GRADO EN CIENCIA DE DATOS

Análisis Predictivo de Series Temporales en el Entorno Retail

Presentado por:
Álvaro Tébar Folch
Dirigido por:
Vicente Castillo

CURSO ACADÉMICO 2025-2026

AGRADECIMIENTOS
A mi familia

RESUMEN

La gestión masiva de datos y la precisión predictiva son desafíos críticos para la cadena de suministro en el sector retail, donde las fluctuaciones estocásticas de la demanda generan roturas de stock y mermas logísticas. El presente trabajo realiza un análisis comparativo de modelos de predicción de demanda aplicado al dataset M5 Forecasting – Accuracy de Walmart, compuesto por más de 58 millones de registros y 30.490 series temporales jerárquicas (Makridakis, Spiliotis & Assimakopoulos, 2022). Se evalúan cinco modelos de naturaleza heterogénea: un baseline ingenuo estacional, dos modelos estadísticos clásicos (ARIMA y SARIMA), un modelo aditivo descomponible (Prophet) y un algoritmo de Gradient Boosting (XGBoost) sobre un horizonte de validación de 28 días, empleando el RMSE como métrica principal y la función de pérdida Tweedie para manejar la intermitencia del 68,2 % de ceros del dataset. Los resultados sitúan a XGBoost como el modelo más adecuado a escala granular, con una reducción del error del 56,87 % respecto al baseline a nivel SKU. El trabajo se vincula con el ODS 12 al orientar la gestión de inventarios hacia una mayor eficiencia logística y la reducción del desperdicio alimentario.

Palabras clave: predicción de demanda, series temporales, XGBoost, dataset M5, ODS 12.

ABSTRACT
The massive management of data and predictive accuracy are critical challenges for the retail supply chain, where stochastic demand fluctuations generate stockouts and logistical waste. This work presents a comparative analysis of demand forecasting models applied to the M5 Forecasting – Accuracy dataset from Walmart, comprising over 58 million records and 30,490 hierarchical time series (Makridakis, Spiliotis & Assimakopoulos, 2022). Five models of heterogeneous nature are evaluated: a seasonal naïve baseline, two classical statistical models (ARIMA and SARIMA), a decomposable additive model (Prophet) and a Gradient Boosting algorithm (XGBoost) over a 28-day validation horizon, using RMSE as the main metric and the Tweedie loss function to handle the dataset's 68.2 % zero inflation. Results identify XGBoost as the most suitable approach at the granular scale, achieving a 56.87 % error reduction over the baseline at SKU level. The work is aligned with SDG 12 by orienting inventory management towards greater logistical efficiency and the reduction of food waste.
Keywords: demand forecasting, time series, XGBoost, M5 dataset, SDG 12.

ÍNDICE

1. Introducción 9
1.1 Contexto y Relevancia del Sector 9
1.2 Justificación y Objetivos 9
1.3 Motivación Personal 10
2. Objetivos 10
2.1 Objetivo Principal 10
3. Marco Teórico 11
3.1 Series Temporales 11
3.1.1 Componentes de series temporales 11
3.2 Modelos de descomposición de series temporales 13
3.3 Series Temporales Jerárquicas 14
3.3.1 El reto de la coherencia entre niveles 16
3.4 Modelos Estadísticos Clásicos 17
3.4.1 Modelos base de referencia 18
3.4.2 Media histórica 18
3.4.3 Método ingenuo (Naive) 19
3.4.4 Predicción Naive estacional 19
3.5 Modelos ARMA Y ARIMA 20
3.5.1 ARMA (autorregresivo media móvil) 21
3.5.2 ARIMA (autorregresivo integrado con media móvil) 22
3.5.3 De ARIMA a SARIMA: la necesidad de capturar la estacionalidad 23
3.6 SARIMA (estacional autorregresivo integrado con media móvil) 23
3.7 Modelo Algorítmico Avanzado 25
3.7.1 Árboles de decisión 25
3.7.2 Ensamblado de árboles y Gradient Boosting 27
3.7.3 Gradient Boosting 28
3.7.4 XGBoost 29
3.7.5 Prophet 31
3.7.6 Redes Neuronales y Aprendizaje Profundo 34
4. Metodología 34
4.1 Conjunto de datos: (M5 Forecasting Competition - Accuracy) 34
4.2 Restricciones Computacionales Y Optimización de Memoria 35
4.3 Métricas de Evaluación 37
4.3.1 Métrica seleccionada: RMSE (Root Mean Squared Error) 37
4.3.2 Descarte del MAPE (Mean Absolute Percentage Error) 38
4.3.3 Métrica oficial de la competición M5: WRMSSE 38

4.4 Entorno Tecnológico 39
5. Desarrollo y Resultados 41
5.1 Análisis Exploratorio de Datos (EDA) 41
5.2 Preparación de los datos para cada modelo 46
5.3 Resultados de los Modelos Base 47
5.3.1 Aplicación Naive Estacional 48
5.3.2 Aplicación modelo ARIMA 49
5.3.3 Aplicación modelo SARIMA 50
5.4 Aplicación Modelos Prophet 52
5.5 Aplicación Modelo XGBoost 55
5.5.1 Construcción del dataset maestro e ingeniería de características 55
5.5.2 Abordaje de la Intermitencia: Función de Pérdida Tweedie 57
5.5.3 Partición temporal y prevención de la data leakage 58
5.5.4 Resultados sobre la serie agregada de CA_1 59
5.5.5 Escalado al dataset completo 60
5.6 Contraste de Resultados 61
5.6.1 Marco interpretativo: la comparación entre escalas 61
5.6.2 Resultados por grupo de comparación 62
5.6.3 El hallazgo metodológico central 64
5.6.4 Limitación de la comparativa y trabajo futuro 65
6. Objetivos de desarrollo sostenible (ODS) 66
6.1 Impacto en el ODS 12: Producción y Consumo Responsables 66
6.1.1 El problema del desperdicio en la cadena de suministro retail 66
6.1.2 La conexión cuantitativa con los resultados del trabajo 66
6.1.3 Sostenibilidad económica y ambiental como objetivos complementarios 67
7. Conclusión 68
7.1 En relación con el objetivo principal 68
7.2 En relación con los modelos estadísticos clásicos 68
7.3 En relación con el proceso metodológico 69
7.4 Limitaciones del trabajo 69
7.5 Líneas de investigación futura 70
7.6 Reflexión final 71
8. Anexo Disponibilidad de Datos y Código 71
9. Referencias 71

ÍNDICE DE FIGURAS

Figura 1: Ejemplos de componentes de series temporales 12
Figura 2: Comparación Serie de alta frecuencia y baja frecuencia (Mauricio, 2007) 13
Figura 3 Serie Aditiva Viñals (2009) 14
Figura 4 Serie Multiplicativa Viñals (2009) 14
Figura 5: Diagrama de la jerarquía M5 con sus 12 niveles, equivalente a la Figura 1 del Adaptado de The M5 Competition: Competitors' Guide (p. 3), por University of Nicosia (UNIC, 2020). 16
Figura 6: Predicción de producción trimestral de cerveza australiana (Hyndman & Athanasopoulos, 2018) 20
Figura 7: Ejemplo de árbol de decisión CART aplicado a la demanda retail - Fuente: Elaboración propia. 27
Figura 8: Evolución histórica del volumen de ventas a nivel corporativo 41
Figura 9: Resultado de la descomposición STL: paneles de tendencia, estacionalidad (m=7) y residuo. 43
Figura 10: Componente estacional STL: detalle de tres semanas consecutivas con etiquetado por día de la semana 43
Figura 11: Venta media por SKU y día de la semana. 44
Figura 12: Distribución de ventas diarias a nivel SKU: frecuencia relativa para valores 0, 1, 2, 3, 4, ≥5 45
Figura 13: Evaluación comparativa de los modelos de predicción. Panel superior izquierdo: RMSE por modelo sobre la serie agregada de la tienda CA_1. Panel superior derecho: RMSE por modelo sobre el dataset completo agregado a nivel corporativo. Panel inferior: comparativa diaria entre las ventas reales observadas (línea negra) y las predicciones de los tres modelos directamente comparables sobre la serie corporativa agregada durante los 28 días del horizonte de evaluación. Fuente: elaboración propia a partir de la ejecución del notebook adjunto. 64

ÍNDICE DE TABLAS

TABLA 1: ESTRUCTURA JERÁRQUICA DEL DATASET M5 – NIVELES DE AGREGACIÓN Y NÚMERO DE SERIES. ELABORACIÓN PROPIA A PARTIR DE MAKRIDAKIS ET AL. (2022). 15
TABLA 2: REDUCCIÓN DE MEMORIA POR FICHERO TRAS LA APLICACIÓN COMBINADA DEL DOWNCASTING NUMÉRICO Y LA CONVERSIÓN AL TIPO CATEGORY. DATOS OBTENIDOS DE LA SECCIÓN «OPTIMIZACIÓN DE MEMORIA» DEL NOTEBOOK ADJUNTO. 37
TABLA 3: COMPARATIVA DE MODELOS EN EL NIVEL AGREGADO CORPORATIVO. FUENTE: ELABORACIÓN PROPIA A PARTIR DE LA EJECUCIÓN DEL NOTEBOOK ADJUNTO. 62
TABLA 4: COMPARACIÓN DE LOS CINCO MODELOS SOBRE LA SERIE AGREGADA DE LA TIENDA CA_1 EN EL HORIZONTE DE 28 DÍAS. MEDIA DE VENTAS DIARIAS EN EL HORIZONTE DE EVALUACIÓN: 4.787 UNIDADES. FUENTE: ELABORACIÓN PROPIA A PARTIR DE LA EJECUCIÓN DEL NOTEBOOK ADJUNTO. . 63

1. Introducción

1.1 Contexto y Relevancia del Sector

La gestión de la cadena de suministro en grandes superficies de retail se enfrenta actualmente a un desafío de escalabilidad sin precedentes. La precisión en el pronóstico de la demanda a nivel de unidad de mantenimiento de stock SKU (stock keeping unit) es el eje que determina la rentabilidad operativa, dado que decisiones críticas como la fijación de precios, la gestión del inventario y la disponibilidad del producto dependen directamente de su pronóstico. Utilizando como referencia (benchmark) el dataset M5 Forecasting – Accuracy, un conjunto de datos público de ventas minoristas (de Walmart) usado para predicción de demanda, este trabajo aborda la complejidad de modelar 58.3 millones de registros repartidos en series jerárquicas, donde factores como la intermitencia de las ventas, la estacionalidad cruzada y las promociones limitan la aplicabilidad directa de los métodos estadísticos tradicionales (Fildes et al., 2022; Syntetos & Boylan, 2005). La incapacidad de los sistemas de gestión actuales para determinar con precisión la demanda futura inmediata de cada producto (Fildes et al., 2022), produce una incertidumbre crítica a nivel de unidad. Aunque la automatización permite el movimiento rápido de mercancías (conocida como autostores en el sector logístico), el sistema opera con un conocimiento limitado de la demanda futura, lo que deriva en un desajuste constante: la aparición de roturas de stock (falta de disponibilidad de producto) en periodos de alta demanda, o el sobreabastecimiento de productos frescos, que genera mermas innecesarias (Fildes et al., 2022; Silver et al., 2016). Como señalan Eriksson et al. (2014), la falta de optimización en la previsión es una de las causas principales del desperdicio alimentario y la pérdida de margen en la cadena de suministro moderna (ODS 12).

El núcleo de esta investigación es la comparación de modelos de análisis de series temporales (ARIMA, SARIMA, Prophet, XGBoost) aplicados al dataset M5 Forecasting – Accuracy. Cabe señalar que la naturaleza jerárquica de los datos (múltiples niveles de agregación: SKU, categoría, tienda, estado y total), y las técnicas de optimización de memoria como el downcasting (reducción del tipo de dato para ahorrar memoria) constituyen retos técnicos de gran envergadura (cada uno suficientemente ambicioso como para centrar un TFG independiente), por lo que quedan fuera del alcance de este trabajo. El foco se pone directamente en comparar el rendimiento predictivo de los modelos mencionados sobre dicho dataset.

1.2 Justificación y Objetivos

Ante este desafío, la presente investigación propone la ejecución de un estudio comparativo entre diversos enfoques matemáticos y algorítmicos para dotar de capacidad proactiva sobre la gestión de inventarios.

La metodología comprende la evaluación de modelos estadísticos, como ARIMA y SARIMA, frente a algoritmos de aprendizaje automático (machine learning) como Prophet y XGBoost, con el fin de determinar la solución con menor error predictivo. Al resolver la incertidumbre sobre la cantidad de producto requerida, se busca sincronizar el volumen de ventas con la demanda real, garantizando la disponibilidad para el cliente

(Fildes et al., 2022; Silver et al., 2016). Asegurando la sostenibilidad económica y ambiental mediante un consumo y producción responsable (ODS 12).

1.3 Motivación Personal

El interés por desarrollar este proyecto nace de la observación directa de ineficiencias operativas en el ámbito minorista, concretamente de los desajustes entre la oferta disponible y la demanda real que generan tanto roturas de stock como sobreabastecimiento de productos perecederos. La realización de prácticas curriculares en una empresa logística especializada en el sector retail ha permitido contextualizar este problema en un entorno operativo real, identificando la ausencia de herramientas predictivas precisas como una de sus causas principales. De este modo, el presente trabajo aplica conocimientos de ciencia de datos y análisis predictivo para aportar una solución técnica fundamentada a un problema de alta relevancia práctica, contribuyendo a transformar una debilidad estructural del servicio en una ventaja competitiva y en una mejora medioambiental alineada con el ODS 12 (principalmente).

2. Objetivos

2.1 Objetivo Principal

El objetivo principal de este trabajo es comparar el rendimiento predictivo de distintos modelos de análisis de series temporales mediante modelos estadísticos y algorítmicos aplicados al dataset M5 Forecasting – Accuracy, con el fin de determinar qué enfoque ofrece menor error de predicción en el contexto del retail. Para ello, en esta investigación se perseguirán los siguientes objetivos específicos:

O1.1 – Caracterizar el dataset y analizar la naturaleza estadística de los datos: mediante un análisis exploratorio exhaustivo del conjunto de datos mencionado, identificando sus propiedades estadísticas fundamentales (intermitencia de la demanda, estacionalidad semanal y estructura jerárquica) con el fin de justificar las decisiones metodológicas adoptadas en las fases posteriores.

O1.2 – Establecer una línea base de referencia (Baseline): implementar el modelo Naive Estacional como benchmark mínimo de calidad, calculando su error sobre el conjunto completo de 30.490 SKUs para fijar el listón que cualquier modelo más complejo debe superar.

O1.3 – Evaluar los modelos estadísticos clásicos: aplicar los modelos ARIMA/SARIMA y Prophet sobre la serie temporal agregada de la tienda CA_1, evaluando su capacidad predictiva en el nivel de agregación para el que fueron diseñados y documentando empíricamente su inviabilidad a escala completa del dataset M5.

O1.4 – Diseñar e implementar el modelo XGBoost: construir un modelo global de Gradient Boosting con ingeniería de características temporales (lags semanales, medias móviles y variables exógenas de precio, calendario y SNAP) y función de pérdida Tweedie, justificando cada decisión de diseño a partir de los hallazgos del análisis exploratorio.

O1.5 – Comparar el rendimiento predictivo de los modelos: contrastar los resultados obtenidos mediante el RMSE calculado en escala comparable, evaluando la mejora de XGBoost respecto al Baseline Naive y discutiendo el WRMSSE como métrica oficial de la competición M5 y línea de trabajo futuro.

O1.6 – Evaluar el impacto en el ODS 12: analizar cómo la mejora en la precisión predictiva contribuye a la reducción del desperdicio alimentario y a la optimización logística en el sector retail, en el marco del Objetivo de Desarrollo Sostenible número 12 de la Agenda 2030.

3. Marco Teórico

3.1 Series Temporales

Las series temporales comprenden una serie de datos ordenados en el tiempo. Cada dato se asigna a una marca de tiempo específica en orden cronológico y en intervalos de tiempo regulares (t = 1, 2, 3, ... n). Se trata de un proceso donde los datos presentan una estructura de dependencia interna; es decir, el valor de la variable en el instante *t* está condicionado, en mayor o menor grado, por sus realizaciones previas t − 1, t − 2, ...

