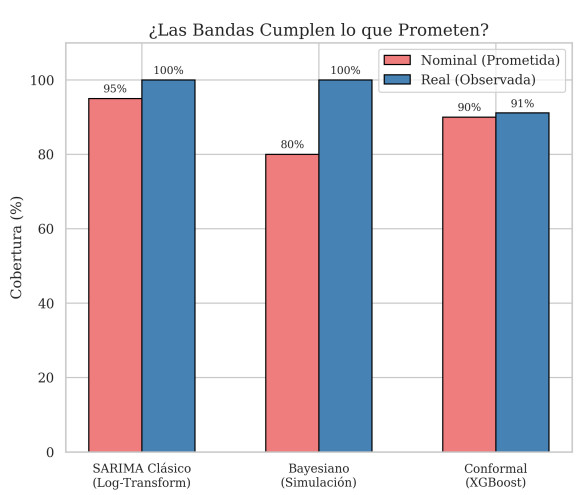


Figura 6.4: Arriba: Intervalos paramétricos (asumiendo normalidad) logran solo 72% de cobertura cuando el objetivo era 90%. **Abajo:** Conformal Prediction logra 91%, sin asumir distribución alguna.

Los intervalos paramétricos fallan porque los errores de XGBoost no son normales. Conformal Prediction alcanza el objetivo porque no hace suposiciones: mide los errores reales y usa su distribución empírica.

6.5 Comparación y guía de selección



Guía Rápida: ¿Cuándo Usar Cada Enfoque?

Método	Supuestos	Complejidad	Mejor Para
SARIMA Clásico	Sí (Normalidad)	Baja	Series univariadas simples
Bayesiano	Parcial (Priors)	Alta	Cuando necesitas la distribución completa
Conformal	No	Baja	Cualquier modelo (ML, DL, etc.)

Figura 6.5: Cobertura real vs nominal de los tres métodos. Los Intervalos Clásicos (log-transform) alcanzan cobertura perfecta; el enfoque Bayesiano supera ampliamente el objetivo (intervalos conservadores); Conformal Prediction cumple la garantía teórica sin supuestos distribucionales.

Método	Supuestos	Complejidad	Modelos compatibles	Tipo de problema
Intervalos clásicos	Normalidad, homocedasticidad	Baja	ARIMA, ETS	Series univariadas mensuales/trimestrales con comportamiento estable.
Bayesiano	Priors (creencias iniciales)	Alta	Prophet, modelos estructurales	Cuando necesitás la distribución completa o tenés mucho conocimiento del dominio y pocos datos.
Conformal	Solo intercambiabilidad	Baja	Cualquiera (XGBoost, Redes Neuronales, etc.)	Series diarias/horarias con modelos de ML. Ideal cuando no sabés la distribución de los errores.

Tabla 6.1: Guía de selección: cada método tiene su contexto ideal. No existe un "mejor" universal.

Reglas prácticas:

- Si usás **ARIMA o ETS** con supuestos verificados (normalidad de residuos, varianza constante) → Los intervalos de predicción nativos del modelo son suficientes y computacionalmente eficientes.
- Si necesitás la **distribución completa** de posibles futuros, o querés incorporar conocimiento experto (priors) → Enfoque Bayesiano. Particularmente útil cuando hay poca data histórica pero mucho conocimiento del dominio.
- Si usás **modelos de ML** (XGBoost, LightGBM, Redes Neuronales) que no generan intervalos nativamente → Conformal Prediction es la opción más práctica. Pero tené en cuenta que el split-conformal básico asume que los errores de calibración son representativos del futuro; para series con fuerte autocorrelación, existen variantes adaptativas (ACI) que mejoran la cobertura.
- Si hay **cambios estructurales recientes** (pandemia, nueva regulación, cambio de estrategia) → Ningún método de incertidumbre será fiable con datos antiguos. La prioridad es recalibrar con datos recientes, independientemente del enfoque elegido.

Nota técnica: En forecasting, lo que calculamos son **intervalos de predicción** (rango donde esperamos que caigan las observaciones futuras), no intervalos de confianza en sentido estricto (rango para un parámetro poblacional). La distinción es sutil pero relevante: los intervalos de predicción son siempre más anchos porque incluyen tanto la incertidumbre del modelo como la variabilidad inherente de los datos.

6.6 Conclusión

Una predicción sin incertidumbre es como un mapa sin escala:
técnicamente correcto, pero peligrosamente incompleto.

1. **Los Intervalos Clásicos** son elegantes pero frágiles: dependen de que los supuestos se cumplan.
2. **El Forecast Bayesiano** es el más rico conceptualmente, pero también el más costoso.
3. **Conformal Prediction** es el más práctico para modelos de ML: funciona sin supuestos de distribución.

En el próximo capítulo, escalaremos nuestros modelos a miles de series simultáneamente con **Modelos Globales**.

CAPÍTULO 7

Modelos globales

7.1 Introducción: el problema de los 10,000 productos

Imaginá que sos el Data Scientist de un gran retailer. Tenés 10,000 productos en tu catálogo. Hasta ahora, hemos aprendido a entrenar un modelo para una serie (univariado). Si seguimos esa estrategia, tendríamos que entrenar, mantener y monitorear 10,000 modelos diferentes.

Esto tiene dos problemas graves:

1. **Costo computacional:** Entrenar 10,000 modelos cada noche es lento y caro.
2. **Desperdicio de información:** El modelo que predice la "Coca-Cola" no sabe nada de lo que aprendió el modelo de la "Pepsi". Si ambos son refrescos, deberían compartir patrones (ej: suben en verano).

7.2 La solución: modelos globales

Un **Modelo Global** es un único modelo que se entrena con los datos de todos los productos a la vez. Es como un médico generalista que ha visto miles de pacientes, en lugar de un médico que solo ha visto a un paciente en toda su vida.

7.2.1 El formato de datos: Panel

Para entrenar un Modelo Global, necesitamos que todos los productos estén en una sola tabla. Esto se llama **formato de Panel o Long Format**. Cada fila tiene una columna `Product_ID` que identifica a qué producto pertenece:

Product_ID	Fecha	Ventas	dow	lag_1	lag_7	roll_mean_7
A_BestSeller	2015-12-01	135.2	1	128.4	132.1	130.5
A_BestSeller	2015-12-02	142.8	2	135.2	138.7	131.2
B_Trending	2015-12-01	155.3	1	148.9	145.2	147.6
B_Trending	2015-12-02	162.1	2	155.3	150.8	149.3
D_Seasonal	2015-12-01	28.5	1	25.1	30.2	27.8

Tabla 7.1: Formato Panel. Todos los productos en una sola tabla. Las features (dow, lags, rolling mean) se calculan dentro de cada producto usando `groupby`, para no mezclar información entre series.

El modelo recibe esta tabla completa y aprende patrones que son comunes a todos los productos ("los sábados se vende más") y también patrones específicos de cada producto (a través de la columna `Product_ID` codificada numéricamente).

7.3 El experimento: local vs global

7.3.1 Diseño del experimento

Hemos simulado 4 productos con personalidades muy diferentes para poner a prueba la hipótesis de que un solo modelo puede manejar toda esta diversidad:

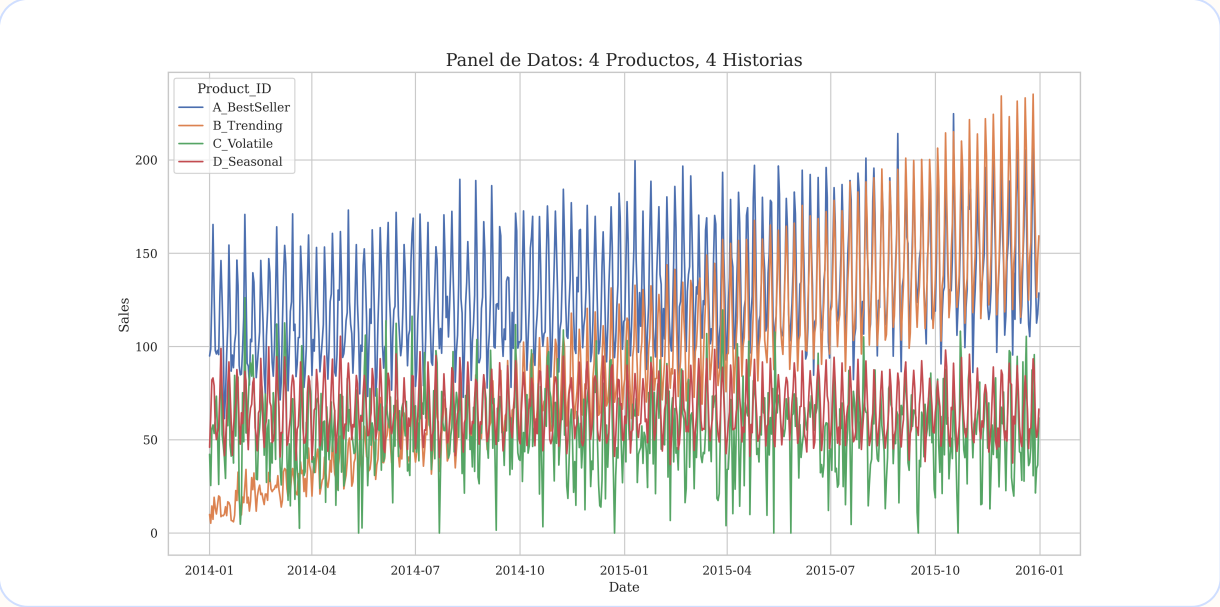


Figura 7.1: Visualización del panel de datos. Cuatro productos con comportamientos muy distintos: A (azul) tiene ventas altas y estables, B (naranja) crece con el tiempo, C (verde) es volátil, y D (rojo) tiene picos estacionales marcados.

- **A (Best Seller):** Ventas altas (~100-150/día) y estables. Poca variabilidad.
- **B (Trending):** Producto nuevo que arranca con ventas bajas (~10/día) y crece hasta ~150/día.
- **C (Volátil):** Ventas medias (~50/día) pero con mucho ruido. Difícil de predecir.
- **D (Estacional):** Ventas bajas (~30/día) con picos marcados los fines de semana (hasta 80/día).

Entrenamos dos enfoques y los comparamos:

1. **Enfoque local:** 4 modelos XGBoost independientes, cada uno entrenado solo con los datos de su producto. Cada modelo solo "conoce" su propia historia.
2. **Enfoque global:** 1 solo modelo XGBoost entrenado con la tabla Panel completa (datos de los 4 productos juntos). Este modelo "ve" todas las historias simultáneamente.

7.3.2 Velocidad

El modelo Global se entrenó en **0.18 segundos**, mientras que los 4 modelos locales tomaron **0.70 segundos** en total. Con solo 4 productos, la diferencia parece menor. Pero extrapolando: con 10,000 productos, el enfoque local tomaría ~30 minutos mientras el Global tomaría ~45 segundos. Con 100,000 productos, la diferencia es de horas vs minutos.

7.4 Resultados de precisión

7.4.1 El fenómeno del Cross-Learning

El resultado más interesante es el Producto D: el Modelo Global mejoró sobre el Local gracias al **Cross-Learning**. El Producto D tiene una estacionalidad semanal muy marcada. Al entrenar juntos,

Producto	MAE Local	MAE Global	Interpretación
A (Best Seller)	12.06	12.12	Empate técnico. Ambos modelos manejan bien productos estables de alto volumen.
B (Trending)	7.02	7.17	El Local gana levemente: B tiene un patrón de crecimiento único que el Global "diluye" al mezclar con otros.
C (Volátil)	18.28	19.62	El Local gana: con tanto ruido, el cross-learning del Global no aporta valor y empeora levemente.
D (Estacional)	5.81	5.44	El Global mejora: tiene más datos para aprender el patrón semanal de todos los productos.

Tabla 7.2: Comparación de errores (MAE). El Global gana claramente en D (Estacional) gracias al cross-learning del patrón semanal. Para los demás, el enfoque Local iguala o supera levemente.

el modelo Global aprendió que "los sábados se vende más" con un pool de datos más grande, y aplicó ese conocimiento con mayor confianza a las predicciones de D.

Para los otros productos, el enfoque Local igualó o superó levemente al Global. Esto es esperable con solo 4 productos: el Cross-Learning brilla cuando hay cientos o miles de series compartiendo patrones comunes. Con pocos productos, cada modelo Local tiene suficientes datos por sí solo. ¿Cuándo conviene cada enfoque? Lo formalizamos en la siguiente sección.