En este proyecto, se analiza el volumen de ventas del dataset (M5 Forecasting Competition - Accuracy), que abarca datos desde 2011 hasta 2016. Como se ilustra en la Figura 2, la demanda agregada de las diez tiendas distribuidas en los estados de California, Texas y Wisconsin presenta una variabilidad marcada por la interacción de múltiples covariables exógenas que son aquellas que están afectadas por valores externos en el modelo al que se incluye. Factores como la política de precios, eventos especiales (festivos, eventos deportivos) y calendarios promocionales (rebajas) generan un espacio con un gran número de variables explicativas (*features).* Variables que permiten desarrollar modelos con mayor capacidad predictiva, pero a su vez son modelos más complejos y con mayor riesgo de sobreajuste (*overfitting*) que exigen una selección y regularización de modelos robustos.

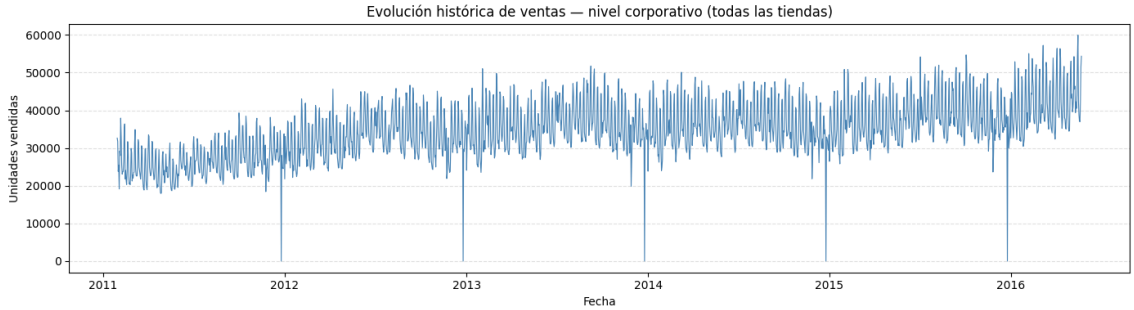

Figura 1: Ejemplo serie temporal evolución histórica - M5 Forecasting – Accuracy.

3.1.1 Componentes de series temporales

Distinguimos series de alta y baja frecuencia, dependiendo del horizonte temporal en el que se ordenan los datos.

En una serie temporal de baja frecuencia, es decir, con datos mensuales o trimestrales, las características habituales son: tendencia y estacionalidad. Además, en muchos casos la varianza aumenta con el nivel de la serie.

Para el análisis de estos componentes, resulta fundamental determinar la estacionariedad de la serie. Una serie se define como estacionaria cuando sus propiedades estadísticas, como la media y la varianza, permanecen constantes a lo largo del tiempo (Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M., 2015). En el

contexto del retail, la presencia de tendencia (T) o estacionalidad (E) rompe esta condición de estacionariedad, obligando a los modelos a aplicar transformaciones como la diferenciación (restar a cada dato el dato anterior) para estabilizar los datos antes de la predicción.

- Tendencia (T): Tendencia es trend, definición general de los datos a lo largo del tiempo, como aumento, disminución o constante.

  Ejemplo: Ingresos de una Empresa emergente.
- Estacionalidad (E): Estacionalidad es seasonal, patrones de datos que se repiten a lo largo de un conjunto de períodos de tiempo como diario, mensual o anual.

  Ejemplo: demanda de electricidad los meses de verano.
- Ciclo (C): Oscilaciones de larga duración, generalmente ligadas a ciclos económicos, que no tienen un periodo fijo como la estacionalidad.

  Ejemplo: ciclos del mercado inmobiliario.
- Residuo (ruido blanco): Por definición comprende todo lo que no quede explicado por la tendencia o estacionalidad. Representa cambios rápidos en la serie que no resultan explicados por las componentes anteriores. Idealmente estos cambios son completamente aleatorios.

  Ejemplo: Lecturas de peso (kg): [70.0, 70.3, 69.8, 70.1]; (ruido ≈ ±0.3 kg por variaciones de báscula).

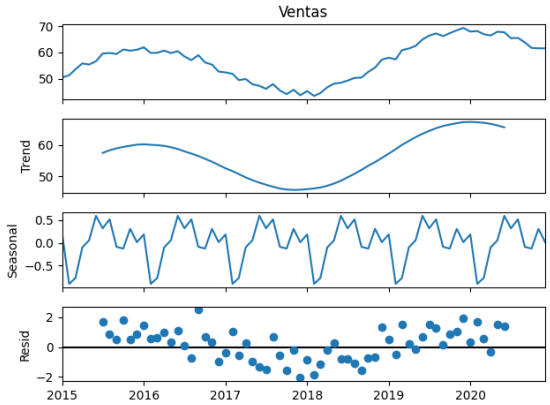

Figura 1: Ejemplos de componentes de series temporales

Las series temporales de alta frecuencia con datos ordenados en segundos, minutos, horarios, diarios… suelen tener patrones intradías o semanales y cambios de nivel de la serie, por lo que la media y la varianza pueden variar con el tiempo (Athanasopoulos H. &., 2021).

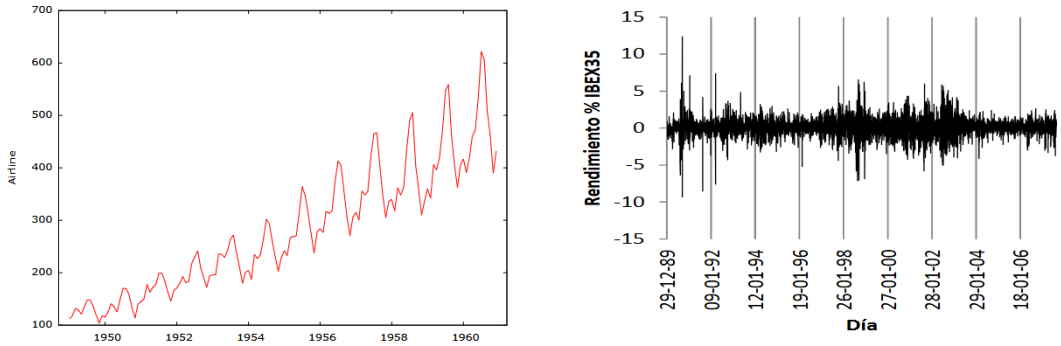

Figura 2: Comparación Serie de alta frecuencia y baja frecuencia (Mauricio, 2007)

La Figura 3, comprende un ejemplo de cada una de estas series temporales. En ejemplos reales las series de baja frecuencia se utilizan para análisis macroeconómicos y planificación a largo plazo en cambio las series de alta frecuencia son críticas para el trading financiero y el monitoreo operativo en tiempo real por ende más difícil de modelar.

En conclusión, las series temporales se definen como estructuras donde la componente aleatoria o residuo (r) impide una predicción determinista exacta. Esta naturaleza implica que nos enfrentamos a un proceso estocástico. Por lo tanto, el interés científico de este análisis no reside en la descripción de los datos pasados, sino en la modelización de la incertidumbre futura. Como bien apunta la literatura académica (Viñals, 2009), si la demanda de un producto fuera un fenómeno puramente determinista (donde el resultado futuro fuera una función exacta y conocida del tiempo), la predicción carecería de valor intelectual y técnico, reduciéndose a un simple cálculo matemático. Siendo conscientes que la predicción de futuros valores no puede ser exacta los modelos deben ser capaces de interpretar las componentes de la serie temporal y replicar la estructura de la mejor manera posible.

3.2 Modelos de descomposición de series temporales

Una vez hemos tratado las características comunes de una serie temporal debemos tener en cuenta como extraer esas cualidades dependiendo de cómo interactúen los componentes de la serie.

1. Modelo Aditivo: se puede considerar que, si el patrón estacional se mantiene con amplitud constante independientemente del nivel de la tendencia se tratará de modelo aditivo (Viñals, 2009).

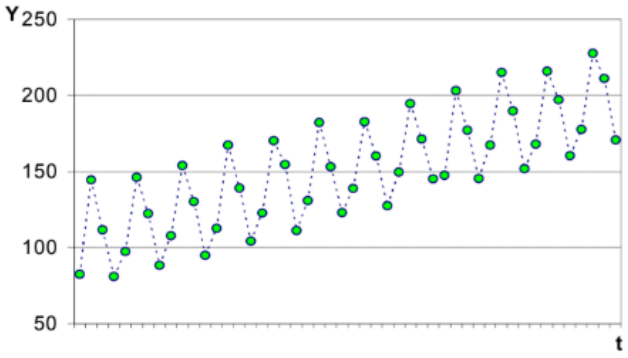

Figura 3 Serie Aditiva Viñals (2009)

Si la amplitud de los "picos" y "valles" no cambia mucho a medida que los datos crecen o decrecen, este es un modelo aditivo.

2. Modelo Multiplicativo: las variaciones estacionales cambian proporcionalmente al nivel de la serie, es decir, si la tendencia va subiendo de nivel la amplitud de la estacionalidad también aumenta proporcionalmente figura 5.

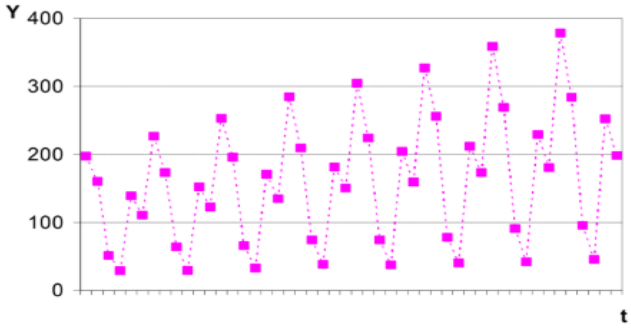

Figura 4 Serie Multiplicativa Viñals (2009)

3.3 Series Temporales Jerárquicas

En numerosos contextos reales, las series temporales no se presentan de forma aislada, sino organizadas en estructuras jerárquicas. Una serie temporal jerárquica es aquella en la que los datos pueden descomponerse de forma coherente en distintos niveles de agregación, de modo que la suma de los valores de los niveles inferiores reproduce siempre el valor del nivel superior (Hyndman & Athanasopoulos, 2018).

El sector retail constituye uno de los ejemplos paradigmáticos de esta estructura. Una cadena de supermercados puede organizar sus datos de ventas en múltiples niveles: la venta de un producto concreto en una tienda concreta (nivel más granular) puede agregarse por departamento, por categoría, por tienda completa, por estado y, finalmente, por el total nacional (nivel más agregado). Cada uno de estos niveles representa una serie temporal distinta, y todas ellas están matemáticamente vinculadas por la condición de coherencia: las predicciones realizadas en niveles inferiores, sumadas, deben coincidir con las predicciones del nivel superior correspondiente (Wickramasuriya et al., 2019).

El dataset (M5 Forecasting Competition - Accuracy) es un ejemplo paradigmático de esta estructura.

Los organizadores de la competición definen formalmente 12 niveles de agregación, que generan un total de 42.840 series temporales distintas a partir de los datos brutos de ventas (Makridakis, 2022). Esta estructura jerárquica abarca desde el nivel más granular (las

ventas de un artículo concreto (SKU) en una tienda concreta, con 30.490 series) hasta el nivel más agregado (las ventas totales de toda la cadena a nivel nacional, representadas por una única serie), pasando por niveles intermedios como categoría, departamento, tienda y estado.

Nivel	Descripción del nivel de agregación	Nº de series
1	Ventas unitarias de todos los productos, agregadas para todas las tiendas/estados	1
2	Ventas unitarias de todos los productos, agregadas por estado	3
3	Ventas unitarias de todos los productos, agregadas por tienda	10
4	Ventas unitarias de todos los productos, agregadas por categoría	3
5	Ventas unitarias de todos los productos, agregadas por departamento	7
6	Ventas unitarias de todos los productos, agregadas por estado y categoría	9
7	Ventas unitarias de todos los productos, agregadas por estado y departamento	21
8	Ventas unitarias de todos los productos, agregadas por tienda y categoría	30
9	Ventas unitarias de todos los productos, agregadas por tienda y departamento	70
10	Ventas unitarias del producto x, agregadas para todas las tiendas/estados	3.049
11	Ventas unitarias del producto x, agregadas por estado	9.147
12	Ventas unitarias del producto x, agregadas por tienda	30.490
	Total	**42.840**

Tabla 1: Estructura jerárquica del dataset M5 – Niveles de agregación y número de series. Elaboración propia a partir de Makridakis et al. (2022).

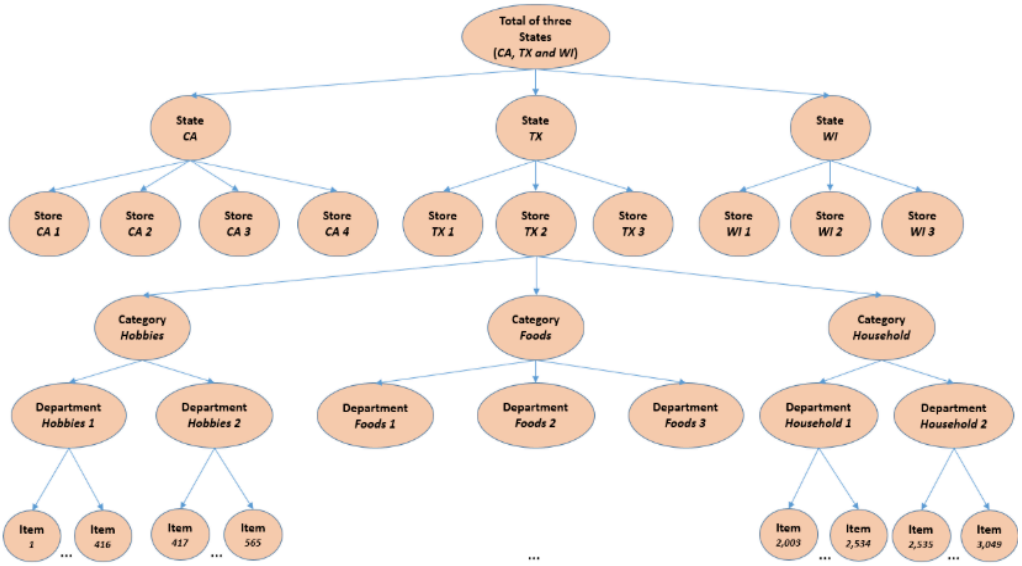

Figura 5: Diagrama de la jerarquía M5 con sus 12 niveles, equivalente a la Figura 1 del Adaptado de The M5 Competition: Competitors' Guide (p. 3), por University of Nicosia (UNIC, 2020).

3.3.1 El reto de la coherencia entre niveles

El modelado de series jerárquicas plantea un desafío específico: garantizar que las predicciones realizadas en distintos niveles sean mutuamente consistentes. Si se predicen por separado las ventas de cada producto y, también por separado, las ventas totales de la tienda, no hay ninguna garantía matemática de que la suma de las predicciones individuales coincida con la predicción agregada. Esta incoherencia es problemática tanto desde el punto de vista estadístico como operativo, ya que distintos departamentos de la empresa tomarían decisiones a partir de cifras inconsistentes (Hyndman & Athanasopoulos, 2018).

Para resolver este problema, la literatura ha desarrollado tres familias principales de métodos de reconciliación:

Enfoque de abajo hacia arriba Bottom-Up: se generan predicciones únicamente en el nivel más granular y se obtienen los niveles superiores mediante agregación directa. Es el enfoque más intuitivo y garantiza coherencia automática, pero presenta dos limitaciones importantes: por un lado, las series del nivel inferior suelen ser las más ruidosas e intermitentes, lo que dificulta la predicción; por otro, los errores cometidos en el nivel granular se acumulan al agregar hacia arriba (Hyndman & Athanasopoulos, 2018).

Enfoque arriba hacia abajo Top-Down: se predice únicamente la serie del nivel superior y se distribuye hacia los niveles inferiores aplicando proporciones históricas. Es robusto frente al ruido del nivel granular, pero tiene una limitación estructural: pierde la información específica de cada producto o tienda, lo que lo hace menos preciso en niveles inferiores (Hyndman & Athanasopoulos, 2018).

Reconciliación Óptima **(**Optimal Reconciliation**)**: representa el estado del arte actual en predicción jerárquica. En lugar de elegir un único nivel de partida, se generan predicciones independientes en todos los niveles simultáneamente y se ajustan posteriormente mediante mínimos cuadrados generalizados para garantizar la coherencia global, minimizando el error total. Este enfoque mantiene la riqueza informativa de cada

nivel y produce predicciones simultáneamente coherentes y óptimas en términos de varianza (Wickramasuriya, 2019).

La elección entre estos enfoques en la práctica depende del nivel de la jerarquía sobre el que se evalúe el modelo. De hecho, los propios organizadores del M5 utilizan estos métodos en sus benchmarks oficiales: los modelos aplicados sobre el nivel más granular (como Naive, ARIMA Bottom-Up o los algoritmos de Machine Learning) emplean el método Bottom-Up para reconstruir el resto de niveles, mientras que los modelos aplicados sobre el nivel agregado superior (como ARIMA Top-Down) utilizan el método Top-Down con proporciones históricas estimadas sobre los últimos 28 días (Makridakis, 2022).

Alcance del presente trabajo. El modelado jerárquico completo, con reconciliación óptima entre los 12 niveles del M5, constituye por sí solo un campo de investigación de gran envergadura y permanece como una de las líneas de investigación más activas en *forecasting* (Wickramasuriya, 2019). Por este motivo, queda fuera del alcance del presente trabajo, que se centra en la comparación del rendimiento predictivo de los modelos en su nivel de agregación natural: los modelos estadísticos clásicos (SARIMA, Prophet) sobre la serie agregada de la tienda CA_1, y el modelo XGBoost sobre el nivel SKU completo (los 30.490 productos individuales). Esta decisión metodológica permite evaluar de forma rigurosa la viabilidad de cada modelo en el nivel de agregación para el que está diseñado, sin introducir la complejidad adicional de la reconciliación jerárquica como variable confusora del análisis.

La comprensión de esta estructura jerárquica es, no obstante, esencial para interpretar correctamente los resultados de los modelos: como se argumentará en los apartados siguientes, la elección del nivel de agregación es una de las decisiones de diseño más críticas del trabajo, ya que determina directamente qué modelos resultan aplicables y cómo deben interpretarse sus errores predictivos.

Una vez establecidos los componentes de las series temporales y la estructura jerárquica del dataset, en los siguientes apartados se presentan los modelos predictivos que se aplicarán sobre estas series, comenzando por los enfoques estadísticos clásicos.

3.4 Modelos Estadísticos Clásicos

Los modelos estadísticos para series temporales constituyen el enfoque analítico fundamental para el pronóstico de la demanda. A diferencia de los algoritmos de *Machine Learning*, estas metodologías se basan en formulaciones matemáticas explícitas que utilizan datos secuenciales para modelar, aislar y proyectar los componentes internos de la serie, tales como la tendencia, la estacionalidad y el ruido blanco (residuos). Dentro de esta familia conviven tanto formulaciones clásicas fuertemente consolidadas como extensiones contemporáneas. Los principales enfoques son los siguientes:

- **Modelos Clásicos de Referencia (Naive y Naive Estacional):** Métodos probabilísticos fundamentales que asumen que el futuro replicará la observación más reciente o el último ciclo estacional. Constituyen el pilar clásico indispensable que sirve como línea base (*baseline*) rigurosa para evaluar y justificar el uso de algoritmos más complejos.
- **ARIMA / SARIMA (Autoregressive Integrated Moving Average):** Representa uno de los enfoques clásicos más robustos (basado en la histórica metodología

Box-Jenkins). Mediante su variante estacional (SARIMA), permite modelar y proyectar datos capturando patrones cíclicos a través de la diferenciación y la combinación de componentes autorregresivos y de media móvil.

- **Suavizado Exponencial y Holt-Winters:** Familia clásica de modelos predictivos que asignan pesos decrecientes de forma exponencial a las observaciones más antiguas. La formulación de Holt-Winters es un estándar histórico altamente eficaz para aislar y predecir los componentes de nivel, tendencia y estacionalidad.
- **Modelo Theta:** Metodología basada en la descomposición de la serie temporal original en múltiples líneas Theta, lo que permite capturar tanto la curvatura local como la tendencia a largo plazo. Es un método reconocido por su excelente rendimiento empírico.
- **MSTL, BATS y TBATS:** Algoritmos modernos de espacio de estados diseñados específicamente para el tratamiento de series temporales complejas. Destacan por su capacidad para identificar y modelar estacionalidades múltiples superpuestas de forma simultánea.

De los mencionados, se toma como punto de partida la familia ARIMA por ser el benchmark estadístico de referencia en la competición M5 (Makridakis, 2022). Como se demostrará en el marco teórico y en el análisis exploratorio, las características específicas del dataset (estacionalidad semanal, intermitencia extrema y escala jerárquica) exigirán evolucionar desde ARIMA hacia SARIMA.

3.4.1 Modelos base de referencia

Antes de aplicar modelos estadísticos o algorítmicos más complejos, se consideran varios métodos de referencia extremadamente simples, cuyo objetivo es proporcionar una línea base mínima de rendimiento. Estos métodos no realizan ningún proceso de aprendizaje ni estimación de parámetros, sino que se basan en reglas heurísticas directas. Su función es doble: por un lado, establecer un punto de comparación riguroso; por otro, ilustrar qué información mínima puede explotarse para generar una predicción razonable.

3.4.2 Media histórica

El método de la **media histórica** asume que el mejor estimador de los valores futuros de la serie es el promedio de todas las observaciones pasadas disponibles. Bajo este enfoque, se considera que la serie no presenta ni tendencia ni estacionalidad relevante, y que las fluctuaciones observadas responden únicamente a ruido aleatorio alrededor de un nivel constante.

Formalmente, la predicción para un horizonte h se define como:

$$\hat{y}_{T+h|T} = \frac{1}{T}\sum_{t=1}^{T} y_t$$

donde T representa el número total de observaciones disponibles hasta el instante actual. Esta predicción es constante para cualquier valor del horizonte h, ya que el método ignora por completo la dinámica temporal de la serie.

Aunque su simplicidad lo hace inapropiado para la mayoría de las series reales, el método de la media histórica resulta útil como referencia teórica, ya que permite cuantificar el beneficio de incorporar información temporal frente a un enfoque completamente estático.

3.4.3 Método ingenuo (Naive)

El método ingenuo (Naive) introduce una mínima dependencia temporal al asumir que el valor futuro de la serie será igual al último valor observado. En este caso, se considera que el estado más reciente de la serie encapsula toda la información relevante para la predicción inmediata.

La formulación del método es:

$$\hat{y}_{T+h|T} = y_T$$

donde y_T es el último valor observado antes del horizonte de predicción. Esta predicción es idéntica para todos los pasos del horizonte h.

El método Naive captura de forma implícita la inercia de la serie, pero ignora cualquier patrón sistemático más complejo, como tendencias o estacionalidades. A pesar de su simplicidad, constituye un baseline sorprendentemente competitivo en muchas aplicaciones y se utiliza de forma habitual como referencia mínima en la literatura de pronóstico.

3.4.4 Predicción Naive estacional

La predicción Naive estacional extiende el método ingenuo incorporando información sobre la estacionalidad de la serie. En lugar de copiar el último valor observado, el método asume que el comportamiento futuro replicará el valor observado en el mismo punto del ciclo estacional anterior.

La predicción se define como:

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$$

donde $m$es el periodo estacional (por ejemplo, $m = 7$ en series diarias con estacionalidad semanal) y $k = \lfloor (h-1)/m \rfloor$ indica cuántos ciclos completos hacia atrás debe retrocederse para localizar la observación de referencia.

Este método permite capturar de forma directa patrones estacionales dominantes (como el comportamiento semanal en el contexto del retail) sin necesidad de estimar ningún parámetro. Por este motivo, la predicción Naive estacional se utiliza habitualmente como benchmark mínimo en problemas con estacionalidad clara y constituye uno de los modelos de referencia oficiales de la competición M5 (Hyndman & Athanasopoulos, 2018; Makridakis et al., 2022).

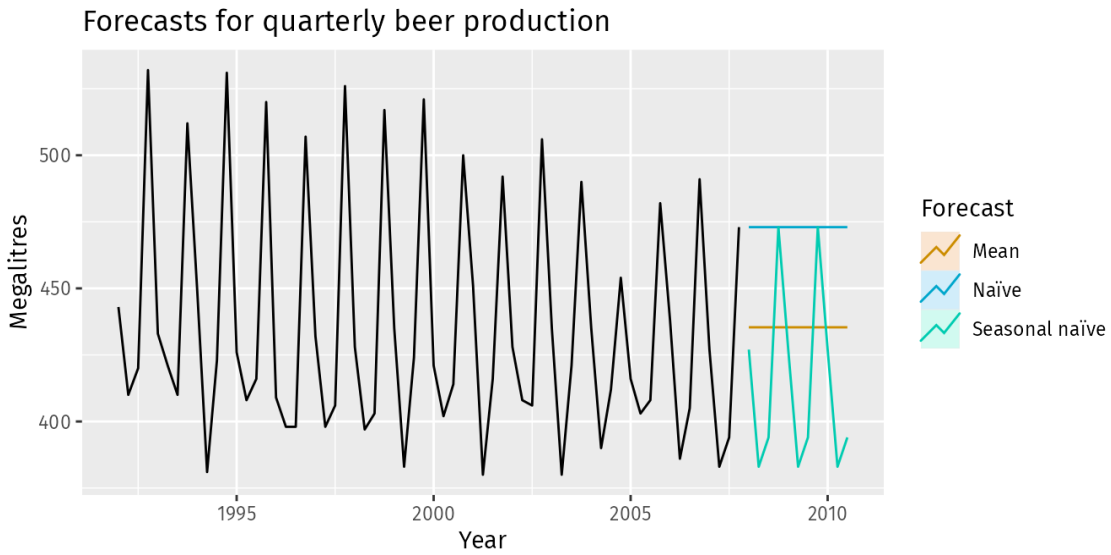

Figura 6: Predicción de producción trimestral de cerveza australiana (Hyndman & Athanasopoulos, 2018)

En la Figura 6 se muestra un ejemplo de los tres ultimo métodos aplicados una serie temporal que recoge el volumen de cerveza australiana producida trimestralmente.

3.5 Modelos ARMA Y ARIMA

Los modelos de series temporales univariados son una clase de especificación para modelar y predecir variables utilizando solo la información contenida en sus propios valores pasados, o bien, errores pasados. Son univariados simplemente por el hecho de que tratan de explicar la serie a través de ella misma (Brooks, 2019).

En series temporales, el valor de hoy depende fuertemente del valor de ayer (autocorrelación), por lo que se usan modelos específicos como ARIMA (Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M., 2015) o Exponential Smoothing (Winters, 1960). La función de predicción de un modelo ARIMA tiene una estructura simple, donde el término de integración (I) determina la predicción a largo plazo, mientras que los operadores estacionarios, AR y MA, determinan la predicción a corto plazo (Hyndman & Athanasopoulos, 2018). Un modelo ARIMA se clasifica por sus tres parámetros principales (p, d, q).

AR (componente autorregresivo): donde el valor actual de la variable (se predice utilizando una combinación lineal de los valores anteriores más el error aleatorio o ruido blanco (, es decir, el valor de la variable (dependerá solo de los valores que la variable tuvo en periodos anteriores más un término de error (Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M., 2015).

Sea una secuencia de errores aleatorios que cumple dos propiedades: su valor promedio es cero (los errores se compensan a largo plazo y no introducen un sesgo sistemático) y su dispersión es constante en el tiempo (la magnitud típica de los errores no cambia con el tiempo). Formalmente:

$$E[\varepsilon_t] = 0 \text{ y } \text{Var}(\varepsilon_t) = \sigma^2$$

$E[\varepsilon_t] = 0$ $\text{Var}(\varepsilon_t) = \sigma^2$ donde expresa que el valor esperado del error es nulo (los errores se compensan a largo plazo y no introducen un sesgo sistemático), y indica que la

dispersión de los errores es constante en el tiempo, propiedad conocida como homocedasticidad (Hyndman & Athanasopoulos, 2018)

Un modelo AR(p) predice el valor actual de como una combinación lineal de sus valores anteriores más un término de error aleatorio:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

donde $y_t$es el valor de la serie en el instante t, c es una constante o intercepto del modelo, p indica cuántos periodos anteriores utiliza el modelo para realizar la predicción (orden autorregresivo), $\phi_1, \phi_2, \ldots, \phi_p$ son los pesos que el modelo asigna a cada uno de esos retardos (cuanto mayor es el peso en valor absoluto, más influencia tiene ese instante pasado sobre la predicción actual) y $\varepsilon_t$es el término de error aleatorio o ruido blanco en el instante t.

MA (Media Móvil): Modela el valor actual como una combinación lineal de los errores de predicción pasados, es decir, el valor actual (dependerá de lo valores de error actuales y anteriores.

Sea un proceso de ruido blanco tal que $E[\varepsilon_t] = 0$ y $\text{Var}(\varepsilon_t) = \theta^2$. Un modelo MA(q) modela el valor actual como una combinación lineal de los errores de predicción pasados, suavizando las anomalías aleatorias de la serie (Hyndman & Athanasopoulos, 2018):

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

donde $y_t$es el valor de la serie en el instante t, c es una constante, q indica cuántos errores de predicción anteriores utiliza el modelo para ajustar la predicción actual (orden de media móvil), $\theta_1, \theta_2, \ldots, \theta_q$ son los pesos asignados a cada uno de esos errores pasados (cuanto mayor es el peso, más influye el error pasado sobre la corrección que aplica el modelo) y $\varepsilon_t, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q}$ son los errores aleatorios o ruido blanco en cada instante.

3.5.1 ARMA (autorregresivo media móvil)

Una vez definidos por separado los procesos autorregresivos (AR) y de media móvil (MA), el siguiente paso natural es combinarlos en un único modelo: el modelo ARMA (p, q), que captura simultáneamente dos tipos de información del pasado: los valores anteriores de la propia serie (parte AR) y los errores de predicción cometidos en periodos previos (parte MA). Su formulación general es:

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

donde y_t representa el valor de la serie en el instante t, c es una constante del modelo, y los coeficientes ϕ_i ponderan la influencia de los $p$valores pasados de la propia serie sobre el valor actual. El segundo sumatorio introduce la componente de media móvil, en la que los coeficientes θ_j capturan el efecto de los errores de predicción cometidos en los $q$periodos anteriores. El término ε_t corresponde a un proceso de ruido blanco, con esperanza nula y varianza constante, que recoge la parte no explicada por el modelo.

De este modo, el modelo ARMA combina simultáneamente dos fuentes de información: la dependencia temporal de la serie respecto a sus propios valores pasados (componente autorregresiva, AR) y la estructura temporal de los errores de predicción (componente de media móvil, MA). Esta formulación permite capturar patrones dinámicos de corto plazo

con mayor flexibilidad que los modelos puramente autorregresivos o puramente de media móvil, constituyendo la base de los modelos ARIMA y SARIMA utilizados posteriormente en este trabajo (Hyndman & Athanasopoulos, 2018).
De este modo, el modelo aprovecha tanto la inercia temporal de la serie como la estructura del error, proporcionando una representación más flexible y precisa que los modelos puramente autorregresivos o puramente de media móvil (Hyndman & Athanasopoulos, 2018).
Sin embargo, ARMA presenta una limitación crítica: solo es aplicable a series estacionarias, es decir, series cuya media y varianza permanecen constantes en el tiempo y que no presentan tendencia ni estacionalidad (Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M., 2015). En la práctica, la mayoría de las series temporales reales (y muy especialmente las de retail, como el dataset de este trabajo) no cumplen esta condición: presentan tendencias crecientes, ciclos estacionales y cambios de nivel a lo largo del tiempo.
Para superar esta limitación se introduce el componente de integración (I), dando lugar al modelo ARIMA, que extiende ARMA aplicando previamente una o varias diferenciaciones a la serie original hasta convertirla en estacionaria. De esta forma, ARIMA puede entenderse como un modelo ARMA aplicado sobre la versión diferenciada de la serie, lo que le permite operar sobre series con tendencia.