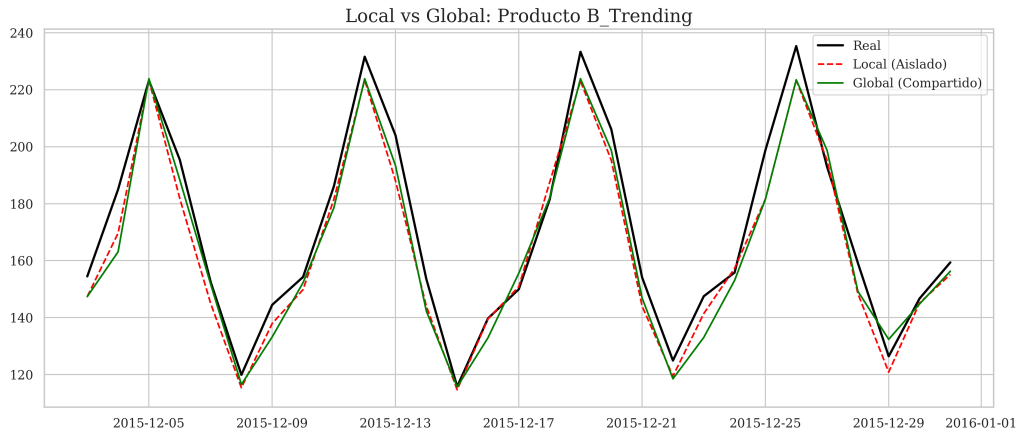


Figura 7.2: Predicción del Producto B (Trending) en los últimos 28 días. Ambos modelos capturan el patrón semanal. La diferencia visual es sutil, pero el Local (rojo punteado) se ajusta mejor a los picos, mientras el Global (verde) tiende a suavizar. Para productos con comportamientos únicos, el enfoque local puede ser preferible.

7.5 Cuándo global, cuándo local, cuándo híbrido

El experimento anterior ilustra un patrón general. Criterios prácticos para elegir:

El **enfoque híbrido** merece énfasis: entrenás un modelo Global sobre todas las series, luego entrenás un modelo local liviano sobre los residuos de las series que consistentemente muestran error alto. Esto captura patrones universales (vía Global) y comportamiento idiosincrático (vía corrección local) sin

Enfoque	Cuándo usarlo	Por qué
Global	Muchas series (>50), pocos datos por serie (<100 obs), patrones similares	Cross-learning compensa la escasez de datos individuales
Local	Pocas series, datos abundantes por serie (>500 obs), patrones únicos	Suficientes datos para aprender individualmente; global diluiría las especificidades
Híbrido	Portfolio mixto: algunas series con muchos datos, otras con pocos	Global como base + ajuste local de residuos para series atípicas

Tabla 7.3: Framework de decisión Global vs Local vs Híbrido. El factor clave es la relación entre datos disponibles y complejidad de patrones por serie.

7.6 Evaluación más allá del MAE

Un solo número de MAE sobre todas las series puede ser engañoso. Un modelo con MAE = 5 puede ser excelente para productos de alto volumen pero terrible para los de bajo volumen (donde 5 unidades pueden significar 100% de error).

MAE por tier: Desglosá la precisión por categoría de producto, tier de volumen o unidad de negocio. Un modelo Global suele brillar en el long tail (series con pocos datos) pero rendir peor en los top sellers versus un modelo Local bien afinado. Reportar por tier expone estos trade-offs.

Calibración de intervalos de predicción: Si producís intervalos al 90% (Capítulo 6), verificá si realmente contienen el valor real el 90% de las veces. Dos métricas importan:

- **Coverage Rate (MCR):** Fracción de valores reales que caen dentro del intervalo. Objetivo: igualar el nivel nominal (ej: 90%).
- **Mean Scaled Interval Width (MSIW):** Qué tan anchos son los intervalos. Angostos + bien calibrados es lo ideal. Intervalos anchos que logran 99% de coverage son poco informativos.

Un modelo con buen MAE pero pobre coverage es peligroso para planning de inventario: da predicciones precisas pero sobreconfiadas.

7.7 Clustering de series antes de entrenar

Cuando tu catálogo tiene series muy heterogéneas (ej: alimentos congelados vs electrónica vs decoración estacional), entrenar un solo modelo Global lo fuerza a comprometerse. Un enfoque intermedio: **modelos globales por cluster**.

1. Agrupá las series por similitud: por categoría, por patrón de demanda (suave vs intermitente), o por características estadísticas (media, varianza, fuerza de estacionalidad).
2. Entrenás un modelo Global por cluster.
3. Cada modelo se especializa en su grupo mientras se beneficia del cross-learning dentro del cluster.

Esto balancea escalabilidad (menos modelos que full local) con especificidad (mejor ajuste que un único global). Investigación reciente (2025) confirma que los enfoques por cluster mejoran consistentemente sobre estrategias puramente globales y puramente locales en datasets heterogéneos.

7.8 Ensembling: la ganancia más simple

Uno de los hallazgos más consistentes en competiciones de forecasting (M4, M5) es que **promediar predicciones de 3–5 modelos diversos supera a cualquier modelo individual**. El ensemble no necesita ser sofisticado: un promedio aritmético simple de LightGBM, XGBoost y CatBoost suele alcanzar.

¿Por qué funciona? Cada modelo comete errores diferentes. Promediar cancela los errores individuales, produciendo un pronóstico más suave y robusto. Contraintuitivamente, optimizar los pesos del ensemble (en vez de pesos iguales) suele empeorar las cosas, porque la estimación de pesos introduce su propio error.

7.9 Retraining en producción

Una pregunta común al desplegar modelos Globales: **¿cada cuánto hay que reentrenar?** La respuesta intuitiva es "lo más seguido posible", pero investigación reciente (2025) desafía esto. Estudios sobre grandes datasets de retail muestran que modelos reentrenados mensualmente rinden casi idéntico a modelos reentrenados diariamente, con menos de 5% de diferencia en accuracy incluso en el escenario de no reentrenar.

La implicancia práctica: empezá con retraining semanal o mensual y solo aumentá la frecuencia si el monitoreo muestra degradación significativa. Esto ahorra recursos computacionales importantes, especialmente a escala.

7.10 Conclusión

Los Modelos Globales son el estándar moderno en grandes empresas (Amazon, Uber, Meta). Permiten:

- Escalar a millones de series sin entrenar millones de modelos.
- Mejorar predicciones en productos con poca data gracias al Cross-Learning.
- Predecir productos nuevos ("Cold Start") usando patrones universales.
- Reducir costos de mantenimiento (1 modelo vs 10,000).

Sin embargo, no son una solución universal. Para productos con comportamientos muy particulares (como el Producto B), un enfoque híbrido o local puede ser preferible.

Los Modelos Fundacionales (Capítulos 8–9) llevan la idea Global a su extremo: un solo modelo pre-entrenado con miles de millones de puntos temporales de dominios diversos. Donde nuestro modelo Global aprendió de miles de series de una empresa, los Modelos Fundacionales aprenden de millones de series de múltiples industrias. El trade-off es control: ganás capacidad zero-shot pero perdés la posibilidad de afinar el feature engineering. Entender cuándo conviene cada enfoque es el tema de los próximos dos capítulos.

CAPÍTULO 8

Modelos Fundacionales: teoría

8.1 Introducción: más allá del entrenamiento

Hasta ahora, siempre hemos seguido una regla de oro: “Para predecir una serie, debés entrenar con esa serie”. Incluso en el Modelo Global (Capítulo 7), el modelo necesitaba ver los datos de todos los productos durante el entrenamiento.

Pero, ¿qué pasa si llega un producto nuevo hoy y queremos predecir sus ventas mañana? No tenemos tiempo para re-entrenar. Este es el problema del **Cold Start** , y es uno de los desafíos más comunes en la industria.

8.2 El problema del Cold Start

8.2.1 ¿Cuándo ocurre?

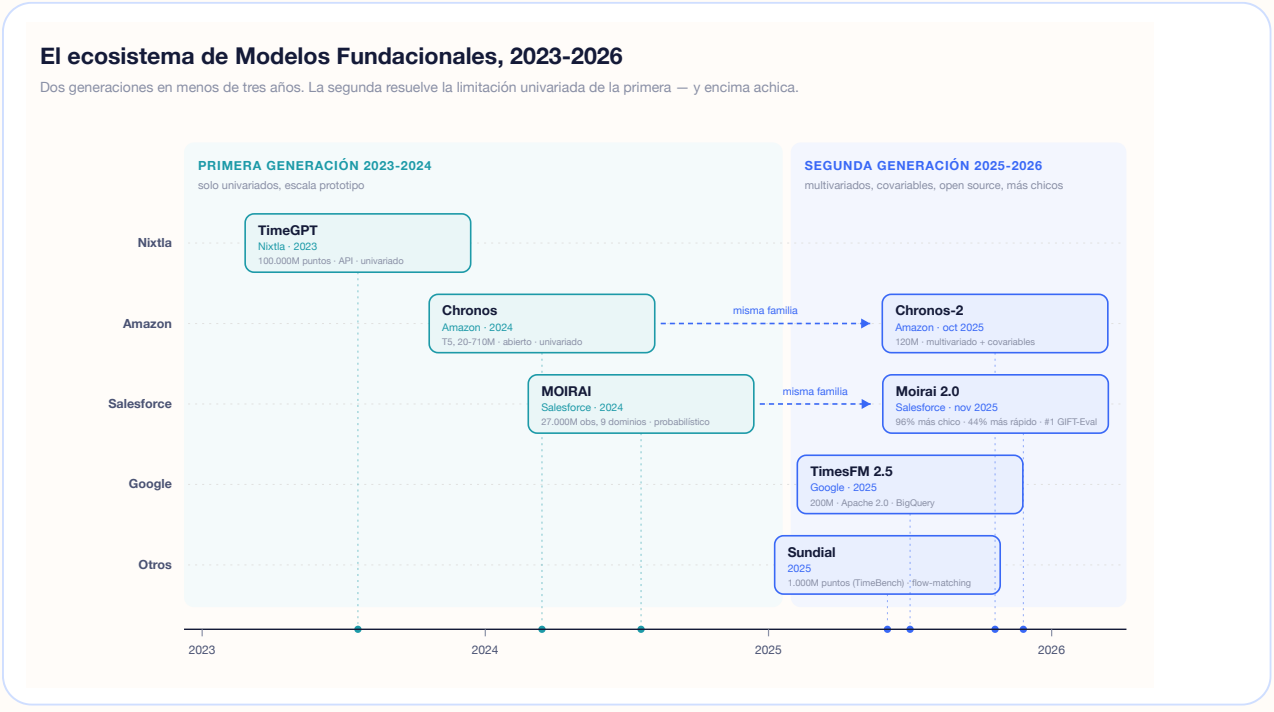
El Cold Start (arranque en frío) aparece en múltiples escenarios reales:

- **Lanzamiento de producto nuevo:** Una tienda lista un producto que nunca se vendió antes. No hay historial.
- **Expansión geográfica:** Una cadena abre una tienda en una ciudad nueva. Los patrones de consumo pueden ser diferentes.
- **Productos estacionales:** Un artículo navideño que solo se vende 2 meses al año. Cuando llega noviembre, solo tenés datos del año pasado.
- **Productos de baja rotación:** Repuestos industriales que se venden 3 veces al año. No hay suficiente historia para un modelo estadístico.

8.2.2 ¿Por qué los métodos tradicionales fallan?

Los modelos que hemos aprendido hasta ahora requieren historial:

- **ARIMA (Cap 3):** Necesita al menos 2-3 ciclos estacionales completos (24-36 meses para datos mensuales) para estimar parámetros.
- **XGBoost (Cap 4):** Necesita suficientes filas para calcular lags. Con 7 días de datos, `lag_7` no existe.
- **Modelo Global (Cap 7):** Puede ayudar parcialmente (si el producto nuevo se parece a los existentes), pero necesita al menos algunos datos para calcular features.



8.2.3 Estrategias modernas

1. Transferencia por similitud: Usar metadatos del producto (categoría, precio, marca) para encontrar productos "hermanos" con historial y transferir sus patrones. Amazon Forecast reporta mejoras significativas en precisión con este enfoque.

La industria ha desarrollado tres enfoques principales para el Cold Start:

2. Meta-Learning: Entrenar un modelo que "aprenda a aprender": dado un producto nuevo con pocos datos, el modelo adapta rápidamente sus parámetros. Investigaciones recientes con el dataset M5 muestran reducciones sustanciales del error usando solo 7 días de observaciones.

En el Capítulo 10 (Sección Cold Start) profundizamos en las estrategias 1 y 2 con ejemplos prácticos y una tabla comparativa. Aquí nos enfocamos en la estrategia 3: zero-shot con Foundation Models.

3. Zero-Shot Forecasting: Modelos fundacionales que predicen sin haber visto nunca la serie. Esta es la frontera actual.

8.3 Zero-shot forecasting: predecir sin entrenar

Es la capacidad de un modelo para predecir una serie de tiempo que nunca ha visto antes, sin necesidad de re-entrenamiento.