3.5.2 ARIMA (autorregresivo integrado con media móvil)

La solución que propone el modelo ARIMA para tratar series con tendencia es la diferenciación. En lugar de modelar directamente la serie original, el modelo trabaja sobre los cambios entre observaciones consecutivas, es decir, sobre la diferencia entre el valor actual y el inmediatamente anterior. Esta transformación permite eliminar la tendencia y convertir la serie en estacionaria, condición necesaria para aplicar modelos autorregresivos y de media móvil.
Formalmente, la diferenciación de primer orden se define como:

$$y'_t = y_t - y_{t-1}$$

donde la nueva serie y'_t ya no representa el nivel original de la variable, sino la variación de un periodo a otro. En una serie con tendencia creciente, esta operación transforma una trayectoria ascendente en una serie que oscila alrededor de un valor aproximadamente constante, eliminando así la componente de tendencia.
Desde el punto de vista intuitivo, la diferenciación convierte una serie de niveles en una serie de incrementos. Por ejemplo, si las ventas aumentan de forma progresiva cada día, la serie original presenta una media creciente; sin embargo, tras diferenciarla, los incrementos diarios tienden a estabilizarse alrededor de un valor fijo. Esta propiedad es la que permite que los modelos ARMA (que asumen estacionariedad) puedan aplicarse sobre la serie transformada.
El parámetro d del modelo ARIMA indica cuántas veces se aplica esta operación. En la práctica, una diferenciación de orden uno suele ser suficiente para eliminar tendencias lineales, mientras que valores mayores son poco frecuentes y suelen indicar problemas estructurales distintos de la tendencia (Hyndman & Athanasopoulos, 2018).

3.5.3 De ARIMA a SARIMA: la necesidad de capturar la estacionalidad

A pesar de las capacidades que aporta la integración (I) para manejar series con tendencia, la formulación clásica de ARIMA presenta una segunda limitación estructural: no es capaz de modelar de forma adecuada los patrones estacionales, es decir, los ciclos que se repiten a intervalos regulares y predecibles a lo largo del tiempo (Hyndman & Athanasopoulos, 2018).

La diferencia entre tendencia y estacionalidad es fundamental para entender por qué ARIMA se queda corto. Como antes se ha explicado la tendencia es un movimiento sostenido y direccional (ventas que crecen mes a mes, por ejemplo) y se corrige mediante diferenciación ordinaria (I). La estacionalidad, en cambio, es un patrón cíclico que se repite cada cierto número fijo de periodos: las ventas de un supermercado son sistemáticamente más altas los sábados que los martes, semana tras semana. Esta repetición no es una tendencia que se pueda eliminar restando el día anterior, sino una estructura que conecta cada observación con la del mismo punto del ciclo anterior (el sábado pasado, no el viernes pasado).

ARIMA, en su formulación estándar, solo modela dependencias entre observaciones consecutivas o cercanas mediante sus retardos AR y MA. Esto le permite capturar la inercia de la serie (lo que sucedió ayer influye en lo que pasa hoy) pero le impide aprovechar la información estructural más relevante de muchas series reales: que el mismo día de la semana anterior (o el mismo mes del año anterior) suele ser un predictor mucho más preciso que los días inmediatamente previos. En el contexto del retail, donde el comportamiento semanal del consumidor es uno de los patrones más estables y predecibles, esta limitación resulta crítica.

Para resolver este problema, se propone una extensión del modelo ARIMA que incorpora un segundo conjunto de operadores autorregresivos, de media móvil y de diferenciación que actúan específicamente sobre los retardos estacionales. Esta extensión se conoce como SARIMA (*Seasonal AutoRegressive Integrated Moving Average*) y constituye el modelo estadístico clásico de referencia para series temporales con estacionalidad (Hyndman & Athanasopoulos, 2018), siendo el enfoque que se desarrollará en el siguiente apartado.

3.6 SARIMA (estacional autorregresivo integrado con media móvil)

El modelo SARIMA (Seasonal AutoRegressive Integrated Moving Average) constituye la extensión natural de ARIMA para series temporales que presentan estacionalidad. Su formulación parte exactamente de la misma base que ARIMA, pero incorpora un segundo conjunto de operadores (autorregresivo, de integración y de media móvil) que actúan específicamente sobre los retardos del ciclo estacional, multiplicándose con los operadores no estacionales del modelo base (Hyndman & Athanasopoulos, 2018).

La notación oficial del modelo es:

$$\text{SARIMA}(p, d, q)(P, D, Q)_m$$

donde p, d y q son los parámetros no estacionales (orden autorregresivo, orden de integración y orden de media móvil, idénticos a los de un modelo ARIMA), P, D y Q son sus equivalentes estacionales (orden autorregresivo estacional, número de

diferenciaciones estacionales y orden de media móvil estacional) y m es la frecuencia estacional, es decir, el número de observaciones que componen un ciclo completo (m = 7 para datos diarios con patrón semanal). Por convención se utilizan letras minúsculas para la parte no estacional y mayúsculas para la parte estacional, lo que permite identificar visualmente el rol de cada parámetro (Hyndman & Athanasopoulos, 2018).

Cada parámetro estacional cumple una función análoga a su homólogo no estacional, pero aplicada a los retardos del ciclo:

P (orden autorregresivo estacional): captura la relación entre el valor actual y los valores observados en el mismo punto de ciclos estacionales anteriores. Por ejemplo, con y , el modelo aprende cómo influye el valor de hace 7 días sobre el valor actual.

D (orden de integración estacional): representa el número de diferencias estacionales aplicadas para eliminar la estacionalidad. Una diferencia estacional consiste en restar a cada observación el valor del mismo punto del ciclo anterior. Habitualmente toma valores 0 o 1, ya que rara vez se necesitan más de dos diferenciaciones estacionales para estabilizar la serie (Hyndman & Athanasopoulos, 2018).

Q (orden de media móvil estacional): modela el error actual como una combinación lineal de los errores de predicción cometidos en ciclos estacionales previos.

m (frecuencia estacional): define el número de observaciones que componen un ciclo completo. Su valor depende directamente de la naturaleza de los datos: para series trimestrales, para series mensuales con estacionalidad anual, o para series diarias con estacionalidad semanal (Hyndman & Athanasopoulos, 2018).

La formulación matemática del modelo se expresa de forma compacta utilizando el operador de retardo. Por ejemplo, un modelo se escribe como:

$$(1 - \phi_1 B)(1 - \Phi_1 B^7)(1 - B)(1 - B^7)\, y_t = (1 + \theta_1 B)(1 + \Theta_1 B^7)\, \varepsilon_t$$

donde B es el operador de retardo, definido como $B\, y_t = y_{t-1}$ de modo que $B^k\, y_t = y_{t-k}$; ϕ_1 es el coeficiente autorregresivo no estacional, Φ_1 es el coeficiente autorregresivo estacional que actúa sobre el retardo B^7, θ_1 es el coeficiente de media móvil no estacional y Θ_1 su análogo estacional. Los factores $(1 - B)$ y $(1 - B^7)$ representan, respectivamente, la diferenciación ordinaria de orden 1 que elimina la tendencia y la diferenciación estacional de orden 1 que elimina el ciclo semanal; ε_t es el ruido blanco. Los términos estacionales se multiplican por los términos no estacionales, lo que permite al modelo capturar simultáneamente las dependencias a corto plazo (días consecutivos) y las dependencias estacionales (mismo día de la semana anterior) (Hyndman & Athanasopoulos, 2018).

Esta arquitectura resulta especialmente adecuada para el contexto del retail. En el caso del dataset M5, donde el análisis exploratorio identificará una estacionalidad semanal dominante, la frecuencia se establece en, lo que permite al modelo evaluar la demanda no solo en función del día inmediatamente anterior, sino conectándola directamente con el comportamiento del mismo día de la semana previa, capturando así el efecto estructural del fin de semana sobre las ventas (Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M., 2015).

No obstante, SARIMA hereda de ARIMA dos limitaciones críticas que condicionarán su aplicación práctica en este trabajo: por un lado, es un modelo univariante que debe ajustarse de forma independiente para cada serie temporal, lo que dificulta su

escalabilidad a datasets con miles de series como el dataset de este trabajo (Hyndman & Athanasopoulos, 2018); por otro, mantiene la asunción de estacionariedad del componente residual, lo que lo hace inadecuado para series con inflación de ceros como las que se observan a nivel de artículo único en retail (Box & Jenkins, 1970). Estas limitaciones se desarrollarán en detalle en el apartado de Análisis Exploratorio y justificarán las decisiones metodológicas posteriores.

3.7 Modelo Algorítmico Avanzado

Los modelos estadísticos clásicos presentados en el apartado anterior (ARIMA, SARIMA) se fundamentan en supuestos teóricos sólidos sobre la estructura de la serie temporal: estacionariedad, ruido blanco y relaciones lineales entre rezagos. Estas asunciones permiten que los modelos sean fáciles de interpretar y matemáticamente bien fundamentados, pero también imponen limitaciones cuando las series reales no las cumplen, como ocurre con la demanda intermitente del comercio minorista (retail) a nivel de producto individual (Hyndman & Athanasopoulos, 2018).

Frente a esta filosofía, los modelos algorítmicos avanzados (enmarcados dentro del aprendizaje automático *machine learning*) adoptan un enfoque radicalmente distinto: en lugar de imponer una estructura previa a la serie, aprenden los patrones directamente de los datos mediante algoritmos flexibles capaces de capturar relaciones no lineales, interacciones complejas entre variables y comportamientos heterogéneos entre series. Esta capacidad los hace particularmente adecuados para problemas con alta dimensionalidad, grandes volúmenes de datos y estructuras irregulares, como es el caso del dataset M5 (Meldgaard, 2021).

En este trabajo se selecciona XGBoost (*eXtreme Gradient Boosting*) como representante de esta familia. La elección no es arbitraria: en la competición M5 organizada por Walmart en 2020, los cinco primeros equipos clasificados en la modalidad de pronóstico puntual utilizaron variantes de potenciación del gradiente (Gradient Boosting) sobre árboles de decisión, lo que evidencia el dominio práctico de esta familia de algoritmos en problemas de pronóstico de demanda en retail (Makridakis, 2022). Esta tendencia no es exclusiva del M5: una revisión sistemática de seis competiciones de pronóstico en Kaggle observa el mismo patrón, con los modelos de *Gradient Boosting* dominando consistentemente las tablas de clasificación frente a los modelos estadísticos clásicos (Meldgaard, 2021). Para entender por qué XGBoost resulta tan eficaz en este contexto, los siguientes apartados recorren brevemente su genealogía conceptual: árboles de decisión, ensamblado de árboles mediante *Gradient Boosting* y, finalmente, las aportaciones específicas de XGBoost que lo distinguen del resto de implementaciones disponibles.

3.7.1 Árboles de decisión

Un árbol de decisión es un modelo predictivo que toma una decisión paso a paso, haciéndose preguntas sencillas del tipo sí/no sobre los datos hasta llegar a una conclusión. Cada pregunta divide los datos en dos grupos más parecidos entre sí (por ejemplo, "¿el día es fin de semana?" separaría los datos en dos: los que sí lo son y los que no) y el proceso se repite sobre cada uno de esos grupos hasta que todos los datos del grupo son lo suficientemente parecidos como para hacer una predicción común (Hastie et al., 2009).

El resultado es una estructura ramificada parecida a un árbol invertido: la pregunta inicial es la raíz, cada pregunta posterior es una bifurcación, y al final del recorrido están las predicciones, llamadas hojas porque son los extremos del árbol.

Para problemas en los que queremos predecir un número (como las ventas de un producto) se utilizan habitualmente los árboles CART (Classification and Regression Trees), propuestos originalmente por Breiman, Friedman, Olshen y Stone (Breiman et al., 1984). El algoritmo CART decide en cada bifurcación qué pregunta hacer probando todas las posibles divisiones de los datos. Para cada posible división, calcula la predicción que daría a cada uno de los dos grupos resultantes (normalamente el promedio de los valores reales del grupo, en problemas de regresión) y mide cuánto se aleja esa predicción de los valores reales mediante el error cuadrático medio. La pregunta que produce los dos grupos más homogéneos, es decir, los grupos donde el promedio se acerca más a los valores individuales es la que se selecciona.

Para ilustrar el funcionamiento sobre el dataset M5, considérese el caso en que el árbol debe decidir su primera pregunta sobre los 58 millones de observaciones del conjunto de entrenamiento. Si no hiciera ninguna pregunta, su predicción sería simplemente el promedio global de ventas, por ejemplo 1,5 unidades por SKU y día, una predicción muy imprecisa porque ignora las enormes diferencias entre productos, tiendas y días. El árbol prueba entonces cientos de preguntas posibles: "¿es fin de semana?", "¿el precio es superior a 5 €?", "¿está activo el programa SNAP?", "¿el producto vendió más de 3 unidades hace 28 días (lag_28 ≥ 3)?", etc. Para cada pregunta, divide los datos en dos grupos y calcula la predicción de cada uno como el promedio de las ventas reales de ese grupo. La pregunta sobre el lag_28, por ejemplo, podría producir un grupo de productos históricamente muy vendidos cuyo promedio sea 4,2 unidades, y otro grupo de productos poco vendidos cuyo promedio sea 0,4 unidades. La pregunta que el árbol selecciona como primera bifurcación es aquella cuyos dos grupos resultantes minimizan el error cuadrático medio total del grupo (es decir, aquella que mejor separa los datos en grupos donde los valores reales son consistentes con el promedio del grupo). Este proceso se repite recursivamente sobre cada nuevo grupo hasta que se cumple alguna condición de parada que el analista haya definido, como un número máximo de bifurcaciones o un mínimo de observaciones por hoja (Hastie et al., 2009).

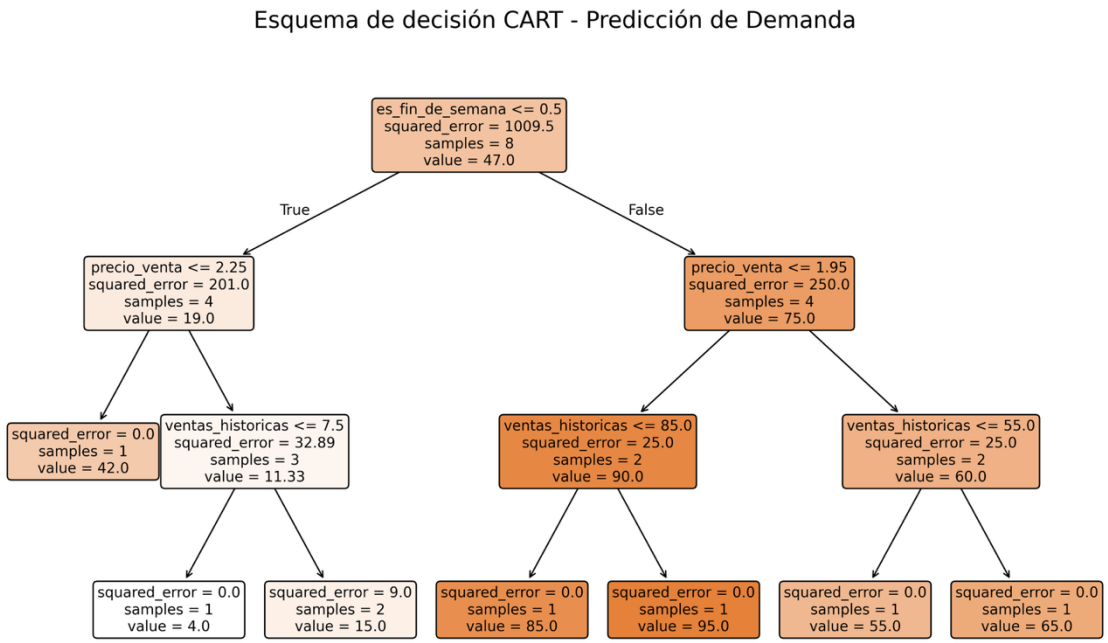

Figura 7: Ejemplo de árbol de decisión CART aplicado a la demanda retail - Fuente: Elaboración propia.