Es análogo a los modelos de lenguaje como GPT: no entrenás a GPT para que escriba un poema sobre tu empresa específica; él ya sabe escribir poemas porque leyó millones de textos. Le das el "prompt" (el inicio de tu serie) y él completa el resto, basándose en patrones universales que aprendió durante su pre-entrenamiento masivo.

8.3.1 Nuestro experimento

Para demostrarlo, hicimos un experimento con nuestro mini-retailer simulado:

1. Entrenamos un modelo Global solo con los productos A, B y C.
2. Le ocultamos totalmente el Producto D (Nuevo).
3. Le pedimos que prediga el Producto D usando solo su conocimiento general sobre "cómo funcionan las ventas".

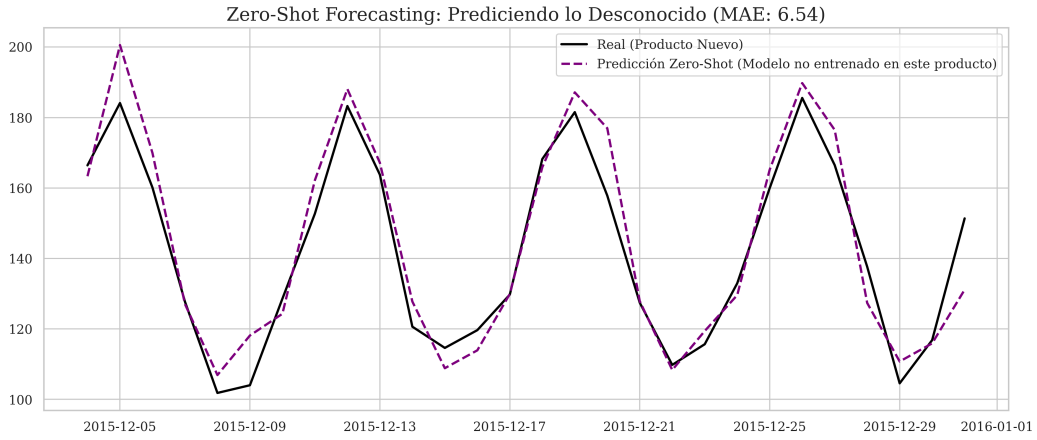


Figura 8.1: Predicción Zero-Shot. El modelo (línea morada) predice razonablemente bien el Producto Nuevo (línea negra) sin haberlo visto nunca, basándose en patrones universales como la estacionalidad semanal.

El modelo entendió que "hay una estacionalidad semanal" (los sábados se vende más) y la aplicó al nuevo producto. No fue perfecto, pero fue una predicción útil sin ningún dato previo del producto.

8.4 Transformers para series de tiempo

Nuestro experimento usó XGBoost, pero la verdadera revolución viene de la arquitectura **Transformer** , originalmente diseñada para procesamiento de lenguaje natural (traducción, generación de texto) y que se ha adaptado con éxito a series de tiempo.

8.4.1 ¿Qué es la atención?

Imaginá que estás leyendo una frase larga y necesitás recordar una palabra que estaba al principio para entender el final. Los modelos tradicionales (RNNs, LSTMs) procesan la secuencia paso a paso, lo que hace que la información lejana se "diluya" con el tiempo (el problema del vanishing gradient).

El mecanismo de **Atención** resuelve esto permitiendo al modelo conectar directamente cualquier punto del pasado con el presente, sin importar la distancia temporal. El modelo "decide" en qué puntos del pasado enfocarse para hacer cada predicción.

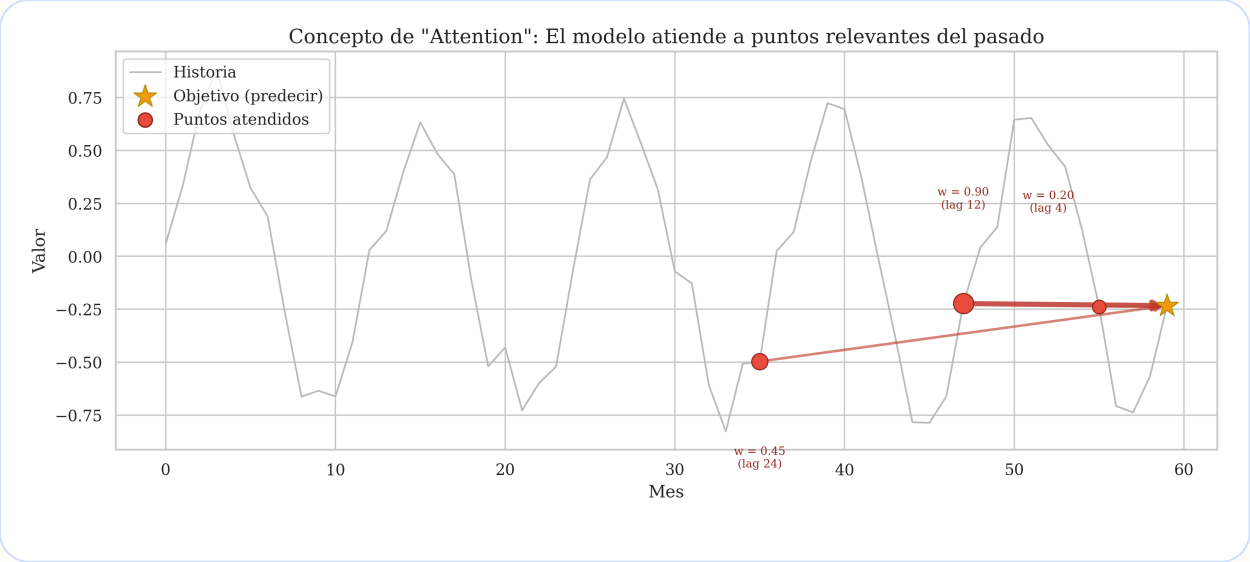


Figura 8.2: Concepto de Atención. Para predecir el valor objetivo (estrella), el modelo asigna diferentes pesos a distintos pasos temporales del pasado (puntos rojos). La conexión más fuerte va al valor de hace 12 meses: el modelo aprende a explotar la estacionalidad anual. A diferencia de las RNNs, estas conexiones son directas y paralelas.

8.4.2 Componentes clave de un Transformer para series de tiempo

Los Transformers adaptados a series de tiempo incluyen:

1. **Patching (parches):** En vez de alimentar al modelo punto por punto (como haría una RNN), la serie se divide en "parches" o subsecuencias. Por ejemplo, una serie diaria de 180 días se divide en 30 parches de 6 días cada uno. Cada parche se trata como un "token" (similar a una palabra en un LLM). Esto reduce la complejidad computacional y permite capturar patrones locales.
2. **Multi-Head Attention:** El modelo tiene múltiples "cabezas" de atención que aprenden diferentes tipos de relaciones simultáneamente. Una cabeza puede enfocarse en la estacionalidad semanal, otra en la tendencia a largo plazo, y otra en eventos puntuales.
3. **Codificación posicional:** Como el Transformer procesa todos los parches en paralelo (no secuencialmente), necesita saber el orden de cada parche. Se añade una señal que codifica la posición temporal.
4. **Predicción directa vs autoregresiva:** Los Transformers para series de tiempo suelen usar predicción directa: generan todos los valores futuros de una vez, en lugar de predecir uno por uno y alimentar la predicción anterior como input (que acumula errores).

8.5 Los Modelos Fundacionales

Un **modelo fundacional** es un Transformer que ha sido pre-entrenado con una cantidad masiva de datos de múltiples dominios (finanzas, energía, retail, clima, tráfico) y que puede hacer predicciones en cualquier dominio sin entrenamiento adicional.

8.5.1 ¿Cómo se entrenan?

El proceso de pre-entrenamiento es análogo al de los LLMs de lenguaje:

1. Recopilación masiva de datos: Se juntan millones o miles de millones de series de tiempo de fuentes públicas y privadas.

2. Entrenamiento del Transformer: El modelo aprende a predecir el "siguiente valor" (o parche) dada la historia, igual que un LLM aprende a predecir la "siguiente palabra".

3. Transferencia: Una vez entrenado, el modelo puede aplicarse a cualquier serie nueva sin re-entrenar. Solo necesita la historia como input ("prompt") y genera la predicción.



8.5.2 El ecosistema de Modelos Fundacionales

El campo ha evolucionado a una velocidad notable. En menos de tres años (2023-2026), hemos pasado de los primeros prototipos a un ecosistema maduro con múltiples alternativas competitivas.

Primera generación (2023-2024)

TimeGPT (Nixtla, 2023): El primer modelo fundacional comercial para series de tiempo. Entrenado con más de 100 mil millones de puntos de datos. Se accede vía API. Su fortaleza es la facilidad de uso: una sola llamada genera predicciones con intervalos de confianza. Limitación: solo univariado, sin soporte nativo de covariables.

Chronos (Amazon, 2024): Modelo open source que tokeniza las series, convirtiendo valores numéricos en categorías discretas. Usa la familia T5 de Google (20M a 710M parámetros). Pre-entrenado con datasets públicos y datos sintéticos. Limitación: solo univariado.

MOIRAI (Salesforce, 2024): Entrenado con el archivo LOTSA: 27 mil millones de observaciones de 9 dominios. Funciona con cualquier frecuencia y genera predicciones probabilísticas nativamente. Usa múltiples tamaños de parche y distribuciones mixtas.

Segunda generación (2025-2026)

La segunda generación resuelve varias limitaciones de la primera:

Sundial (2025): Introduce TimeFlow Loss basada en flow-matching, una alternativa a la tokenización de Chronos.
Pre-entrenado con mil millones de puntos (TimeBench). Logra inferencia en milisegundos y genera múltiples predicciones probables. Representa un enfoque fundamentalmente diferente: en vez de tokenizar la serie (discretizar),
trabaja directamente con valores continuos.

TimesFM 2.5 (Google, 2025): Modelo decoder-only de 200M parámetros, pre-entrenado con 100 mil millones de puntos. Open source (Apache 2.0) e integrado en Google BigQuery, lo que facilita su uso en entornos empresariales sin necesidad de infraestructura de ML propia.

Moirai 2.0 (Salesforce, Nov 2025): Cambio de arquitectura de masked-encoder a decoder-only. Es 96% más pequeño y 44% más rápido que Moirai 1.0-Large, con mejor precisión. Ocupa el primer puesto en GIFT-Eval (uno de los benchmarks de referencia). Demuestra que en modelos fundacionales, "menos puede ser más".

covariables mediante un mecanismo de atención grupal. Genera más de 300 predicciones por segundo en una GPU. Resuelve la principal limitación de la primera generación (solo univariado).

Chronos-2 (Amazon, Oct 2025): Un salto significativo respecto al original. Con 120M parámetros y arquitectura encoder-only, ahora soporta forecasting multivariado y

8.5.3 Tendencias del campo

Observando la evolución de 2023 a 2026, se identifican varias tendencias claras:

- **De univariado a universal:** Los modelos pasan de predecir una sola serie a manejar múltiples series y covariables simultáneamente.
- **Más pequeños y más rápidos:** Moirai 2.0 es 96% más pequeño que su predecesor con mejor rendimiento. La eficiencia es una prioridad.
- **Open source dominante:** La mayoría de los modelos de segunda generación son open source, a diferencia de TimeGPT que inició como servicio comercial.
- **Diversidad de enfoques:** Tokenización (Chronos), distribuciones mixtas (MOIRAI), flow-matching (Sundial), arquitecturas multi-resolución. No hay un paradigma único ganador.

8.6 Conclusión

Ahora conocemos la teoría detrás de los Modelos Fundacionales: cómo funcionan los Transformers, qué es la Atención, cómo se adaptan a series de tiempo con patching, y cómo se pre-entrenan con datos masivos. También hemos visto cómo el ecosistema ha madurado rápidamente, con una segunda generación que resuelve muchas limitaciones de la primera.

En el próximo capítulo, evaluaremos estos modelos en la práctica contra nuestros baselines (SARIMA y XGBoost).

CAPÍTULO 9

Modelos Fundacionales en la práctica

9.1 Introducción

En el Capítulo 8, estudiamos la teoría detrás de los Modelos Fundacionales: Transformers, Atención, Patching y pre-entrenamiento masivo. En este capítulo, evaluaremos estos modelos en la práctica, comparándolos contra los baselines que hemos construido a lo largo del ebook.

Evaluaremos tres modelos fundacionales:

1. **TimeGPT (Nixtla):** Modelo comercial, acceso vía API.
2. **Chronos (Amazon):** Open source, basado en tokenización.
3. **MOIRAI (Salesforce):** Open source, predicciones probabilísticas.