La principal virtud de los árboles de decisión es que sus predicciones son fáciles de entender: las preguntas que el modelo hace para llegar a una predicción son claras y se pueden leer directamente, lo que permite a un humano seguir el razonamiento paso a paso. Esta característica (conocida en machine learning como interpretabilidad) los hace especialmente atractivos en contextos donde la transparencia del modelo es relevante, como entornos empresariales o regulatorios en los que hay que justificar cada decisión automatizada (Hastie et al., 2009).

Sin embargo, un árbol único presenta dos limitaciones importantes que afectan a su precisión predictiva. La primera es que es muy sensible a pequeñas variaciones en los datos: si entrenamos dos árboles sobre dos versiones ligeramente distintas del mismo dataset (por ejemplo, eliminando unas pocas observaciones) los árboles resultantes pueden ser muy diferentes entre sí, lo que se traduce en predicciones inestables (alta varianza). La segunda es que tiende a aprenderse de memoria los datos de entrenamiento en lugar de aprender los patrones generales: si dejamos que el árbol crezca sin límite, acaba haciendo tantas preguntas que cada hoja contiene apenas una o dos observaciones específicas del entrenamiento, lo que produce un modelo que funciona perfectamente sobre los datos vistos, pero falla al enfrentarse a datos nuevos. Este fenómeno se denomina sobreajuste (overfitting) y es uno de los problemas centrales que los métodos modernos de aprendizaje automático buscan evitar (Hastie et al., 2009).

Estas limitaciones motivan el desarrollo de métodos que combinan múltiples árboles para construir modelos más estables y precisos, como se describe en el siguiente apartado.

3.7.2 Ensamblado de árboles y Gradient Boosting

Las limitaciones de un árbol único (su inestabilidad y su tendencia al sobreajuste) motivaron el desarrollo de los métodos de ensamblado (ensemble methods), que combinan múltiples árboles para obtener un modelo agregado más robusto y preciso. La idea de fondo es combinar adecuadamente muchos modelos sencillos para producir

predicciones más estables y precisas que cualquier modelo individual por sí solo: un conjunto de modelos débiles, combinados adecuadamente, puede superar el rendimiento de cualquier modelo individual (Hastie et al., 2009).
Existen dos formas principales de combinar árboles, que se diferencian en cómo se entrenan y cómo se combinan sus predicciones.
La primera forma consiste en entrenar varios árboles en paralelo, cada uno sobre un trozo distinto y aleatorio de los datos, y promediar las predicciones de todos al final. Este enfoque se llama agregación bootstrap (Bagging) y es la base del algoritmo Random Forest. Aunque cada árbol individual tenga errores específicos por haber visto solo una parte de los datos, esos errores tienden a compensarse al promediar muchos árboles, produciendo un modelo final más estable que cualquiera de ellos por separado (Hastie et al., 2009).
La segunda forma consiste en entrenar los árboles uno detrás de otro, donde cada nuevo árbol se construye específicamente para corregir los fallos del modelo acumulado (anteriores) hasta ese momento. Este enfoque se llama potenciación (Boosting) y produce modelos que típicamente alcanzan una precisión predictiva mayor que el Bagging, a costa de un proceso de entrenamiento más lento por su naturaleza secuencial (Hastie et al., 2009).

3.7.3 Gradient Boosting

Dentro de la familia de algoritmos *Boosting*, el más influyente en problemas de regresión es Gradient Boosting, propuesto por Friedman (Friedman, 2001). Su idea central es: en lugar de intentar que cada árbol nuevo prediga directamente la variable objetivo (las ventas, en este caso), se le pide que prediga los errores que está cometiendo el modelo hasta ese momento. Estos errores se denominan residuos (desarrollado ya anteriormente) y son simplemente la diferencia entre lo que el modelo predice y lo que realmente ocurrió.
Para ilustrarlo en el contexto del dataset M5, considérese el caso en que el modelo predice las ventas de un producto. El primer árbol construido predice, por ejemplo**,** 2 unidades para un día concreto. Pero las ventas reales de ese día fueron 5 unidades: el árbol se ha quedado corto en 3 unidades. Esa diferencia de 3 unidades es el residuo. El segundo árbol del modelo no se entrena sobre las ventas reales (5), sino sobre ese residuo (3): su misión específica es aprender a predecir cuánto se está equivocando el primer árbol. Si el segundo árbol predice un residuo de 2,5 unidades para ese mismo día, la predicción acumulada del modelo se actualiza a 2 + 2,5 = 4,5 unidades. Queda todavía un residuo de 0,5 unidades, que el tercer árbol intentará capturar. Y así sucesivamente.
Formalmente, la predicción final del modelo es la suma de las predicciones de todos los árboles construidos:

$$\hat{y}_i = \sum_{k=1}^{K} f_k\,(x_i), f_k \in \mathcal{F}$$

donde $\hat{y}_i$es la predicción final del modelo para la observación i, Kes el número total de árboles construidos durante el entrenamiento, $f_k$es el árbol entrenado en la iteración k(perteneciente al espacio funcional $\mathcal{F}$ de árboles de regresión) y $x_i$es el vector de variables explicativas asociado a la observación i. En cada iteración el nuevo árbol se

ajusta mediante descenso de gradiente sobre la función de pérdida, de modo que cada árbol se construye para reducir al máximo el error residual acumulado (Friedman, 2001; Chen & Guestrin, 2016).

La ventaja conceptual de este enfoque puede ilustrarse mediante un ejemplo. Considérese el caso en que el modelo predice las ventas de un producto. Un modelo lineal (como los que asumen ARIMA o SARIMA) puede aprender una relación del tipo: *"por cada euro que sube el precio, las ventas bajan en 2 unidades"*. Esta relación es simple, fija y la misma para todos los productos. Sin embargo, en el comercio real las cosas no funcionan así: una bajada de precio del 10 % puede no tener efecto en un día normal, pero disparar las ventas un 50 % si coincide con un viernes cerca del día de cobro y un día SNAP (Programa de Asistencia Nutricional Suplementaria). La relación entre precio y demanda no es lineal, depende del contexto y cambia entre productos.

Los algoritmos basados en árboles de decisión capturan exactamente este tipo de comportamiento de forma natural. Cada árbol del modelo puede aprender una regla del estilo "si el día es viernes Y es día SNAP Y el precio ha bajado más del 10 %, entonces las ventas se disparan" sin necesidad de que el analista la especifique de antemano. El modelo descubre estas interacciones entre variables automáticamente, simplemente al ir construyendo árboles que dividen el espacio de datos en regiones cada vez más específicas (Hastie et al., 2009).

Esta capacidad supone una ventaja decisiva en el contexto del dataset M5. Las variables disponibles (día de la semana, mes, festivos, días SNAP, precio, identificador de producto, tienda, departamento) no se relacionan con las ventas de forma lineal ni independiente. El efecto del precio depende del producto. El efecto del fin de semana depende de la categoría. El efecto de un festivo depende del estado. Capturar estas interacciones con un modelo lineal exigiría que el analista especifique manualmente todas las combinaciones relevantes, algo inviable cuando hay decenas de miles de productos. Los modelos basados en *Gradient Boosting*, en cambio, las descubren automáticamente a partir de los datos.

Esto contrasta con los modelos estadísticos clásicos como ARIMA y SARIMA, que asumen relaciones lineales entre la variable y sus rezagos, no admiten variables externas de forma nativa, y requieren transformaciones manuales de la serie (diferenciación, logaritmo) para satisfacer sus supuestos teóricos de estacionariedad y normalidad de los residuos (Hyndman & Athanasopoulos, 2018).

A lo largo de las últimas dos décadas, Gradient Boosting ha dado lugar a un conjunto de implementaciones que han optimizado su eficiencia computacional y su capacidad de generalización. Las más extendidas en la literatura y en la industria son XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017) y CatBoost (Prokhorenkova et al., 2018). De estas, XGBoost es la implementación seleccionada en este trabajo, y se desarrolla en detalle en el siguiente apartado.

3.7.4 XGBoost

XGBoost (eXtreme Gradient Boosting) es la implementación de Gradient Boosting propuesta por Chen y Guestrin (Chen & Guestrin, 2016) que ha consolidado a esta familia de algoritmos como referencia en el aprendizaje automático aplicado. Aunque conserva

la idea fundamental del algoritmo de Friedman en 2001 (construir un modelo aditivo de árboles entrenados secuencialmente para corregir los errores de los anteriores), XGBoost introduce un conjunto de innovaciones algorítmicas, estadísticas y computacionales que explican su dominio en competiciones de pronóstico y, en particular, su victoria en la M5 (Makridakis et al., 2022).

Las aportaciones de XGBoost respecto al *Gradient Boosting* clásico se agrupan en tres frentes complementarios.

Penalización de la complejidad del modelo. La primera contribución de XGBoost es estadística. El *Gradient Boosting* clásico se limita a minimizar el error sobre los datos de entrenamiento, lo que puede llevar al modelo a construir árboles cada vez más complejos para acertar en cada observación particular (incluyendo el ruido aleatorio que no representa ningún patrón generalizable). XGBoost añade una penalización que castiga directamente esa complejidad: árboles con muchas hojas o con valores extremos en sus predicciones reciben una puntuación peor, aunque su error sobre los datos de entrenamiento sea bajo. Esta técnica se denomina regularización en aprendizaje automático (Chen & Guestrin, 2016).

Formalmente, la fórmula que el modelo intenta minimizar durante el entrenamiento se expresa como:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \Omega(f) = \gamma T + \frac{1}{2}\lambda \parallel w \parallel^2$$

donde $\mathcal{L}(\phi)$ es la función objetivo que el modelo intenta minimizar durante el entrenamiento. El primer término, $\sum_i l(\hat{y}_i, y_i)$, mide la pérdida sobre las observaciones (cuánto se aleja la predicción $\hat{y}_i$ del valor real y_i), y el segundo término, $\sum_k \Omega(f_k)$, penaliza la complejidad de cada árbol del conjunto. En la expresión $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$, T es el número de hojas del árbol y w el vector de pesos asignados a cada hoja; γ controla la penalización por hoja (favorece árboles pequeños) y λ controla la penalización L2 sobre los pesos (evita valores extremos). De este modo el modelo no solo busca acertar, sino acertar con árboles tan simples como sea posible, lo que reduce de forma sistemática el riesgo de sobreajuste (Chen & Guestrin, 2016).

Tratamiento nativo de datos con muchos ceros. La segunda contribución de XGBoost es algorítmica. Los conjuntos de datos reales contienen frecuentemente datos con muchos ceros o valores ausentes (*sparse data*): observaciones con campos vacíos, codificaciones binarias resultantes de variables categóricas o, como ocurre en el caso del M5, una proporción muy alta de ceros en la variable que se intenta predecir. El *Gradient Boosting* clásico requiere rellenar estos huecos antes del entrenamiento mediante alguna estrategia de imputación, lo que introduce sesgos y aumenta el coste computacional.

XGBoost incorpora un mecanismo, denominado *sparsity-aware split finding*, que aprende automáticamente durante la construcción de cada árbol cuál es la dirección óptima de partición para los valores ausentes o nulos. Cuando una observación tiene un cero o un valor faltante en una variable, el modelo decide por sí mismo si es más informativo enviarla a la rama izquierda o a la derecha del árbol, basándose en el comportamiento del resto de los datos (Chen & Guestrin, 2016). En la práctica, esto significa que el algoritmo no necesita decisiones de imputación previas: trata los ceros y

los valores faltantes como información válida del problema, no como datos a corregir. Esta característica resulta especialmente relevante en el contexto del pronóstico de demanda en *retail*, donde los ceros no son errores sino la respuesta correcta para la mayoría de los productos en la mayoría de los días.

Optimizaciones para procesar grandes volúmenes de datos. La tercera contribución de XGBoost es de ingeniería. El algoritmo incorpora paralelización en la construcción de los árboles, optimización del acceso a memoria caché y un método de búsqueda de divisiones basado en agrupar los valores continuos en intervalos discretos antes de probar cada posible punto de corte, en lugar de evaluar uno por uno todos los valores únicos del dataset. Esta técnica se denomina construcción basada en histogramas (*histogram-based*) y reduce drásticamente el coste computacional sin sacrificar precisión apreciable (Chen & Guestrin, 2016). Estas optimizaciones permiten escalar el algoritmo a datasets con miles de millones de observaciones manteniendo tiempos de entrenamiento manejables, lo que ha sido determinante para su adopción en problemas industriales reales como el del presente trabajo, donde el dataset de entrenamiento contiene 58 millones de observaciones.

Adicionalmente, XGBoost permite especificar funciones que miden el error del modelo de forma personalizada (*custom loss functions*), más allá del error cuadrático medio estándar. Esta flexibilidad resulta relevante porque cada tipo de función implica un supuesto distinto sobre la distribución estadística de los datos: la pérdida cuadrática asume que los errores siguen una distribución normal simétrica, la pérdida logística asume una distribución binomial, y otras funciones (como la pérdida Tweedie) permiten modelar distribuciones específicas como las que se observan en demanda intermitente, donde coexisten una masa de probabilidad concentrada en cero con una cola de valores positivos continuos (Smyth & Jørgensen, 2002). La elección de la función adecuada en función de la naturaleza de los datos constituye una de las decisiones de diseño más importantes al aplicar XGBoost, y se desarrolla en detalle en el apartado de Desarrollo Experimental.

La combinación de estas características explica por qué XGBoost se ha convertido en una de las referencias del *Gradient Boosting* aplicado al pronóstico de demanda. No obstante, en los últimos años han surgido otras implementaciones que parten del mismo paradigma y compiten directamente con él. Las dos más extendidas son LightGBM (Ke et al., 2017), desarrollado por Microsoft, que prioriza la eficiencia computacional mediante un crecimiento de árboles por hojas (*leaf-wise*) en lugar de por niveles; y CatBoost (Prokhorenkova et al., 2018), desarrollado por Yandex, que introduce un tratamiento específico de variables categóricas y técnicas para reducir el sesgo de predicción. En la competición M5, los modelos ganadores se construyeron sobre LightGBM, lo que confirma que el éxito de esta familia de algoritmos no se limita a una implementación concreta sino al paradigma de *Gradient Boosting* sobre árboles de decisión (Makridakis et al., 2022).

3.7.5 Prophet

Prophet es una biblioteca de pronóstico de series temporales desarrollada en 2017 por Sean J. Taylor y Benjamin Letham en el equipo de ciencia de datos de Meta (Facebook en aquel momento) y publicada como software libre con el objetivo explícito de

democratizar el pronóstico de demanda dentro de la propia compañía (Taylor & Letham, 2018). A diferencia de XGBoost, que pertenece al paradigma del aprendizaje automático basado en árboles, y de ARIMA y SARIMA, que se inscriben en el paradigma estadístico autorregresivo, Prophet se sitúa en una tercera familia conceptualmente distinta: los modelos aditivos descomponibles. La elección de este enfoque responde a una motivación eminentemente práctica más que estrictamente estadística, lo que constituye una de las características diferenciadoras del modelo.

Los autores diseñaron Prophet a partir de tres observaciones empíricas recurrentes en entornos empresariales reales (Taylor & Letham, 2018). En primer lugar, las series temporales de demanda suelen presentar múltiples ciclos estacionales superpuestos (diarios, semanales y anuales) cuya interacción resulta difícil de capturar con modelos univariantes clásicos. En segundo lugar, estas series se ven afectadas con frecuencia por festivos y eventos especiales que introducen rupturas abruptas en la regularidad esperada del patrón temporal. Finalmente, la mayoría de los analistas que necesitan generar pronósticos en entornos productivos no son expertos en estadística de series temporales, por lo que requieren herramientas capaces de producir resultados razonables con una configuración mínima y con un comportamiento estable cuando se aplican a gran escala. Prophet se construye explícitamente para satisfacer estas tres condiciones, lo que sus autores denominan *forecasting at scale*.