Los enfrentaremos contra SARIMA (Cap 3) y XGBoost (Cap 4) en el mismo dataset de retailer simulado, con los últimos 28 días reservados como test.

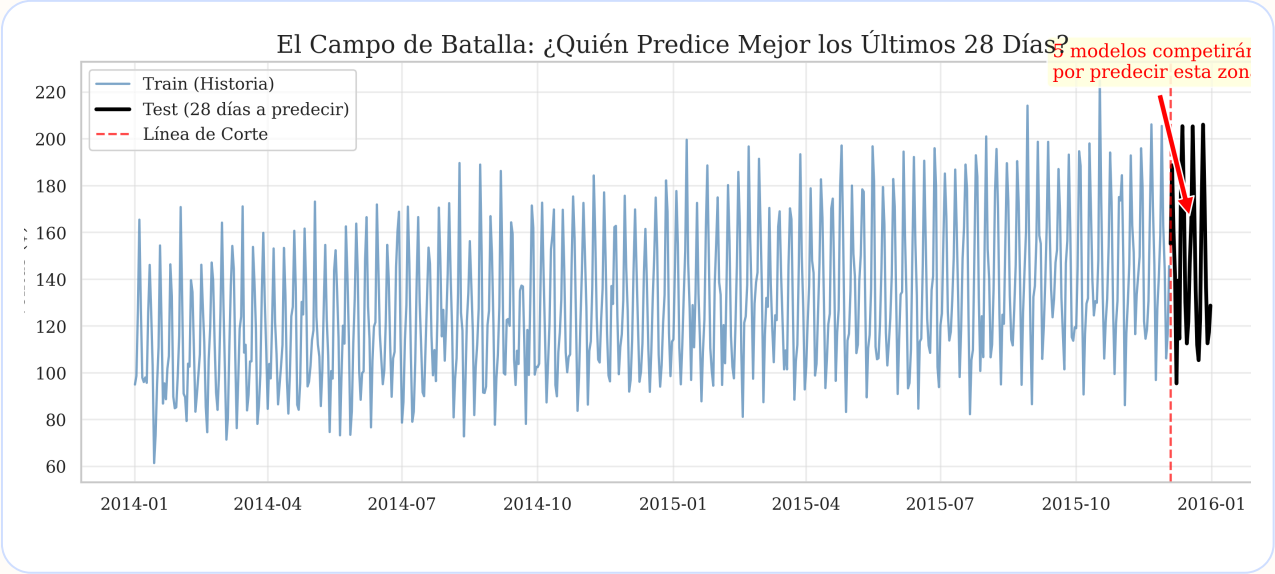


Figura 9.1: Los datos de evaluación: ventas simuladas de retailer. Los últimos 28 días (negro) son el test que todos los modelos deben predecir.

9.2 Caso 1: TimeGPT (Nixtla)

9.2.1 Características

TimeGPT fue el primer modelo fundacional comercial para series de tiempo. Fue entrenado con más de 100 mil millones de puntos de datos de dominios variados.

Su propuesta de valor es la simplicidad: no requiere feature engineering, no requiere elegir hiperparámetros, y no requiere entrenamiento. Se le pasan los datos históricos y genera la predicción con intervalos de confianza incluidos.

9.2.2 Resultados

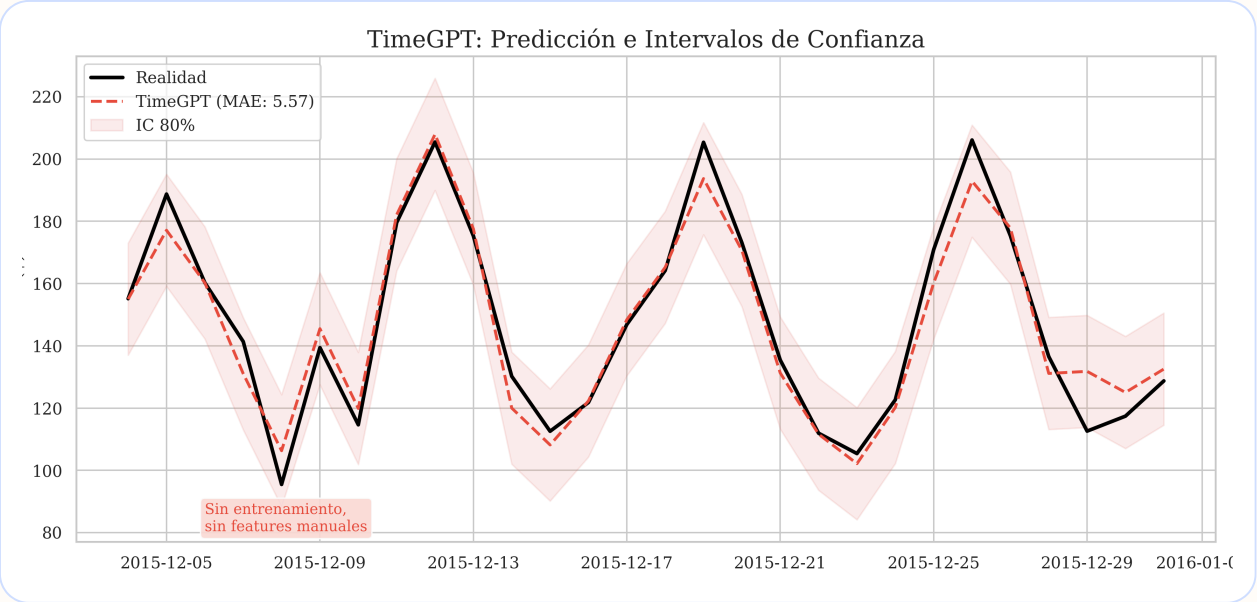


Figura 9.2: TimeGPT: predicción e intervalos de confianza al 80%. MAE de 5.57, superando a ambos baselines sin necesidad de configuración.

TimeGPT logra un MAE de **5.57**, inferior al de SARIMA (8.08) y XGBoost (12.04). Captura la estacionalidad semanal sin que se lo hayamos indicado explícitamente.

9.2.3 Limitaciones

- Requiere API key y tiene costo por uso.
- Los datos deben enviarse a servidores externos (posible problema de privacidad).
- Al ser un modelo de caja negra, la interpretabilidad es nula.

9.3 Caso 2: Chronos (Amazon)

9.3.1 Características

Chronos innova en cómo representa las series de tiempo: las **tokeniza** . Convierte valores numéricos en categorías discretas (tokens), como si la serie fuera un texto. Usa la familia T5 de Google (20M a 710M parámetros).

Chronos logra un MAE de **6.30**. Ligeramente superior (peor) que TimeGPT, pero con una ventaja: es completamente open source y gratuito. No requiere enviar datos a servidores externos.

Nota: En octubre de 2025, Amazon lanzó Chronos-2, que soporta forecasting multivariado y covariables (limitación del original), con 120M parámetros y más de 300 predicciones por segundo.

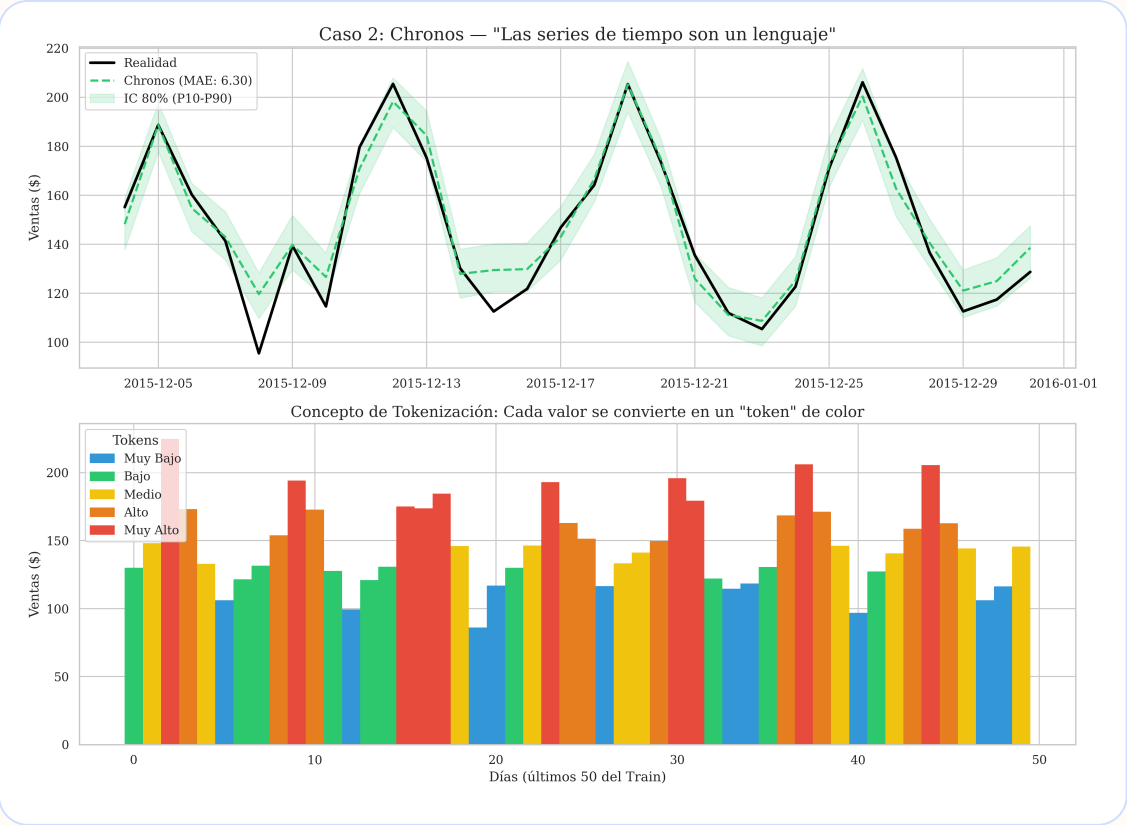


Figura 9.3: Arriba: Predicción de Chronos con intervalos probabilísticos (MAE: 6.30). **Abajo:** Visualización del concepto de tokenización: cada barra tiene un color según su categoría (token).

9.4 Caso 3: MOIRAI (Salesforce)

9.4.1 Características

MOIRAI funciona con cualquier frecuencia (horaria, diaria, mensual) sin cambiar configuración.
Genera predicciones probabilísticas nativamente, produciendo cientos de escenarios del futuro.

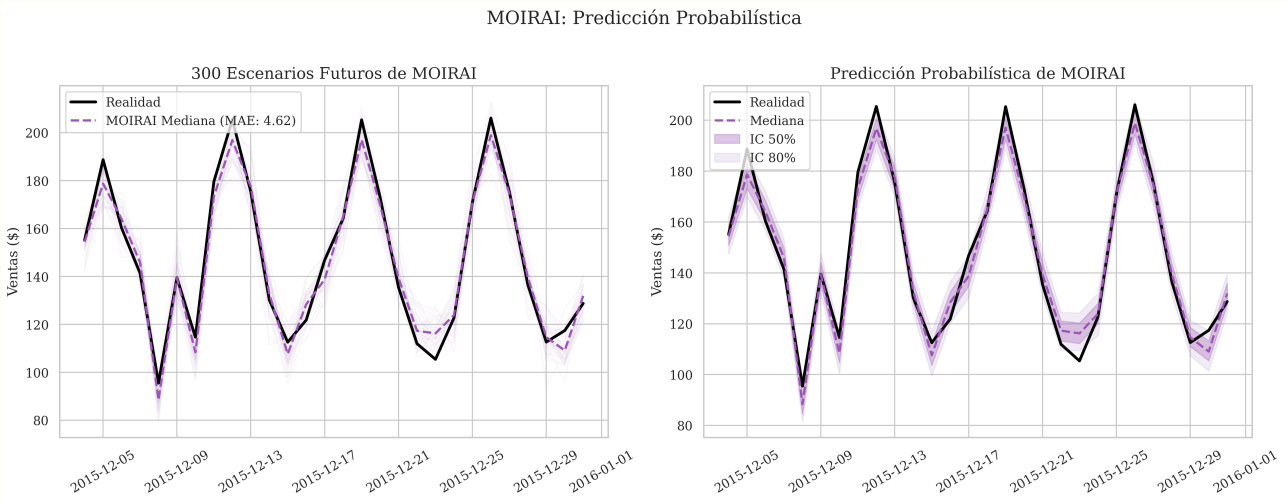


Figura 9.4: Izquierda: 300 escenarios futuros generados por MOIRAI. **Derecha:** Intervalos derivados de esas trayectorias. MAE de la mediana: 4.62.

MOIRAI obtiene el mejor MAE del benchmark: **4.62**. Su enfoque probabilístico permite no solo predecir el valor esperado, sino cuantificar la incertidumbre completa.

Nota: En noviembre de 2025, Salesforce lanzó Moirai 2.0, un modelo 96% más pequeño y 44% más rápido que MOIRAI 1.0, con mejor precisión. Ocupa el primer puesto en el benchmark GIFT-Eval.

9.5 El benchmark consolidado

Modelo	MAE	Features	Entren.	Costo	Open Source	Tipo
SARIMA	8.08	No	Sí	Gratis	Sí	Clásico
XGBoost	12.04	Sí (manual)	Sí	Gratis	Sí	ML
TimeGPT	5.57	No	No	Pago	No	Fundacional
Chronos	6.30	No	No	Gratis	Sí	Fundacional
MOIRAI	4.62	No	No	Gratis	Sí	Fundacional

Tabla 9.1: Benchmark completo. La columna "Entren." indica si el modelo necesita entrenarse con los datos del problema específico.[2]

9.5.1 Interpretación de los resultados

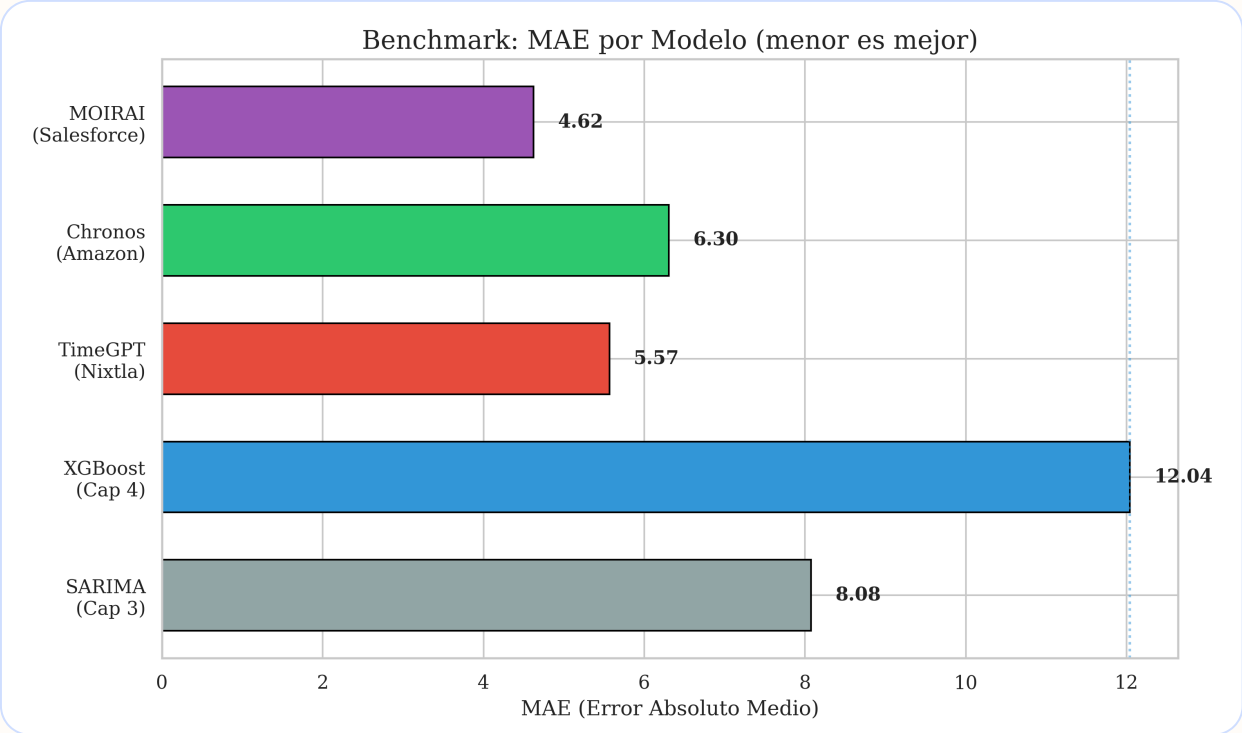


Figura 9.5: MAE de cada modelo (menor es mejor).

Los resultados revelan varios hallazgos relevantes:

1. En este benchmark simulado (ver nota al pie para detalles metodológicos), los tres modelos fundacionales superan a ambos baselines. MOIRAI (4.62), TimeGPT (5.57) y Chronos (6.30) obtienen errores menores que SARIMA (8.08) y XGBoost (12.04). Esto ocurre sin feature engineering, sin entrenamiento específico y sin configuración de hiperparámetros.

2. XGBoost es el peor del benchmark. Esto puede parecer contradictorio con el Capítulo 4, donde XGBoost superó a SARIMA. La diferencia es que en el Cap 4 usamos un set de features completo (lag_1, lag_7, lag_365, rolling_mean, rolling_std, day_of_year, is_weekend). Aquí usamos un set simplificado (sin lag_365 ni is_weekend). La lección: **XGBoost sin buenas features es peor que ARIMA**. La potencia del ML depende totalmente del feature engineering. Con la familia Boosting completa (LightGBM, CatBoost) y features avanzados (Fourier, target encoding) como los del Capítulo 4, los resultados de ML serían significativamente mejores.

3. SARIMA sorprende. Con cero esfuerzo de configuración (más allá de elegir los parámetros), obtiene un MAE de 8.08, superando a XGBoost. Los modelos clásicos bien configurados siguen siendo competitivos.

4. MOIRAI lidera. Su enfoque probabilístico (múltiples escenarios del futuro) le permite capturar mejor la incertidumbre y generar predicciones más precisas. Además, es open source y gratuito.

5. Importante caveat: Este benchmark usa un dataset simulado con patrones estacionales claros y modelos de primera generación (2024). Con las versiones más recientes (Chronos-2, Moirai 2.0), los resultados serían probablemente más favorables para los fundacionales. En datos reales con ruido, variables exógenas complejas o cambios estructurales, los resultados podrían diferir. Por eso, la siguiente sección es fundamental.

9.6 La otra cara de la moneda: limitaciones y críticas

Los resultados del benchmark son favorables para los modelos fundacionales. Sin embargo, la literatura científica reciente (2024-2025) presenta un panorama más matizado.

9.6.1 Crítica 1: no siempre superan a modelos simples

En un benchmark de predicción de precios de electricidad en mercados europeos (2024), el modelo estadístico MSTL (Multi-Seasonal-Trend Decomposition using Loess), un método relativamente simple que captura estacionalidades diarias y semanales, obtuvo resultados superiores y más consistentes que Chronos, TimeGPT, MOIRAI y otros modelos fundacionales.

Lección: En dominios donde los patrones estacionales son claros y bien definidos, un modelo estadístico bien configurado puede ser igual o mejor que un modelo fundacional de miles de millones de parámetros.

9.6.2 Crítica 2: el problema de la generalización

Una investigación de 2025 argumenta que los modelos fundacionales para series de tiempo cometen un error de categoría: a diferencia del lenguaje (donde las palabras tienen significado compartido), las series de tiempo de diferentes dominios no comparten semántica. Un pico en datos financieros, un pico en un electrocardiograma y un pico en vibraciones de una turbina tienen causas completamente diferentes.

Los modelos fundacionales aprenden propiedades estadísticas genéricas (estacionalidad, tendencia), pero no las causas subyacentes. Esto limita su capacidad de generalización real.

9.6.3 Crítica 3: zero-shot depende del dominio de pre-entrenamiento

El rendimiento zero-shot depende fuertemente de si el modelo fue pre-entrenado con datos similares al dominio objetivo. Si tu serie pertenece a un dominio poco representado en los datos de entrenamiento (ej: datos biomédicos especializados), el rendimiento puede ser mediocre.

9.6.4 Crítica 4: covariables y variables exógenas

Los modelos fundacionales de primera generación (TimeGPT, Chronos, MOIRAI) tenían dificultades para incorporar variables exógenas (precio, clima, promociones). En contraste, XGBoost puede integrar cualquier feature adicional de forma natural.
Esta limitación está siendo resuelta por la segunda generación: **Chronos-2 (Oct 2025)** ya soporta covariables nativamente mediante su mecanismo de atención grupal. Sin embargo, la integración de covariables en modelos fundacionales sigue siendo un área de investigación activa, y para problemas donde las variables externas son el factor dominante (ej: impacto de promociones en ventas), los modelos de ML clásicos con features manuales pueden seguir siendo preferibles por su control y explicabilidad.

9.7 ¿Cuándo Foundation Model, cuándo no?

Para la guía general de selección de modelos (incluyendo SARIMA, Boosting y Modelos Globales), consultá el Mapa de Ruta al inicio del libro. Aquí nos enfocamos en la pregunta específica: **¿cuándo un Foundation Model agrega valor frente a las alternativas?**

Situación	Foundation Model	Alternativa preferible
Prototipo rápido, sin tiempo para feature engineering	Sí (zero-shot)	–
Cold start: producto nuevo sin historia	Sí (Chronos, MOIRAI)	Transfer por similitud (Cap 10)
Variables exógenas dominan (precio, promos)	No	Boosting con features (Cap 4)
Dominio con patrones claros y bien definidos	Evaluar	SARIMA bien configurado puede igualar (Cap 3)
50+ series, necesidad de escalar	Evaluar	Global Model + ensembling (Cap 7)
Necesidad de explicabilidad	No	SARIMA (coef.) o Boosting + SHAP (Cap 4)
Datos sensibles, no pueden salir de la empresa	Solo modelos open source locales (Chronos, MOIRAI)	Boosting o Global local

Tabla 9.2: ¿Cuándo Foundation Model agrega valor? La respuesta depende de la combinación de restricciones del problema.

La recomendación más robusta: usá Foundation Models como baseline rápido y luego evaluá si un modelo entrenado con tus datos (Boosting, Global) lo supera. Si no lo supera, el Foundation Model te ahorra todo el trabajo de feature engineering y mantenimiento. Si lo supera, ya tenés un benchmark sólido contra el cual medir tu modelo. En ambos casos, ganás.

9.8 Conclusión

Los modelos fundacionales representan un avance significativo en la accesibilidad del forecasting: permiten obtener predicciones razonables sin expertise en modelado. Sin embargo, no son una solución universal.

La evidencia científica actual (2024-2026) muestra que:

1. En dominios con patrones bien definidos, los modelos simples pueden igualar o superar a los fundacionales.

2. La generalización real ("entender" cualquier serie) aún es limitada, aunque la segunda generación muestra mejoras significativas.

3. Las variables exógenas, que eran territorio exclusivo de los modelos de ML, comienzan a ser soportadas por la segunda generación (Chronos-2).

El campo evoluciona rápidamente: entre 2023 y 2026 hemos visto dos generaciones completas de modelos. Los principios de evaluación rigurosa (backtesting, métricas adecuadas, comparación contra baselines) siguen siendo más importantes que la elección del modelo en sí.

En el próximo capítulo, enfrentaremos las casuísticas del mundo real que ninguno de estos modelos resuelve automáticamente: demanda intermitente, cold start y detección de anomalías.

CAPÍTULO 10

Desafíos del mundo real

10.1 Introducción

A lo largo de los capítulos anteriores, hemos trabajado con datos relativamente bien comportados: series con tendencia clara, estacionalidad regular y sin valores faltantes. Sin embargo, en la práctica industrial, los datos rara vez son así de limpios.

En este capítulo, abordaremos tres problemas que aparecen constantemente en proyectos reales de forecasting y que requieren técnicas específicas:

1. **Demanda intermitente:** Series donde el 60-70% de los días no se vende nada.
2. **Cold Start:** Productos nuevos sin historia. ¿Cómo predecir sin datos?
3. **Detección de anomalías:** Picos, caídas y eventos raros que contaminan tus modelos.

Y cerraremos con una reflexión sobre los errores más frecuentes en la práctica del forecasting.

10.2 Demanda intermitente: el problema de los ceros

10.2.1 El problema real

Hasta ahora, nuestros datos han sido "bonitos": AirPassengers tiene pasajeros todos los meses, nuestro retailer simulado tiene ventas todos los días. Pero en la realidad del retail, una gran proporción de los SKUs (frecuentemente el 60-80%) tienen **demanda intermitente**.

Usamos datos reales de la competición M5 de Kaggle. El producto `FOODS_1_002` tiene ventas en solo el 34% de los días. El 66% restante, la demanda es cero.

10.2.2 El contrafactual: ¿por qué fallan los modelos del Cap 3-4?

Aplicamos SARIMA y XGBoost a esta serie. Los MAEs son similares (todos cerca de 0.93), pero eso es engañoso: con ventas tan bajas (0-4 unidades), cualquier modelo tiene un MAE pequeño. El problema real se ve en el **sesgo acumulado**: cuántas unidades de más o de menos pedirías al proveedor si usaras estas predicciones durante 28 días.