Desde el punto de vista matemático, Prophet modela la serie temporal como la suma de varios componentes funcionales independientes definidos explícitamente sobre el eje temporal. La formulación general del modelo es:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

donde la función $g(t)$ describe la evolución de la tendencia a largo plazo, $s(t)$recoge los patrones estacionales periódicos, $h(t)$representa el efecto de festivos y eventos especiales conocidos con antelación, y $\varepsilon_t$es un término de error aleatorio que captura la variabilidad no explicada por los componentes anteriores (Taylor & Letham, 2018).

Esta arquitectura diferencia de manera fundamental a Prophet de los modelos autorregresivos clásicos como ARIMA y SARIMA. Mientras que estos últimos asumen que el valor de la serie en un instante dado depende explícitamente de sus propios valores pasados y de errores anteriores, Prophet trata la serie temporal como una función explícita del tiempo y la descompone en componentes interpretables que pueden analizarse de forma independiente. Esta diferencia conceptual justifica que Prophet sea descrito en la literatura como un modelo de regresión aplicado a series temporales, más que como un modelo de series temporales en sentido estrictamente autorregresivo (Hyndman & Athanasopoulos, 2021).

La componente de tendencia $g(t)$ puede modelarse mediante un crecimiento lineal por tramos (*piecewise linear*), que constituye la opción por defecto, o mediante un crecimiento logístico saturado cuando existe un límite superior conocido. En el caso lineal por tramos, Prophet incorpora un mecanismo automático de detección de puntos de cambio (*changepoints*), que identifica momentos del histórico en los que el ritmo de crecimiento de la serie se modifica de forma estructural. Por defecto, el modelo sitúa un conjunto de puntos de cambio candidatos uniformemente distribuidos sobre el primer 80

% del histórico y regula su influencia mediante una prior bayesiana, lo que permite capturar cambios de tendencia sin intervención manual del analista (Taylor & Letham, 2018).

La estacionalidad $s(t)$ se modela mediante series de Fourier truncadas, una técnica que permite aproximar funciones periódicas como la suma finita de funciones sinusoidales de distintas frecuencias (Bracewell, 2000). Para un periodo estacional P, el componente estacional se expresa como:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right)$$

donde el parámetro N controla la flexibilidad del componente estacional: valores mayores permiten capturar patrones más complejos, a costa de un mayor riesgo de sobreajuste. Esta formulación permite a Prophet representar simultáneamente múltiples ciclos estacionales, como la estacionalidad semanal y anual, mediante la suma de componentes independientes (Taylor & Letham, 2018).

El componente $h(t)$ modela explícitamente el efecto de festivos y eventos especiales. A diferencia de ARIMA y SARIMA, que requieren introducir estas variables como regresores exógenos mediante extensiones específicas, Prophet incorpora de forma nativa un mecanismo para tratarlos. El usuario proporciona un calendario de fechas relevantes y el modelo estima automáticamente el impacto asociado a cada una, pudiendo además definir ventanas temporales de influencia alrededor del evento. Esta capacidad resulta especialmente relevante en el contexto del dataset M5, donde el calendario incluye festivos religiosos, eventos deportivos y celebraciones comerciales cuyo efecto sobre la demanda varía significativamente entre categorías de producto.

El ajuste de los parámetros del modelo se realiza mediante inferencia bayesiana, implementada sobre el lenguaje probabilístico Stan (Carpenter et al., 2017). Por defecto, Prophet emplea estimación de máxima a posteriori (MAP), que ofrece un buen equilibrio entre precisión y eficiencia computacional, aunque también permite recurrir a muestreo MCMC completo cuando se requiere una caracterización más detallada de la incertidumbre. Esta arquitectura permite al modelo escalar a series con miles de observaciones manteniendo tiempos de ajuste del orden de segundos, cumpliendo así el objetivo de *forecasting at scale* establecido por sus autores.

Las ventajas principales de Prophet identificadas en la literatura aplicada son su facilidad de uso, su elevada interpretabilidad y su robustez frente a datos faltantes y valores atípicos. No obstante, estas mismas características introducen limitaciones derivadas de su arquitectura aditiva. Estudios empíricos muestran que Prophet puede ser superado por baselines simples, como el Naive Estacional, en series dominadas por un único ciclo estacional y con horizontes de predicción cortos, donde la descomposición aditiva penaliza la precisión punto a punto a cambio de propiedades de interpretabilidad que no se explotan plenamente en ese contexto (Hewamalage et al., 2021; Triebe et al., 2021).

En este trabajo, Prophet se incorpora como representante de un enfoque intermedio entre los modelos estadísticos clásicos y los algoritmos de aprendizaje automático, permitiendo

documentar empíricamente las condiciones bajo las cuales cada paradigma resulta más adecuado para el pronóstico de demanda en el contexto del retail.

3.7.6 Redes Neuronales y Aprendizaje Profundo

Aunque el presente trabajo se centra en la comparación entre modelos estadísticos clásicos (ARIMA, SARIMA) y algoritmos de aprendizaje automático (Prophet, XGBoost), resulta pertinente contextualizar el lugar que ocupan las redes neuronales dentro del panorama del pronóstico de series temporales, ya que representan la frontera más activa de la investigación en este campo.

Las arquitecturas más relevantes para series temporales son las Redes Neuronales Recurrentes (RNN) y su variante LSTM (Long Short-Term Memory). A diferencia de los modelos clásicos, son capaces de aprender dependencias temporales a muy largo plazo sin necesidad de especificar manualmente la estructura del modelo, lo que las convierte en herramientas poderosas ante series con patrones complejos y no lineales. Su aplicación al dataset M5 sería especialmente interesante por tres razones: (1) su capacidad de capturar el concept drift sin reentrenamiento manual; (2) su aptitud para integrar múltiples series simultáneamente mediante modelos globales; y (3) la riqueza de variables exógenas disponibles en M5 (precios, eventos, SNAP), que las redes neuronales pueden explotar con mayor flexibilidad que los modelos lineales.

Sin embargo, su implementación requiere recursos computacionales elevados y una fase de ajuste de hiperparámetros que excede el alcance de este TFG. Queda por tanto identificada como una línea de investigación futura de alto interés: comparar los resultados de XGBoost obtenidos en este trabajo con una arquitectura LSTM entrenada sobre el mismo dataset M5 constituiría una extensión natural y de gran valor académico.

4. Metodología

4.1 Conjunto de datos: (M5 Forecasting Competition - Accuracy)

El conjunto de datos empleado en esta investigación procede de la quinta edición de las M-Competitions , liberada por Walmart en colaboración con la Universidad de Nicosia y publicada en la plataforma Kaggle en marzo de 2020 (Makridakis et al., 2022). A diferencia de las cuatro ediciones anteriores, centradas en series macroeconómicas heterogéneas y descontextualizadas, M5 supone un hito metodológico en la literatura del pronóstico al ofrecer datos reales de un único contexto de negocio (las ventas minoristas) estructurados de forma jerárquica y enriquecidos con variables explicativas exógenas (Makridakis et al., 2022).

El conjunto recopila el histórico de ventas unitarias diarias de 3.049 productos distribuidos en 10 tiendas de los estados de California, Texas y Wisconsin, durante 1.941 días consecutivos comprendidos entre el 29 de enero de 2011 y el 19 de junio de 2016.

La información se reparte en cuatro ficheros que conforman un modelo relacional en esquema de copo de nieve (snowflake schema): la matriz principal de ventas (sales_train_validation.csv), su extensión cronológica para la evaluación (sales_train_evaluation.csv), que contiene los 28 días finales del horizonte de predicción y permite reproducir el escenario de test ciego de la competición M5; el calendario de variables temporales y eventos (calendar.csv) y el registro semanal de precios por tienda y producto (sell_prices.csv) (UNIC, 2020).

La estructura jerárquica del dataset se organiza en doce niveles de agregación, desde el nivel más granular (ventas de un artículo concreto en una tienda concreta) hasta el total nacional de la compañía, generando un total de 42.840 series temporales individuales que deben pronosticarse de forma coherente (Makridakis et al., 2022). Esta organización multinivel queda recogida de forma esquemática en la Figura 6, que reproduce la representación oficial de la jerarquía M5 publicada por la organización de la competición. La inclusión de variables exógenas como precios, festividades y activaciones del programa de asistencia alimentaria SNAP (Supplemental Nutrition Assistance Program) convierte a M5 en un entorno de pruebas (benchmark) particularmente representativo de los retos reales del sector minorista: la gestión de datos masivos, la elasticidad de precios y la extrema intermitencia de la demanda a nivel unitario (Fildes et al., 2022).

4.2 Restricciones Computacionales Y Optimización de Memoria

La gestión de recursos computacionales constituye uno de los retos reconocidos por la propia organización de la competición M5 (Makridakis et al., 2022). El comité organizador justifica la elección de los modelos de referencia (benchmarks) en función, entre otros criterios, de sus requisitos computacionales relativamente bajos, dado que numerosos participantes operaron bajo restricciones de memoria y tiempo de ejecución. Resulta pertinente, por tanto, documentar de forma explícita las decisiones técnicas adoptadas para gestionar la huella de memoria del dataset antes de abordar el modelado propiamente dicho.

El dataset M5 en su forma original suma 1.309,5 MB de memoria RAM al cargarse en pandas con los tipos asignados por defecto, según se obtiene en la sección «Optimización de Memoria» del notebook adjunto, mediante el método memory_usage(deep=True) aplicado a los tres ficheros del dataset. El parámetro deep=True garantiza que el método contabilice el tamaño real ocupado por las columnas de texto, y no solo el espacio que ocupan sus referencias internas (pandas development, 2024). La distribución de ficheros es asimétrica, es decir, la matriz de ventas sales_train_validation.csv contiene 30.490 filas y 1.919 columnas (de las cuales 6 son identificadores categóricos y 1.913 son los días d_1 a d_1913); el calendario calendar.csv cuenta con 1.969 filas y 14 columnas; y el registro de precios sell_prices.csv agrupa 6.841.121 observaciones en 4 columnas. La asimetría en la distribución de memoria (donde precios y ventas concentran prácticamente la totalidad de la huella) se debe tanto al volumen de filas como al uso por defecto de tipos numéricos de 64 bits y de cadenas de texto sin compresión.

De acuerdo con la documentación oficial de pandas, las columnas numéricas se inicializan por defecto como int64 o float64 (enteros y flotantes con signo de 64 bits), mientras que las columnas de texto se almacenan como object, lo que delega su gestión en estructuras de Python de tamaño variable y elevada memoria (pandas development, 2024). En una fase posterior del flujo de trabajo, los datos de ventas pasarán por una transformación de formato denominada melt, una operación de pandas que convierte una tabla ancha en una tabla larga (long format). En la matriz original, cada fila representa un producto y las 1.913 columnas de días contienen las ventas observadas. Tras la transformación, cada fila pasa a representar la venta de un producto en un día concreto, lo que multiplica el número de filas, pero reduce el de columnas. El tensor resultante alcanzaría una dimensión teórica de 30.490 productos × 1.913 días = 58.327.370 observaciones.

A esta cifra cabe aplicarle una corrección. Durante la fase de ingeniería de variables (desarrollada en el apartado 5.4) se generan variables de retardo (lag features), que para cada día contienen el valor de las ventas observadas días atrás (por ejemplo, las ventas de hace 28 o 35 días). Para los primeros 35 días de cada serie estos valores no existen y se descartan mediante la función dropna, lo que reduce el tensor efectivo a 57.260.220 observaciones, según refleja la salida de la celda de construcción del dataset maestro en el notebook adjunto. Esta dimensionalidad multiplica la presión sobre la memoria disponible y justifica la optimización previa de los ficheros de origen.

La estrategia adoptada para mitigar este problema es el downcasting numérico, técnica recomendada en la guía oficial de pandas para el escalado de conjuntos de datos de gran volumen (pandas development team, 2024). La idea consiste en reasignar el tipo de dato de cada columna al entero o flotante de menor tamaño capaz de representar su rango de valores, sin pérdida de información. Por ejemplo, una columna que contiene únicamente valores entre 0 y 100 no necesita los 64 bits que pandas asigna por defecto: basta con 8 bits para almacenarla sin alterar ningún valor, lo que divide su consumo de memoria por ocho. La propia biblioteca expone esta operación a través de la función pandas.to_numeric con el parámetro downcast, que automatiza la selección del tipo mínimo entre las opciones int8, uint8 y float32 (pandas development team, 2024).

La implementación adoptada en este trabajo no recurre directamente a pandas.to_numeric, sino a una función personalizada denominada reduce_mem_usage, por dos motivos: en primer lugar, la función nativa opera columna a columna, mientras que aquí conviene una rutina que procese el DataFrame completo en una sola pasada; en segundo lugar, la versión personalizada integra simultáneamente la conversión de columnas de texto al tipo category, optimización que pandas.to_numeric no contempla. La función empleada es una adaptación de la rutina popularizada en la comunidad Kaggle por Groen (2017). Su lógica recorre cada columna del DataFrame, calcula el valor mínimo y el máximo mediante pandas.Series.min() y pandas.Series.max(), y consulta los límites exactos de cada tipo numérico a través de las funciones numpy.iinfo y numpy.finfo (Harris et al., 2020). Por ejemplo, una columna cuyo rango observado quede contenido en el intervalo [−32.768, +32.767] (los límites del tipo int16) se convierte a este tipo, lo que reduce su consumo de memoria a la cuarta parte respecto al int64 por defecto. Conviene señalar que la implementación adoptada utiliza comparaciones estrictas (> para el mínimo y < para el máximo), lo que añade un pequeño margen de seguridad frente a errores de borde a costa de descartar los dos valores extremos del rango.

La segunda optimización aplicada es la conversión de las variables identificadoras de baja cardinalidad (item_id, dept_id, cat_id, store_id y state_id) al tipo category de pandas. Este tipo de dato sustituye internamente cada cadena de texto por un código numérico y mantiene un diccionario aparte con las etiquetas originales (pandas development team, 2024). El término baja cardinalidad (low-cardinality data) hace referencia a columnas que contienen un número reducido de valores únicos respecto al total de filas, situación en la que esta conversión resulta especialmente eficaz (pandas development team, 2024). En el fichero de precios, por ejemplo, la columna item_id contiene únicamente 3.049 valores únicos repartidos entre 6.841.121 filas; esta proporción explica que la reducción observada en dicho fichero (del 95 %) sea la más elevada de los tres.

La aplicación combinada de ambas técnicas sobre los tres ficheros del dataset arroja los resultados recogidos en la Tabla 2. La reducción oscila entre el 44 % del calendario (un fichero ya naturalmente compacto) y el 95 % del archivo de precios, donde la conversión a category produce la mayor ganancia. El consumo total del dataset desciende de 1.309,5 MB a 143,6 MB, lo que supone una reducción agregada del 89 %.

Fichero	Memoria inicial (MB)	Memoria final (MB)	Reducción (%)
sales_train_validation.csv	455,7	97,3	79
calendar.csv	0,7	0,4	44
sell_prices.csv	853,1	45,9	95
Total	**1.309,5**	**143,6**	**89**

Tabla 2: Reducción de memoria por fichero tras la aplicación combinada del downcasting numérico y la conversión al tipo category. Datos obtenidos de la sección «Optimización de Memoria» del notebook adjunto.

4.3 Métricas de Evaluación

Conviene señalar que todas las métricas reportadas en este trabajo se calculan sobre el mismo horizonte de validación de 28 días, garantizando la comparabilidad interna de los resultados. Además, aunque el RMSE permite una interpretación directa en unidades vendidas, sus conclusiones deben interpretarse siempre dentro del mismo nivel de agregación, evitando comparaciones directas entre errores calculados a escala SKU, tienda o total corporativo.

La elección de la métrica de evaluación condiciona la interpretación de los resultados tanto como la elección del modelo. En el ámbito del pronóstico de series temporales, no existe una métrica universalmente superior: cada una pondera de manera distinta los errores y se ajusta mejor o peor a las características del problema (Hyndman & Koehler, 2006). En el caso del dataset M5, dos rasgos del fenómeno condicionan esta elección: la presencia masiva de ceros (el 68 % de las observaciones a nivel de SKU individual) y la heterogeneidad de escala entre las distintas series. A continuación, se justifica la métrica seleccionada y se descartan otras alternativas frecuentes en la literatura.

4.3.1 Métrica seleccionada: RMSE (Root Mean Squared Error)

El error cuadrático medio (*Root Mean Squared Error*, RMSE) se define como la raíz cuadrada del promedio de las diferencias al cuadrado entre los valores reales y los predichos por el modelo:

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

donde y_t es el valor real observado en el instante t, $\hat{y}_t$ es la predicción del modelo para ese mismo instante y n es el número total de observaciones evaluadas. Al elevar al cuadrado los errores antes de promediarlos, esta métrica penaliza con mayor severidad las desviaciones grandes que las pequeñas; además, el RMSE conserva las mismas

unidades que la variable predicha (unidades vendidas por día), lo que facilita su interpretación directa en términos de negocio (Hyndman & Athanasopoulos, 2021).
La utilización del RMSE en este trabajo responde a tres razones. En primer lugar, al elevar al cuadrado los errores antes de promediarlos, esta métrica penaliza con mayor severidad las desviaciones grandes que las pequeñas, propiedad relevante en el contexto retail, donde un error de diez unidades en un producto de alta rotación supone un impacto operativo muy superior al de diez errores de una unidad. En segundo lugar, el RMSE conserva las mismas unidades que la variable predicha (unidades vendidas por día), lo que facilita su interpretación directa en términos de negocio. En tercer lugar, el cálculo basado en errores cuadráticos resulta coherente con el objetivo de la competición M5, donde la propia organización justifica el uso de medidas cuadráticas frente a medidas absolutas porque las primeras se optimizan para la media y las segundas para la mediana, y el objetivo del problema es pronosticar la demanda media, no la mediana (Makridakis et al., 2022).

4.3.2 Descarte del MAPE (Mean Absolute Percentage Error)

El error porcentual absoluto medio (*Mean Absolute Percentage Error*, MAPE) es una de las métricas más extendidas en el pronóstico de demanda por su interpretabilidad inmediata en términos porcentuales. Su fórmula, sin embargo, divide el error absoluto entre el valor real:

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

donde y_t es el valor real observado en el instante t, $\hat{y}_t$ es la predicción del modelo para ese mismo instante y n es el número total de observaciones evaluadas. Cuando $y_t = 0$ el denominador se anula y la métrica produce un valor matemáticamente indefinido (Hyndman & Koehler, 2006), lo que invalida el uso del MAPE en contextos de demanda intermitente como el del dataset M5.
Cuando el valor real es cero (situación que se da en el 68 % de las observaciones del dataset M5) el denominador se anula y la métrica produce un valor matemáticamente indefinido (Hyndman & Athanasopoulos, 2021). Esta limitación resulta especialmente grave en contextos de demanda intermitente como el que aquí se analiza, donde la abundancia de ceros es una característica estructural del fenómeno y no una anomalía puntual. Por este motivo, el MAPE queda descartado como métrica de evaluación.

4.3.3 Métrica oficial de la competición M5: WRMSSE

La competición M5 establece como métrica oficial el error cuadrático medio escalado ponderado (Weighted Root Mean Squared Scaled Error, WRMSSE), introducida por la propia organización como variante del MASE propuesto por Hyndman y Koehler (2006). El WRMSSE se construye sobre el RMSSE, que escala el error cuadrático de cada serie por su variabilidad histórica, y a continuación pondera el RMSSE de cada serie según su importancia en la jerarquía de ventas (Makridakis et al., 2022):

$$\text{WRMSSE} = \sum_{i=1}^{42\,840} w_i \cdot \text{RMSSE}_i$$

$$\text{RMSSE}_i = \sqrt{\frac{\frac{1}{h}\sum_{t=n+1}^{n+h}(y_{i,t} - \hat{y}_{i,t})^2}{\frac{1}{n-1}\sum_{t=2}^{n}(y_{i,t} - y_{i,t-1})^2}}$$

donde w_i es el peso asignado a la serie i (calculado a partir del volumen de ventas en dólares de los últimos 28 días del periodo de entrenamiento), RMSSE_i es el error cuadrático medio escalado de la serie i, $y_{i,t}$ es el valor real de la serie i en el instante t y $\hat{y}_{i,t}$ su predicción; n es el número de observaciones del conjunto de entrenamiento y h el horizonte de predicción. El denominador del RMSSE corresponde al error cuadrático medio del método Naive sobre el conjunto de entrenamiento, que actúa como factor de escalado. El sumatorio externo se extiende sobre las 42 840 series correspondientes a los doce niveles jerárquicos del dataset M5. Esta métrica resulta independiente de la escala de cada serie y otorga mayor peso a los productos económicamente más relevantes (Makridakis et al., 2022).

Esta métrica ofrece dos ventajas frente al RMSE simple: resulta independiente de la escala de cada serie (lo que permite comparar productos de muy distinto volumen) y otorga mayor peso a los SKUs económicamente más relevantes. Sin embargo, su cálculo requiere información detallada sobre los precios de cada producto en los últimos 28 días previos a cada predicción, así como la reconstrucción de las 42.840 series correspondientes a los doce niveles jerárquicos de agregación del dataset. La implementación rigurosa del WRMSSE excede el alcance del presente trabajo y queda planteada como línea de continuación en el apartado 5.5.4.

Cabe señalar que la elección del RMSE como métrica principal no invalida los resultados obtenidos. La comparación entre modelos se realiza siempre sobre las mismas series y bajo las mismas condiciones experimentales, por lo que las diferencias relativas observadas entre algoritmos resultan robustas con independencia de la métrica utilizada. La ventaja del WRMSSE radica en permitir la comparación directa con los resultados publicados por los participantes oficiales de la competición, no en alterar la jerarquía de rendimiento entre los modelos evaluados en este trabajo.

4.4 Entorno Tecnológico

El desarrollo experimental de este trabajo se implementa en Python 3.12, lenguaje consolidado como referencia en el ámbito de la ciencia de datos (Van Rossum & Drake, 2009) por la madurez y amplitud de su ecosistema científico (Van Rossum & Drake, 2009). La ejecución se realiza sobre la plataforma Kaggle Kernels (Kaggle, 2020), entorno que proporciona acceso directo al dataset M5 alojado en el repositorio oficial de la competición y dispone de los recursos de memoria y cómputo necesarios para procesar las cerca de 57 millones de observaciones del tensor maestro descrito en el apartado 4.2. La fijación explícita de la semilla aleatoria (SEED = 42) en la celda de configuración garantiza la reproducibilidad estricta de los resultados a lo largo de todo el pipeline.

A continuación, se describen las principales bibliotecas utilizadas y la función específica que cada una desempeña en el flujo de trabajo.

Pandas y NumPy. Constituyen el núcleo del procesamiento de datos del proyecto. La biblioteca pandas se emplea para la manipulación, limpieza y transformación del dataset M5 (McKinney, 2010; Pandas Development Team, 2024) (operaciones de carga, fusión (merge), reorganización (melt) y agrupación), mientras que NumPy proporciona las estructuras de datos numéricas subyacentes y las operaciones vectorizadas que soportan el cálculo eficiente (Harris et al., 2020) (McKinney, 2010; Harris et al., 2020). Ambas bibliotecas resultan imprescindibles para la aplicación de las técnicas de optimización de memoria descritas en el apartado 4.2.

XGBoost. Implementación de referencia del algoritmo *Gradient Boosting* sobre árboles de decisión, desarrollada por Chen y Guestrin (2016). En este trabajo se utiliza en modo de regresión con función de pérdida Tweedie (parámetro objective='reg:tweedie'), elegida por su capacidad para modelar variables con exceso de ceros, característica estructural del dataset M5. La biblioteca incorpora optimizaciones específicas [construcción de árboles basada en histogramas, paralelización en CPU, parada temprana (*early stopping*)] que la hacen especialmente adecuada para el volumen de datos manejado.

Prophet. Biblioteca de pronóstico desarrollada por Taylor y Letham (2018) en el equipo de ciencia de datos de Meta. Su diseño se basa en un modelo aditivo descomponible que separa la tendencia, la estacionalidad y los efectos de eventos especiales, lo que permite trabajar con series temporales sin necesidad de garantizar la estacionariedad ni especificar manualmente los órdenes autorregresivos. En este trabajo se emplea sobre la serie agregada de demanda, configurando las componentes de estacionalidad semanal y anual y desactivando la diaria, decisión justificada en el apartado 5.3.

Statsmodels. Biblioteca de referencia para modelado estadístico en Python (Seabold & Perktold, 2010). Se utiliza en la fase de modelado clásico para implementar los modelos ARIMA y SARIMA, así como las pruebas de estacionariedad [*Augmented Dickey-Fuller* (ADF) y *Kwiatkowski-Phillips-Schmidt-Shin* (KPSS)] y la selección de parámetros mediante criterios de información (AIC y BIC).

Scikit-learn. Biblioteca general de aprendizaje automático en Python (Pedregosa et al., 2011). En este trabajo se emplea exclusivamente la función *mean_squared_error* del módulo *sklearn.metrics*, sobre la que se aplica la raíz cuadrada para obtener el RMSE definido en el apartado 4.3. La elección de scikit-learn frente a una implementación propia responde a la trazabilidad de los resultados: la función está ampliamente validada por la comunidad científica y permite reproducir el cálculo de forma idéntica en cualquier entorno.

Matplotlib. Biblioteca estándar de visualización en el ecosistema científico de Python (Hunter, 2007). Se utiliza para la representación de series temporales, descomposiciones estacionales, diagnósticos de residuos y comparativas de predicciones entre modelos. Los módulos matplotlib.pyplot y matplotlib.patches se emplean directamente sin envoltorios adicionales.

Gc y warnings. Bibliotecas estándar de Python que complementan el flujo de trabajo experimental. El módulo *gc* se invoca explícitamente tras la liberación de DataFrames intermedios mediante el operador *del*, garantizando la devolución inmediata de memoria

al sistema operativo, tal como se detalla en el apartado 4.2. El módulo *warnings* se configura para suprimir avisos no críticos generados por las distintas bibliotecas durante la ejecución, evitando que el registro de salida se sature con información irrelevante para el análisis.

El entorno de desarrollo es Kaggle Kernels, plataforma basada en Jupyter Notebook que combina la ejecución de código Python, la visualización inmediata de resultados y la incorporación de explicaciones narrativas en celdas Markdown, conformando así un documento reproducible y autodocumentado (Kluyver et al., 2016). El código fuente completo del experimento se encuentra disponible en el repositorio público del proyecto, cuyo enlace figura en el apartado de Anexos.

5. Desarrollo y Resultados

5.1 Análisis Exploratorio de Datos (EDA)

El Análisis Exploratorio de Datos (Exploratory Data Analysis, EDA) constituye la primera fase del desarrollo experimental y tiene como objetivo comprender la naturaleza estadística del dataset M5 antes de aplicar cualquier modelo predictivo (Hyndman & Athanasopoulos, 2021). Conviene subrayar que esta fase no debe considerarse un paso preliminar accesorio: cada hallazgo obtenido en este apartado justifica de forma directa una decisión técnica adoptada en las fases posteriores del trabajo (Hyndman & Athanasopoulos, 2021). En las páginas siguientes se identifican tres hallazgos estructurales (tendencia, estacionalidad e intermitencia) y se exponen las consecuencias metodológicas que cada uno de ellos tiene sobre el diseño experimental. El análisis se desarrolla a dos niveles de agregación. El nivel corporativo (obtenido como la suma de las ventas diarias de los 30.490 productos a lo largo de las diez tiendas) permite identificar patrones estructurales que quedan ocultos por el ruido de la intermitencia a nivel granular. El nivel de SKU individual (cada producto en una tienda y un día concretos) refleja la naturaleza real del fenómeno que el modelo deberá predecir en producción.

Hallazgo 1: Tendencia creciente con perturbaciones estacionales. El primer análisis del EDA consiste en agregar las ventas diarias de los 30.490 productos a lo largo de las diez tiendas del dataset, obteniendo así una visión macro del comportamiento de la demanda a nivel corporativo. Esta perspectiva del nivel 1 en la jerarquía permite identificar patrones estructurales que quedan ocultos a nivel de SKU individual debido al ruido de la intermitencia (Makridakis et al., 2022). La serie resultante cubre los 1.913 días del periodo de entrenamiento y validación, comprendidos entre el 29 de enero de 2011 y el 24 de abril de 2016.

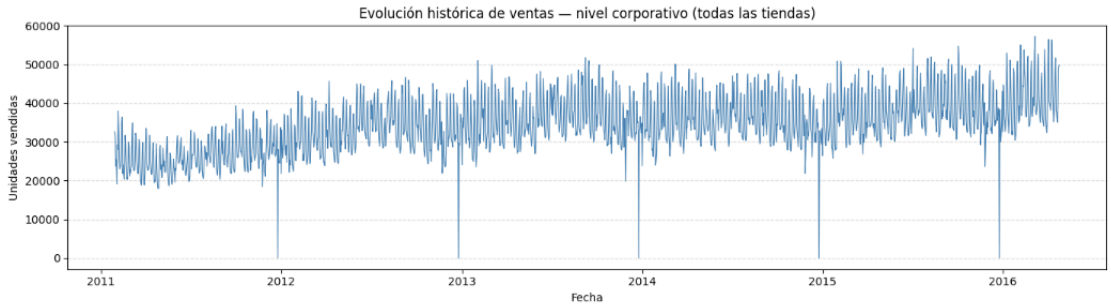

Figura 8: Evolución histórica del volumen de ventas a nivel corporativo

La inspección visual de la serie, práctica habitual en el diagnóstico inicial de series temporales, revela dos patrones estructurales (Hyndman & Athanasopoulos, 2021). El primero es una tendencia de crecimiento sostenido a lo largo del periodo: el volumen de ventas diarias agregadas asciende progresivamente entre el inicio y el final del dataset, comportamiento coherente con la expansión comercial de Walmart en los estados de California, Texas y Wisconsin durante el quinquenio analizado. El segundo es la presencia de caídas abruptas y puntuales de la demanda con periodicidad aproximadamente anual, en las que el volumen diario se reduce de forma drástica respecto a los días contiguos. Estas perturbaciones no afectan a la dinámica estructural de la serie (que se mantiene globalmente ascendente) pero constituyen eventos singulares que cualquier modelo predictivo deberá ser capaz de anticipar para no generar errores sistemáticos en los días afectados.

Desde el punto de vista de la ingeniería de características, este hallazgo tiene una implicación directa sobre el diseño del modelo XGBoost: las variables year y month se incluyen en el vector de features para que el modelo pueda distinguir el comportamiento de un día concreto en función del año y del mes en que se encuentra, capturando así tanto el crecimiento tendencial como los efectos asociados al calendario.

Cabe observar, además, que la amplitud de las fluctuaciones que se superponen sobre la tendencia parece crecer de forma proporcional al nivel de la serie, lo que sugiere que la descomposición adecuada para esta serie es de tipo multiplicativo y no aditivo (Hyndman & Athanasopoulos, 2021). Esta observación condiciona la configuración del modelo Prophet en el apartado 5.3, donde se opta por el modo multiplicativo precisamente por esta razón.

Hallazgo 2: Estacionalidad semanal dominante. El segundo análisis aplica la técnica de descomposición STL (Seasonal and Trend decomposition using Loess) sobre la serie temporal agregada. STL es un algoritmo iterativo desarrollado por Cleveland et al. (1990) que separa una serie temporal en tres componentes: tendencia, estacionalidad y residuo. Su principal ventaja respecto a métodos clásicos como la descomposición por medias móviles reside en su robustez ante valores atípicos (outliers): al estar basado en regresión local ponderada (Locally Estimated Scatterplot Smoothing, LOESS), los eventos anómalos (como los cierres de Navidad) no contaminan la estimación de los componentes estructurales (Cleveland et al., 1990).

La descomposición se configura con un período estacional de $m = 7$, parámetro que indica al algoritmo cada cuántas observaciones consecutivas se espera que se repita el patrón estacional. Dado que el dataset contiene observaciones diarias y la hipótesis de partida es la existencia de un ciclo semanal, el valor adecuado es $m = 7$ (siete días por semana).