¿Qué observamos en la Figura 10.2?

- **SARIMA (panel superior):** Predice una línea ondulante alrededor de 1 unidad cada día. Cuando la venta real es 0 (la mayoría de días), el modelo se equivoca sistemáticamente prediciendo venta donde no la hay. Cuando la venta es 8 (el pico del 25 de mayo), SARIMA predice 1: un error enorme.
- **XGBoost (panel medio):** Similar a SARIMA: predice valores continuos bajos. Los lags (ventas de ayer, de hace una semana) son casi siempre 0, así que XGBoost no tiene información útil para distinguir los días con venta de los días sin venta.

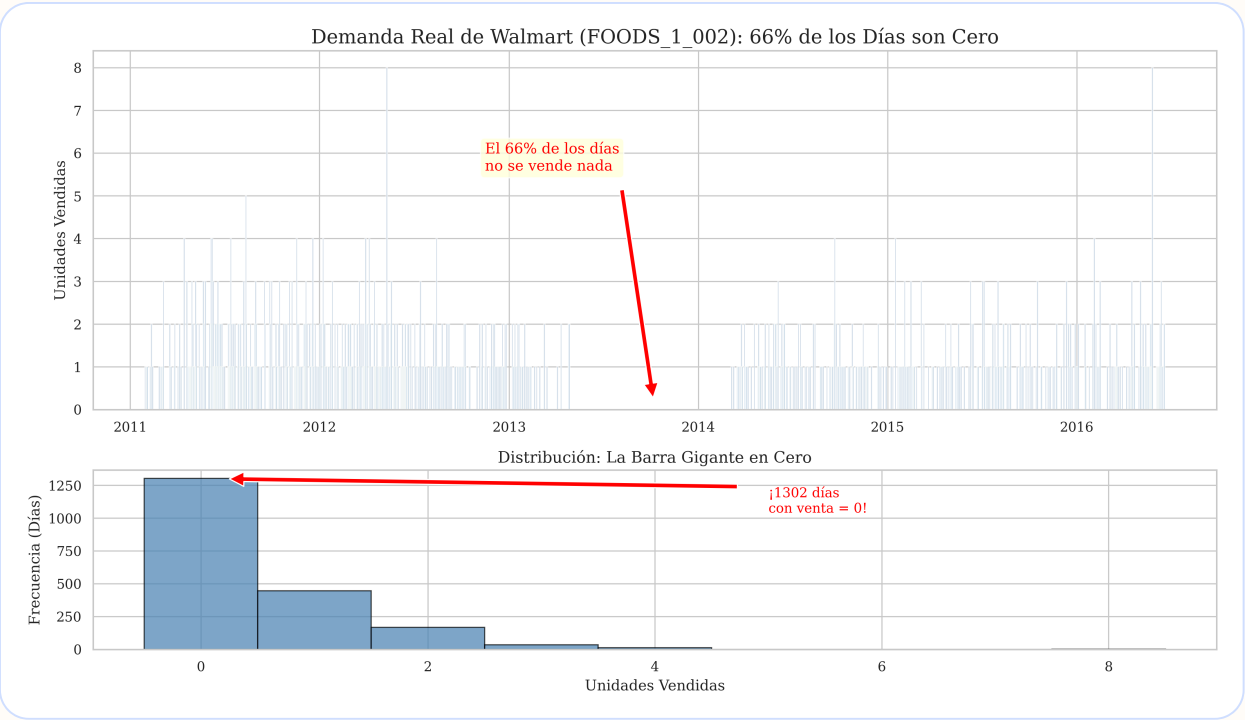


Figura 10.1: Demanda real de un producto de la competición M5. **Arriba:** Ventas esporádicas con mayoría de ceros. **Abajo:** El histograma confirma que la barra en cero domina completamente.

- **Croston SBA (panel inferior):** Predice una línea plana constante (~0.7 unidades/día). A primera vista parece "igual de malo". Pero la diferencia es conceptual: Croston no intenta predecir el día exacto de la venta, sino la tasa promedio de demanda, que es lo relevante para gestión de inventario.

10.2.3 La solución: método de Croston-SBA

El método de Croston (1972) descompone el problema en dos sub-problemas independientes:

1. ¿Cada cuántos días hay demanda? Modela los intervalos entre demandas no-cero usando Suavizado Exponencial Simple (SES).

2. ¿Cuánta demanda cuando ocurre? Modela el tamaño de las demandas no-cero usando otro SES independiente.

La predicción final combina ambos:

$$\hat{Y} = \frac{\hat{z}}{\hat{p}} \times \left(1 - \frac{\alpha}{2}\right)$$

donde:

- $\hat{z}$ = tamaño estimado de la demanda cuando ocurre (ej: si las últimas ventas no-cero fueron 2, 3, 1, 2, $\hat{z} \approx 2$).
- $\hat{p}$ = intervalo estimado entre demandas no-cero en días (ej: si las ventas ocurren cada 3-4 días en promedio, $\hat{p} \approx 3.5$).
- α = parámetro de suavizado (típicamente 0.1, da más peso al historial que a la última observación).

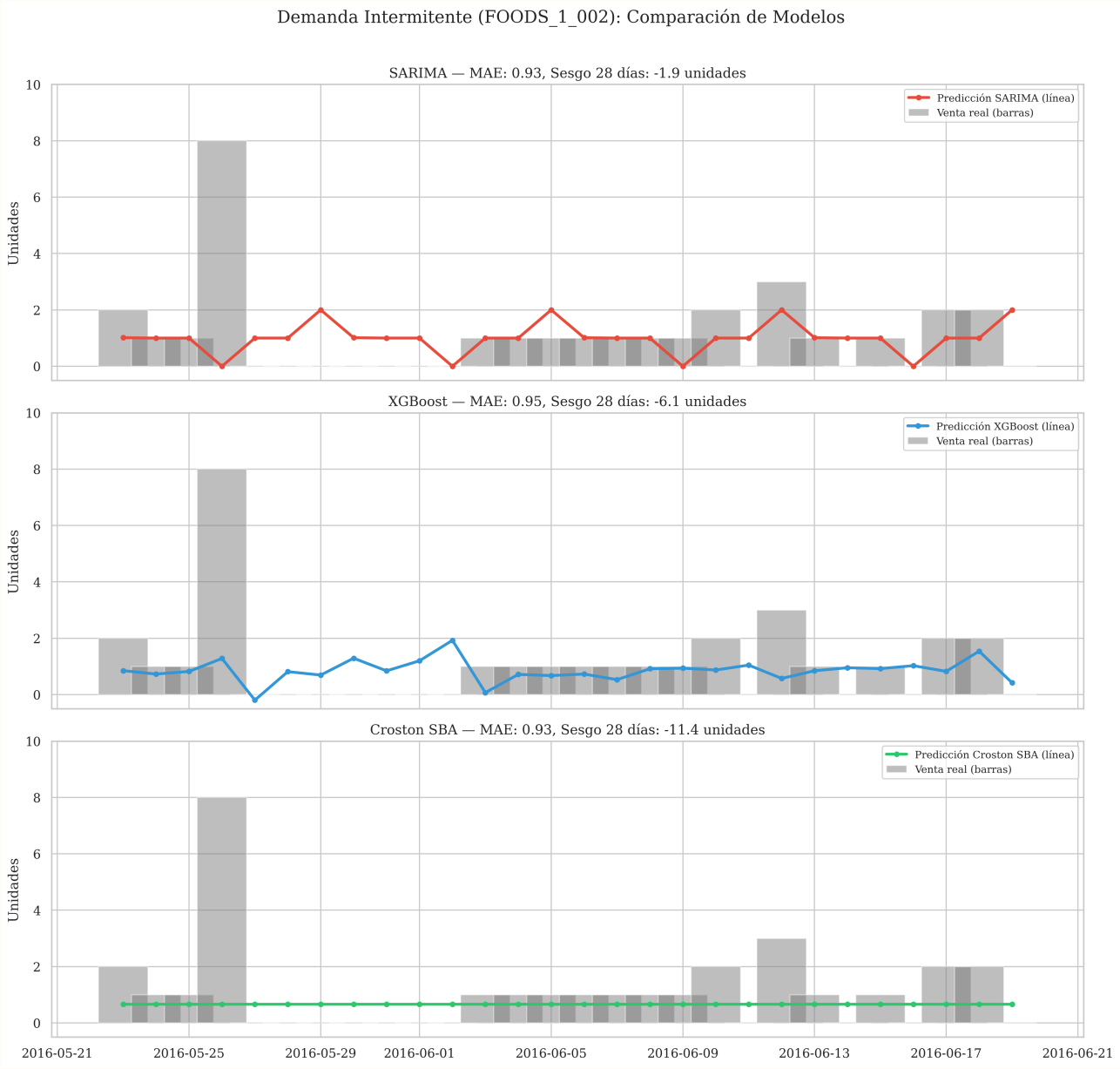


Figura 10.2: Comparación de 3 modelos en 28 días de test. Las barras grises son las ventas reales (la mayoría son cero). Las **líneas de colores con puntos** son las predicciones de cada modelo. Cada panel muestra el MAE y el sesgo acumulado (cuántas unidades de más o de menos pediríamos al proveedor).

- $(1 - \alpha/2)$ = corrección SBA (Syntetos-Boylan) que reduce un sesgo positivo conocido del método original de Croston.

Ejemplo numérico: Si estimamos que la demanda típica es de $\hat{z} = 2$ unidades y ocurre cada $\hat{p} = 3$ días, con $\alpha =$ ::

$$\hat{Y} = \frac{2}{3} \times (1 - 0.05) = 0.667 \times 0.95 = 0.63 \text{ unidades/día}$$

Esto equivale a 0.63 × 7 = 4.4 unidades por semana. Un gerente de inventario puede usar este número directamente: "necesito tener al menos 5 unidades en stock cada semana para cubrir la demanda esperada".

10.2.4 Nota sobre métricas

En el Capítulo 5 vimos que el MAPE se calcula como $\frac{|y-\hat{y}|}{y}$. Cuando $y = 0$ (que en demanda intermitente ocurre el 66% de los días), la división produce $\frac{|0-\hat{y}|}{0} = \infty$. Un solo día con venta cero hace que el MAPE promedio sea infinito, invalidando completamente la métrica.

Alternativas adecuadas para series intermitentes:

- **MAE:** Funciona siempre, pero no da contexto porcentual.
- **RMSSE (Root Mean Squared Scaled Error):** La métrica oficial de la competición M5 de Kaggle. Escala el error respecto a un baseline ingenuo, y no tiene problemas con ceros.
- **Fill Rate:** Métrica de inventario. Responde: "¿Qué porcentaje de la demanda real pudimos satisfacer con el stock que teníamos?". Es la más relevante para decisiones de negocio.

10.3 Cold Start: el producto fantasma

10.3.1 El problema

Un gerente de categoría te llama: "Acabamos de listar un producto nuevo. Lleva 7 días en la tienda. Necesito un pronóstico para las próximas 4 semanas". ¿Qué hacés?

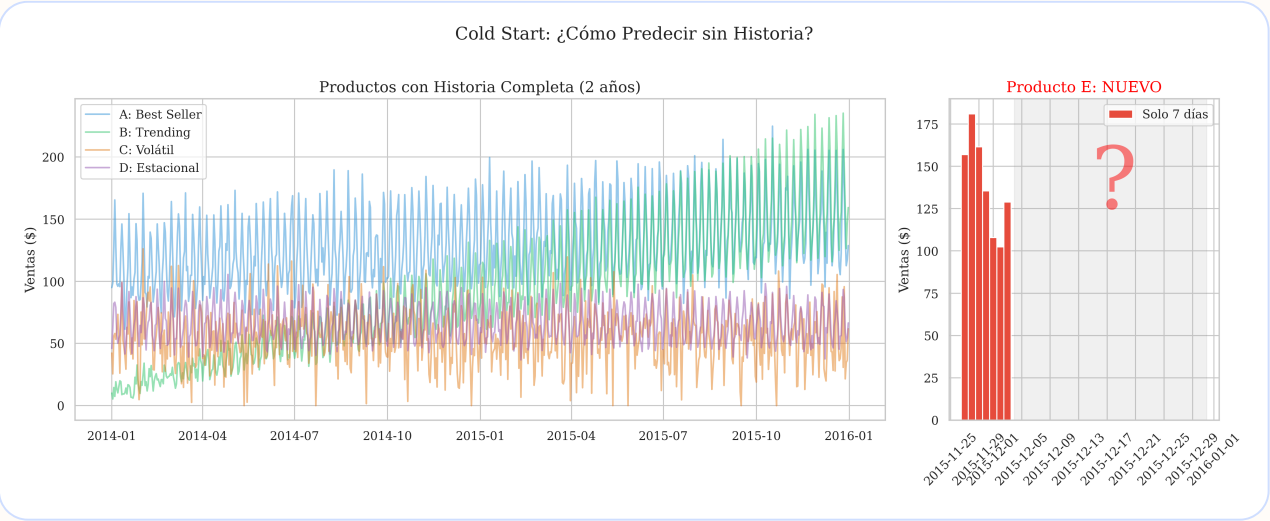


Figura 10.3: Izquierda: 4 productos con 2 años de historia. **Derecha:** El Producto E solo tiene 7 datos y un gran signo de interrogación.

Intentar un SARIMA con 7 datos no funciona (necesita al menos un ciclo estacional completo). XGBoost con 7 filas no tiene suficientes datos para calcular lags. Es un callejón sin salida.

10.3.2 Estrategia 1: transferencia por similitud

La idea es encontrar productos existentes con historial completo que se parezcan al producto nuevo, y usar su comportamiento como proxy. Hay varias formas de medir esa similitud:

- **Por metadatos:** Misma categoría, mismo rango de precio, misma marca. Amazon Forecast utiliza este enfoque y reporta mejoras de hasta 45% en precisión al seleccionar explícitamente productos similares por atributos (marca, color, tamaño, categoría).
- **Por correlación estadística:** Con los pocos datos disponibles del producto nuevo (ej: 7 días), calcular la correlación con los últimos 7 días de cada producto existente y elegir el más parecido.
- **Por similitud visual:** En retail de moda, investigaciones recientes utilizan redes neuronales (ResNet-50) para comparar imágenes de productos y encontrar artículos visualmente similares. Si un vestido nuevo se parece a otro que ya tiene historial, se transfiere su patrón de demanda.

Una vez identificado el producto "hermano", se escala su historial por el nivel de ventas del producto nuevo. Si el hermano vende en promedio 100 unidades/día y el nuevo lleva 7 días vendiendo 70 unidades/día, se multiplica todo el historial del hermano por 70/100 = 0.7.

Ventaja: Funciona con cualquier modelo posterior (podés usar el historial transferido como input de SARIMA, XGBoost, o cualquier otro). **Limitación:** Si no hay productos realmente similares en el catálogo, la transferencia puede introducir sesgos.

10.3.3 Estrategia 2: modelo global con Cross-Learning

El Modelo Global del Capítulo 7, al entrenarse con todos los productos existentes, aprende patrones universales ("los sábados se vende más", "en diciembre hay pico"). Cuando llega un producto nuevo, el modelo aplica estos patrones universales combinándolos con las pocas features disponibles (día de la semana, mes) y los escasos lags que se puedan calcular.

Ventaja: No necesita identificar explícitamente un producto "hermano"; el cross-learning es automático. **Limitación:** Si el producto nuevo tiene un comportamiento radicalmente distinto a todo lo visto (ej: un producto de un dominio completamente nuevo), el modelo aplicará patrones que no corresponden.

10.3.4 Estrategia 3: zero-shot con Modelos Fundacionales

Los modelos fundacionales (Chronos, MOIRAI) de los Capítulos 8-9 fueron pre-entrenados con miles de millones de series de múltiples dominios. Pueden recibir los 7 datos del producto nuevo como "contexto" y generar predicciones basadas en patrones universales aprendidos durante el pre-entrenamiento.

Ventaja: No requiere infraestructura propia ni datos de otros productos. **Limitación:** El rendimiento depende de si el dominio del producto nuevo estuvo representado en los datos de pre-entrenamiento (ver Capítulo 9, Crítica 3).

10.3.5 ¿Cuál elegir?

La elección depende de factores concretos del contexto:

En la práctica, muchas empresas combinan estrategias: usan transferencia por similitud como primera estimación rápida y luego refinan con un modelo global o fundacional cuando hay más datos disponibles.

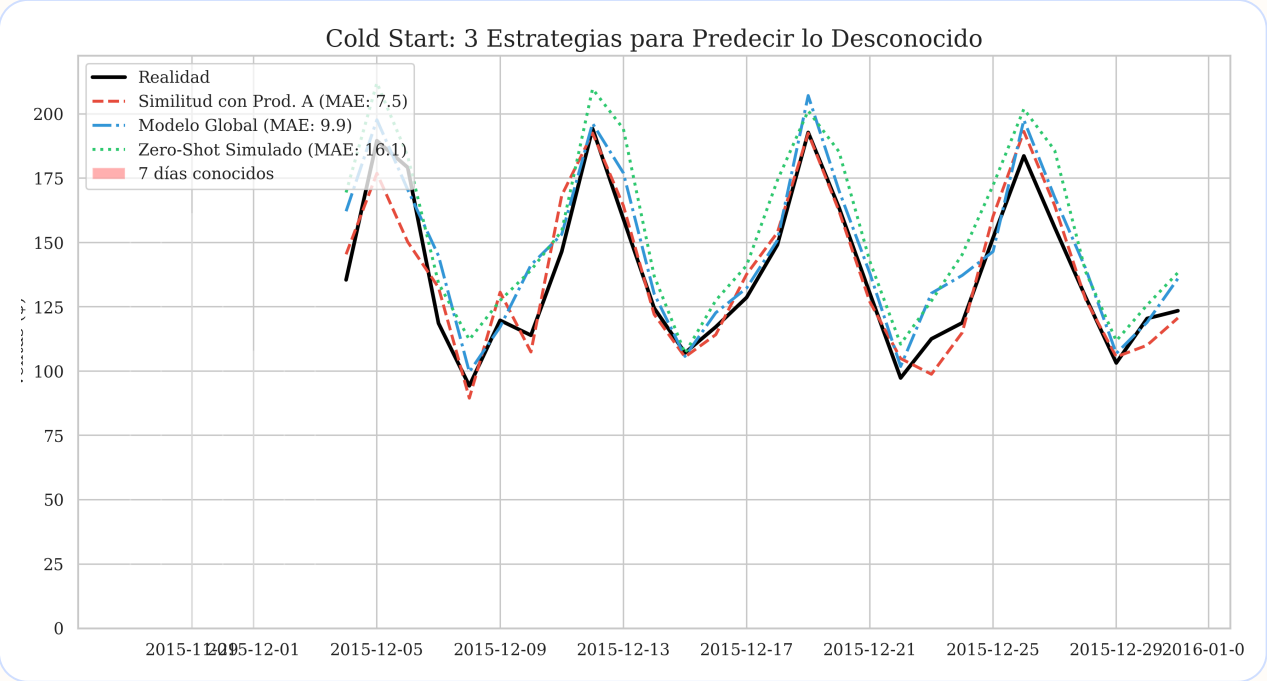


Figura 10.4: Comparación de las 3 estrategias sobre el Producto E (28 días de predicción con solo 7 días de historia). Cada estrategia aborda el problema desde un ángulo diferente.

Factor	Similitud	Modelo Global	Zero-Shot
Catálogo con productos parecidos	Ideal	Funciona bien	No necesario
Producto de un dominio nuevo	No viable	Riesgo de sesgo	Mejor opción
Infraestructura de ML existente	No requiere	Requiere modelo entrenado	Requiere API o GPU
Velocidad de implementación	Rápida (heurística)	Media (re-entrenar)	Rápida (API/descarga)
Control y explicabilidad	Alta (sabés qué hermano eligió)	Media	Baja (caja negra)

Tabla 10.1: Guía de selección para Cold Start. No existe una estrategia universalmente mejor; la elección depende de la infraestructura disponible, la composición del catálogo y los requisitos de explicabilidad.

10.4 Detección de anomalías: encontrar lo raro

10.4.1 ¿Por qué importa?

No siempre queremos predecir el futuro. A veces queremos detectar lo *anormal*: fraudes bancarios, fallas de sensores, picos inexplicables de tráfico.

Usamos datos reales del Numenta Anomaly Benchmark: viajes en taxi en NYC.

10.4.2 El contrafactual: el costo de ignorar anomalías

Inyectamos 5 anomalías artificiales (picos ×1.6 y sus días adyacentes ×1.3) en los datos de entrenamiento. Luego entrenamos dos modelos SARIMA: uno con los datos contaminados y otro con datos limpios.

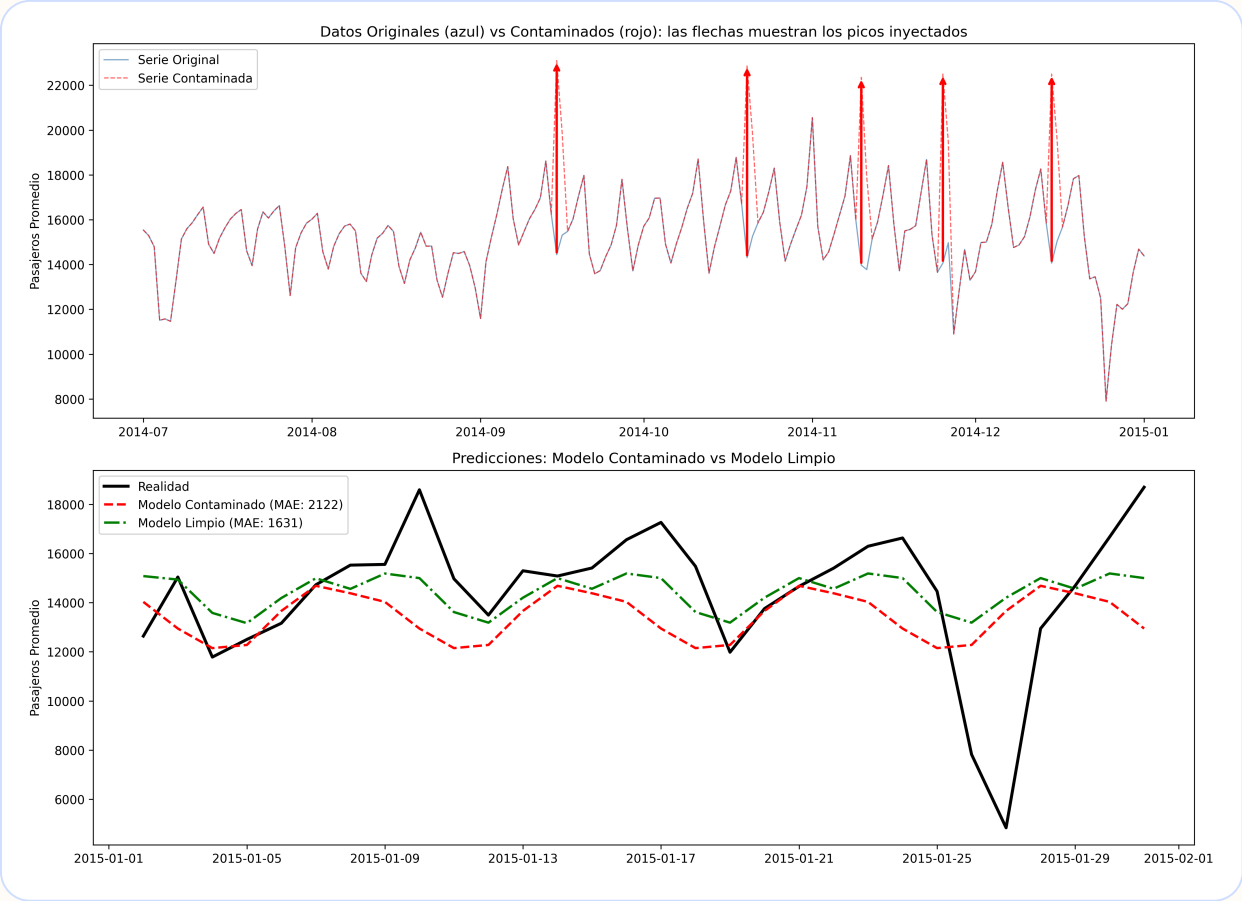


Figura 10.5: Arriba: Datos con anomalías inyectadas (puntos rojos). **Abajo:** Las predicciones del modelo contaminado se distorsionan respecto al modelo limpio.

El modelo contaminado "aprendió" los picos falsos como si fueran patrones reales, distorsionando sus predicciones futuras. **Lección: siempre revisá y limpiá tus datos antes de entrenar.**

10.4.3 Tres métodos de detección

Método 1: Z-Score sobre residuos. Entrenar un modelo, calcular residuos $(y - \hat{y})$, y marcar como anomalía si $|\text{residual}| > 3\sigma$. Es el más conservador (solo detecta anomalías muy evidentes).

Método 2: Intervalos de confianza (Cap 6). Usar los intervalos al 99% del modelo. Si un punto cae fuera, es anomalía. La ventaja es que el intervalo se adapta al nivel de la serie.

Método 3: Isolation Forest. Un algoritmo de ML que "aísla" puntos raros. No necesita un modelo temporal y puede detectar anomalías multivariadas. Es el más sensible.

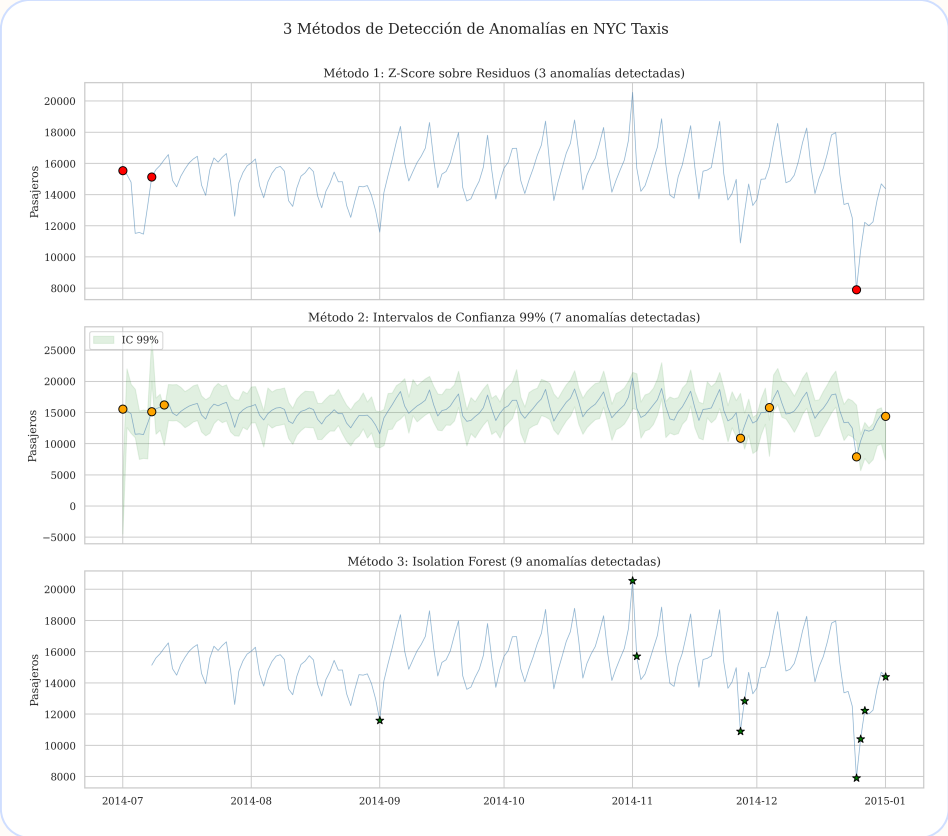


Figura 10.6: Cada panel muestra un método de detección. Z-Score (3 anomalías) es conservador. Intervalos 99% (7) es intermedio. Isolation Forest (9) es el más sensible.

10.5 Errores frecuentes en la práctica del forecasting

A lo largo de este ebook, hemos presentado técnicas y modelos. Pero en la práctica profesional, los errores más costosos no suelen ser técnicos sino metodológicos. La literatura reciente y la experiencia de la industria identifican patrones recurrentes que vale la pena documentar.

10.5.1 Errores en el manejo de datos

Usar splits aleatorios en vez de temporales. Este es probablemente el error más grave y más frecuente entre principiantes. Mezclar aleatoriamente los datos de entrenamiento y test (como se hace en clasificación) viola la estructura temporal: el modelo "ve el futuro" durante el entrenamiento. Los resultados parecen excelentes en validación pero fracasan en producción. La solución es siempre usar splits temporales: entrenar con el pasado y evaluar con el futuro, como vimos en el Capítulo 5.

No inspeccionar visualmente los datos antes de modelar. Las estadísticas resumen (media, varianza) pueden ocultar patrones que un simple gráfico revela instantáneamente: cambios de nivel, datos faltantes, outliers, cambios de frecuencia. Antes de cualquier modelo, siempre hay que graficar.

Ignorar anomalías en los datos de entrenamiento. Como demostramos en este capítulo, entrenar con datos contaminados hace que el modelo "aprenda" patrones falsos. La limpieza de datos no es opcional, es el primer paso.

10.5.2 Errores en la selección de modelos

Usar el modelo más complejo disponible por defecto. En la competición M5 de Kaggle (una de las más importantes en forecasting), métodos estadísticos relativamente simples quedaron entre los 50 mejores. Un SARIMA bien configurado puede superar a una red neuronal mal entrenada. La complejidad sin justificación no es sofisticación, es ruido.

No establecer un baseline ingenuo. Si tu modelo sofisticado no supera a la regla "mañana será igual que hoy" (naive forecast), hay un problema fundamental. Todo modelo debe compararse contra un baseline simple antes de declarar que "funciona".

Usar MAPE como métrica en series con ceros. Como vimos en la sección de demanda intermitente, un solo cero en los datos reales hace que el MAPE explote a infinito, invalidando toda la evaluación.

10.5.3 Errores en la comunicación de resultados

Reportar predicciones puntuales sin incertidumbre. Como discutimos en el Capítulo 6, una predicción sin intervalo de confianza es incompleta. El tomador de decisiones necesita saber no solo "cuánto", sino "con qué nivel de certeza".

Evaluar con un solo periodo de test. Un modelo puede tener suerte en un mes tranquilo y fallar en un mes volátil. El backtesting con múltiples folds (Capítulo 5) es la única forma honesta de evaluar el rendimiento esperado.

10.6 Reflexiones finales

Hemos recorrido un camino largo: desde descomponer una serie en sus componentes básicos (Capítulo 1) hasta enfrentar los problemas que ningún modelo resuelve automáticamente. Es un buen momento para mirar hacia atrás y consolidar lo aprendido.

El forecasting de series de tiempo combina rigor estadístico con pragmatismo ingenieril. No existe una solución universal: cada problema tiene sus particularidades (frecuencia de datos, presencia de ceros, disponibilidad de variables externas, cantidad de series) que determinan qué herramienta es la más adecuada.

Los modelos clásicos como ARIMA siguen siendo relevantes por su interpretabilidad y eficiencia en problemas univariados bien definidos. Los modelos de Machine Learning (XGBoost, LightGBM, CatBoost) dominan cuando hay múltiples fuentes de información, variables exógenas y patrones no lineales. Los Modelos Globales escalan a miles de series compartiendo conocimiento entre ellas. Y los Modelos Fundacionales abren la puerta a predicciones sin entrenamiento específico, aunque con limitaciones que aún se están investigando.

Lo que permanece constante, independientemente de la herramienta elegida, son los principios fundamentales: entender los datos antes de modelarlos, validar con rigor para no autoengañarse, cuantificar la incertidumbre para tomar mejores decisiones, y mantener una actitud crítica ante cualquier resultado que parezca demasiado bueno.

El campo evoluciona a una velocidad sin precedentes: entre 2023 y 2026, hemos visto dos generaciones completas de modelos fundacionales, con mejoras drásticas en eficiencia (Moirai 2.0 es 96% más pequeño que su predecesor) y capacidades (Chronos-2 ya soporta covariables). Es probable que para cuando este texto sea leído, existan nuevos modelos que superen a los aquí descritos. Pero los fundamentos (estacionariedad, validación temporal, feature engineering, cuantificación de incertidumbre) seguirán siendo la base sobre la que se construye cualquier avance futuro.

Próximos pasos

Esta guía cubre los fundamentos y las herramientas más utilizadas, pero el campo tiene capas de complejidad que van más allá de lo que abordamos aquí. Cuatro direcciones naturales para seguir creciendo:

- **Forecasting jerárquico y reconciliación:** En la práctica, rara vez se pronostica una sola serie aislada. Una empresa necesita pronósticos coherentes a nivel de SKU, categoría, tienda y región, y que la suma de los pronósticos individuales coincida con el total. Las técnicas de reconciliación (bottom-up, top-down, optimal reconciliation) resuelven este problema y son esenciales en cualquier sistema de forecasting a escala.
- **Inferencia causal en series de tiempo:** Pronosticar responde "¿qué va a pasar?", pero muchas decisiones de negocio requieren responder "¿qué pasaría si cambio X?". El análisis de intervención y el razonamiento contrafactual permiten estimar el impacto de una promoción, un cambio de precio o un evento externo, separándolo del comportamiento orgánico de la serie.
- **Modelos adaptativos y aprendizaje online:** Los modelos que presentamos se reentrenan periódicamente con lotes de datos. Pero en sistemas de alta frecuencia (pricing dinámico, detección de fraude en tiempo real), se necesitan modelos que se actualicen incrementalmente con cada nueva observación sin reentrenar desde cero.
- **Optimización multi-objetivo:** En producción, minimizar el MAE no es el único objetivo. Un sistema de forecasting real balancea precisión, costo de inventario, nivel de servicio al cliente y costo computacional. Esta intersección entre forecasting y operations research es donde los modelos se convierten en decisiones de negocio.

Bibliografía

Referencias generales

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). OTexts. `https://otexts.com/fpp3/`

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control* (5th ed.). Wiley.

Capítulo 1, la anatomía de una serie de tiempo

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics*, 6(1), 3–73.

Box & Jenkins (2015), Serie G, fuente del dataset `AirPassengers` dataset (monthly international airline passengers, 1949–1960).

Capítulo 2, estacionariedad y análisis exploratorio

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(366), 427–431.

Said, S. E., & Dickey, D. A. (1984). Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3), 599–607. (Augmented Dickey-Fuller.)

Capítulo 3, herramientas clásicas: ARIMA

Ljung, G. M., & Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, 65(2), 297–303.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic Time Series Forecasting: The `forecast` Package for R. *Journal of Statistical Software*, 27(3), 1–22. (Basis of `auto_arima`.)

Capítulo 4, series de tiempo como regresión supervisada

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of KDD '16*, 785–794.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, 30.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31.

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, 30. (SHAP.)

Ben Taieb, S., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, 39(8), 7067–7083. (Recursive, direct and hybrid strategies.)

Capítulo 5, validación y métricas de error

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting*, 16(4), 437–450.

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191, 192–213.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. (MASE.)

Capítulo 6, navegando la incertidumbre

Vovk, V., Gammerman, A., & Shafer, G. (2005). *Algorithmic Learning in a Random World*. Springer.

Angelopoulos, A. N., & Bates, S. (2023). Conformal Prediction: A Gentle Introduction. *Foundations and Trends in Machine Learning*, 16(4), 494–591.

Stankevičiūtė, K., Alaa, A. M., & van der Schaar, M. (2021). Conformal Time-Series Forecasting. *Advances in Neural Information Processing Systems*, 34.

Capítulo 7, modelos globales

Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4), 1632–1653.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191.

Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, 140, 112896.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364.

Capítulos 8 y 9, transformers y modelos fundacionales

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30.

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *ICLR 2023*. (Patching.)

Garza, A., & Mergenthaler-Canseco, M. (2023). TimeGPT-1. arXiv:2310.03589.

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., et al. (2024). Chronos: Learning the Language of Time Series. arXiv:2403.07815.

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., & Sahoo, D. (2024). Unified Training of Universal Time Series Forecasting Transformers. *ICML 2024*. (MOIRAI and the LOTSA archive.)

Das, A., Kong, W., Sen, R., & Zhou, Y. (2024). A decoder-only foundation model for time-series forecasting. *ICML 2024*. (TimesFM.)

Liu, Y., Qin, G., Shi, X., Hu, Z., Wang, J., & Long, M. (2025). Sundial: A Family of Highly Capable Time Series Foundation Models. arXiv:2502.00816.

Aksu, T., Woo, G., Liu, J., Liu, X., Liu, C., Savarese, S., Xiong, C., & Sahoo, D. (2024). GIFT-Eval: A Benchmark for General Time Series Forecasting Model Evaluation. arXiv:2410.10393.

Amazon Web Services (2025). *Chronos-2* model card y reporte técnico. Hugging Face: `amazon/chronos-2`.

Salesforce AI Research (2025). *Moirai 2.0* model card. Hugging Face: `Salesforce/moirai-2.0-R-small`.

Capítulo 10, desafíos del mundo real

Croston, J. D. (1972). Forecasting and Stock Control for Intermittent Demands. *Operational Research Quarterly*, 23(3), 289–303.

Syntetos, A. A., & Boylan, J. E. (2005). The accuracy of intermittent demand estimates. *International Journal of Forecasting*, 21(2), 303–314. (Croston-SBA.)

Kourentzes, N. (2014). On intermittent demand model optimisation and selection. *International Journal of Production Economics*, 156, 180–190.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *Proceedings of ICDM 2008*, 413–422.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *CVPR 2016*, 770–778. (ResNet-50, used for visual similarity transfer.)