Los resultados de la descomposición confirman la hipótesis de partida. El componente de tendencia reproduce el crecimiento sostenido identificado en el Hallazgo 1, esta vez aislado del ruido estacional. El componente estacional revela un patrón cíclico de exactamente siete días, regular y estable a lo largo de todo el período. Finalmente, el componente residual concentra los eventos que no quedan explicados por la tendencia ni por la estacionalidad: los cierres de Navidad aparecen aquí como picos negativos extremos, lo que confirma su naturaleza de perturbaciones puntuales y no estructurales.

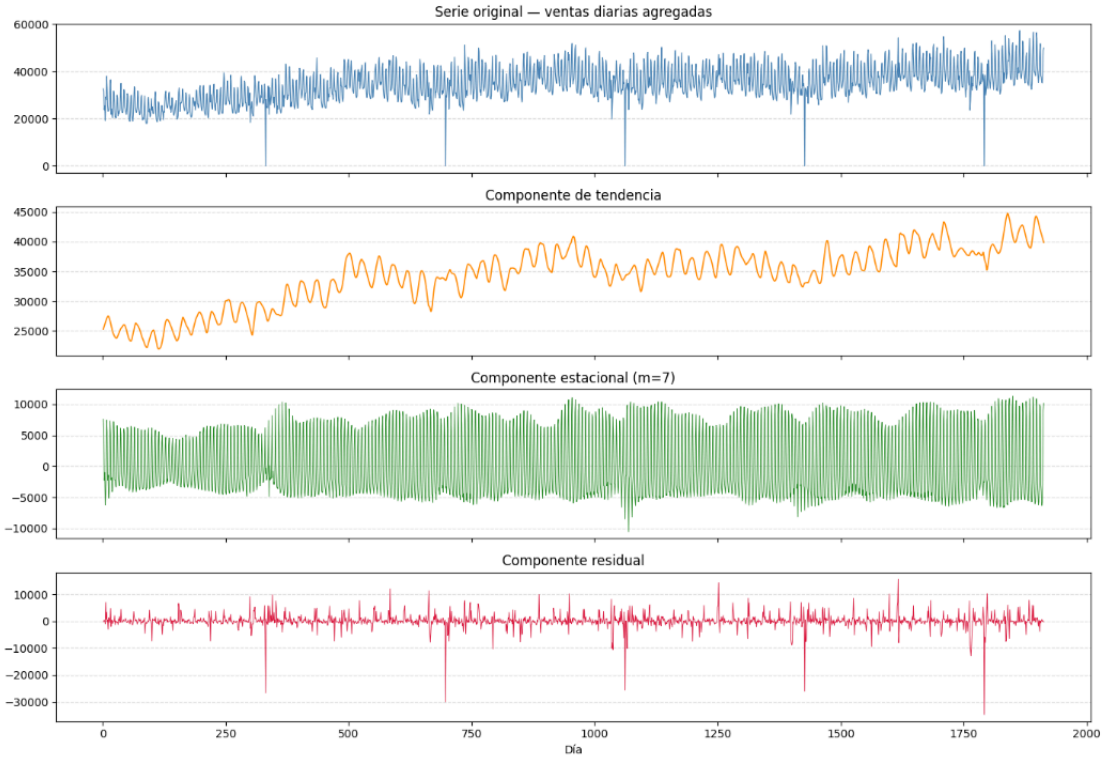

Figura 9: Resultado de la descomposición STL: paneles de tendencia, estacionalidad (m=7) y residuo.

Para verificar visualmente el patrón cíclico identificado por la descomposición, la Figura 11 amplía el componente estacional a un horizonte de tres semanas consecutivas con los días de la semana etiquetados explícitamente. La representación de detalle confirma con claridad las dos afirmaciones que la descomposición completa solo permitía intuir. En primer lugar, las tres ondas son prácticamente idénticas entre sí, lo que demuestra que el patrón semanal se repite con regularidad y amplitud estable a lo largo del dataset. En segundo lugar, dentro de cada semana se observa una estructura recurrente: los lunes y martes registran los valores más bajos del ciclo (entre 4.000 y 5.000 unidades por debajo de la media local), la actividad se mantiene moderada hasta el jueves, asciende de forma pronunciada el viernes y alcanza su máximo el sábado, con desviaciones positivas superiores a las 8.000 unidades. El domingo desciende ligeramente respecto al sábado, pero permanece por encima de la media semanal.

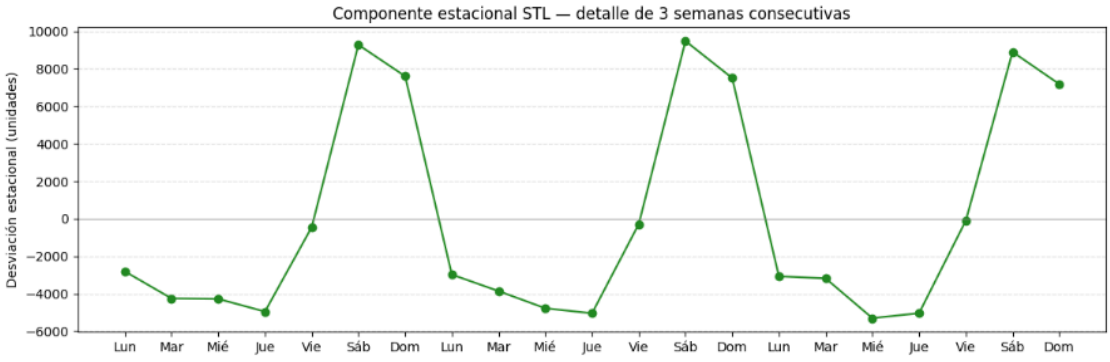

Figura 10: Componente estacional STL: detalle de tres semanas consecutivas con etiquetado por día de la semana

La cuantificación del efecto del día de la semana en unidades absolutas confirma estas observaciones desde un ángulo complementario. La Figura 12 muestra la venta media por SKU y día de la semana, calculada sobre el vector completo de 58.327.370 observaciones

del dataset. Sábado y domingo concentran la mayor actividad de compra, con medias prácticamente equivalentes de 1,36 y 1,35 unidades por SKU y día respectivamente. A continuación, se sitúa el viernes (1,12 unidades), que actúa como día de transición hacia el fin de semana. Los días laborables del centro de la semana (martes, miércoles y jueves) presentan los valores más bajos, en torno a 0,98–1,00 unidades por SKU, mientras que el lunes (1,07) recupera ligeramente respecto a este mínimo, posiblemente por efecto del arrastre del domingo. La diferencia entre el sábado y el miércoles representa un incremento del 39 % en la venta media, magnitud que refuerza la relevancia operativa del ciclo semanal para la gestión de inventarios. Conviene observar que esta cuantificación, obtenida directamente como media aritmética sobre los datos originales, resulta coherente con el componente estacional estimado por STL, lo que constituye una validación independiente del patrón identificado mediante el algoritmo de descomposición.

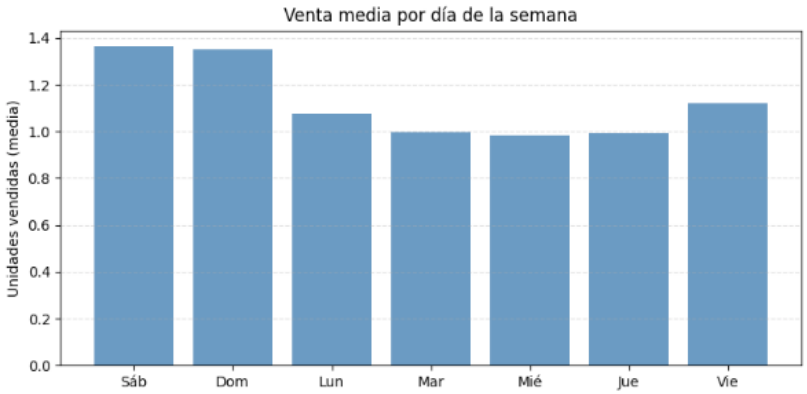

Figura 11: Venta media por SKU y día de la semana.

La implicación metodológica de este hallazgo es directa. La existencia de un ciclo semanal estable justifica que las variables de retardo del modelo XGBoost se construyan sobre múltiplos de siete días: lag_28 (cuatro semanas atrás) y lag_35 (cinco semanas atrás), ambos respetando la restricción operativa del horizonte de predicción de 28 días impuesta por la competición M5. La elección de retardos en posiciones múltiplos del ciclo semanal permite al modelo comparar las ventas de un día concreto con las del mismo día de la semana en períodos anteriores, capturando así el patrón identificado por la descomposición STL sin violar la condición de causalidad temporal. El uso de retardos más cortos como lag_7 o lag_14 quedaría descartado en este contexto, dado que ambos requerirían información sobre días que el modelo no puede haber observado en el momento de la predicción (constituirían lo que la literatura denomina data leakage y que se desarrolla en detalle en el apartado 5.4.6).

Paralelamente, las medias móviles rmean_28_7 y rmean_28_28 (promedios sobre ventanas de siete y veintiocho días aplicados al retardo de 28) refuerzan la captura del ciclo semanal al suavizar las fluctuaciones de corto plazo. La evaluación posterior de la importancia de variables del modelo (feature importance, apartado 5.4.3) confirma empíricamente que estas variables construidas sobre el ciclo semanal figuran entre las que más contribuyen a la predicción.

Hallazgo 3: Intermitencia extrema. El tercer y más determinante hallazgo del EDA se obtiene del análisis de la distribución estadística de las ventas diarias a nivel de SKU

individual. Para construir esta distribución se aplana la matriz completa de ventas (30.490 productos × 1.913 días) en un vector de 58.327.370 observaciones, sobre el que se calcula la frecuencia relativa de cada valor.

El resultado es contundente: el 68,2 % de las observaciones presentan un valor exactamente igual a cero. Es decir, de cada cien registros del dataset, 68 corresponden a días en los que un producto concreto en una tienda concreta no registró ninguna venta. Los valores de una y dos unidades representan el 13,3 % y el 6,7 % respectivamente, y las ventas iguales o superiores a cinco unidades en un solo día concentran únicamente el 6 % de las observaciones, lo que confirma su carácter estadísticamente excepcional a este nivel de granularidad.

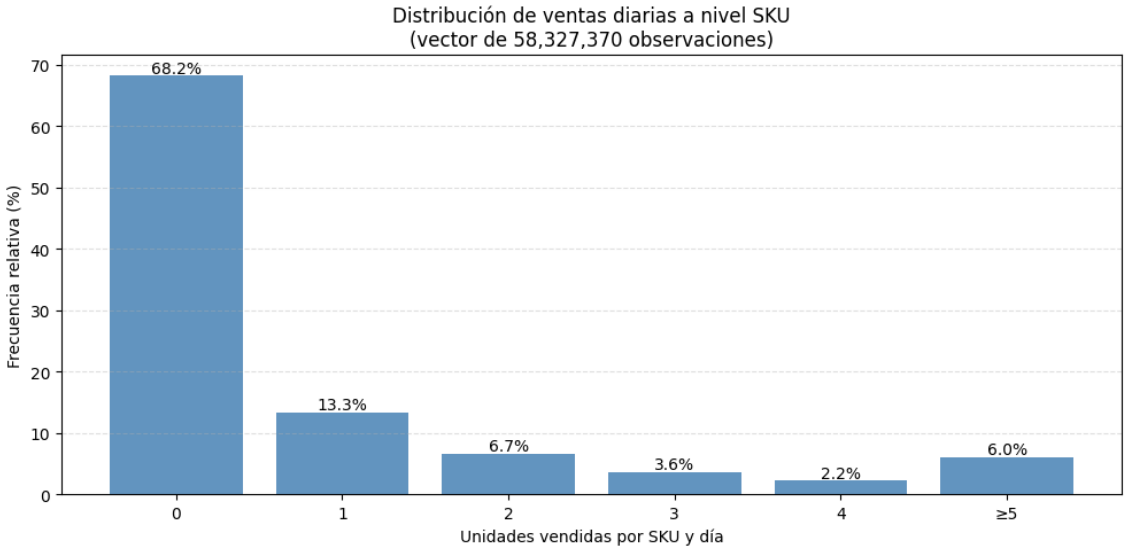

Figura 12: Distribución de ventas diarias a nivel SKU: frecuencia relativa para valores 0, 1, 2, 3, 4, ≥5

Conviene precisar qué significa y qué no significa este 68,2 %. La cifra no implica que el 68 % de los productos de Walmart no se vendan, dado que a nivel de tienda agregada prácticamente todos los productos registran ventas diarias. Lo que refleja es la naturaleza intrínsecamente discontinua de la demanda cuando se observa al nivel más granular posible: un artículo concreto (por ejemplo, un juego de mesa específico de la categoría HOBBIES) en una tienda concreta de California puede registrar ventas tres días a la semana y permanecer sin movimiento los otros cuatro. Este fenómeno, denominado en la literatura especializada demanda intermitente (intermittent demand), constituye una característica inherente del retail a nivel de SKU (Croston, 1972; Syntetos & Boylan, 2005) y ha sido ampliamente documentado como uno de los principales desafíos del pronóstico de demanda (Croston, 1972; Syntetos & Boylan, 2005).

Desde el punto de vista estadístico, una serie con un 68,2 % de ceros no sigue una distribución normal: presenta una masa de probabilidad concentrada en el valor cero, seguida de una cola derecha con valores positivos dispersos (Syntetos & Boylan, 2005; Smyth & Jørgensen, 2002). Esta forma técnicamente denominada distribución con inflación de ceros invalida varios de los supuestos sobre los que se construyen los modelos estadísticos clásicos del tipo ARIMA y tiene tres consecuencias directas sobre el diseño del experimento que se desarrollan a lo largo del trabajo.

En primer lugar, la inflación de ceros invalida la asunción de estacionariedad sobre la que se construyen los modelos ARIMA y, por extensión, SARIMA: una serie con esta

estructura no puede transformarse en estacionaria mediante diferenciación ordinaria, dado que la concentración de masa en el cero no constituye una forma de no estacionariedad corregible por esta vía (Box & Jenkins, 1970). Este argumento, junto con los de escalabilidad desarrollados en el Marco Teórico, justifica que los modelos clásicos se evalúen únicamente sobre series agregadas y no a nivel de SKU, decisión que se desarrolla en el apartado 5.2.

En segundo lugar, la presencia masiva de ceros invalida el uso del MAPE (Mean Absolute Percentage Error) como métrica de evaluación, dado que su fórmula divide el error por el valor real y la división por cero produce un valor indefinido. Esta es la razón por la que en este trabajo se utiliza el RMSE como métrica principal, según se justifica en el apartado 4.3.

En tercer lugar, la inflación de ceros condiciona la elección de la función de pérdida del modelo XGBoost. La función por defecto en problemas de regresión (el error cuadrático medio (MSE)) asume implícitamente que los errores siguen una distribución normal simétrica, supuesto incompatible con la distribución observada. Por este motivo se opta por la función de pérdida Tweedie, cuya justificación detallada se presenta en el apartado 5.4.5.

5.2 Preparación de los datos para cada modelo

En función de los resultados del análisis exploratorio descrito en el apartado anterior, los datos requeridos para la aplicación de cada modelo son los siguientes:

- Modelos base

    - Naïve Estacional: se utiliza la serie agregada de la tienda CA_1, construida sumando las ventas diarias de los 3.049 productos de esa tienda. No requiere entrenamiento ni estima parámetros: aplica la heurística de copiar como predicción del día t el valor real observado 28 días antes, alineado con el horizonte oficial de la competición (Makridakis, Spiliotis & Assimakopoulos, 2022).
    - ARIMA: se utiliza la misma serie agregada de CA_1. La preparación específica consiste en la búsqueda automática del orden (p, d, q) mediante el criterio de información de Akaike (AIC) sobre 18 configuraciones candidatas, resultando seleccionado la configuración ARIMA (2, 1, 2) (Box, Jenkins, Reinsel & Ljung, 2015).
    - SARIMA: se utiliza la misma serie agregada de CA_1, especificando explícitamente el ciclo estacional mediante el parámetro m = 7, coherente con la estacionalidad semanal identificada en la descomposición STL del Hallazgo 2 del EDA (Cleveland et al., 1990). La configuración SARIMA(1, 1, 1)(1, 1, 1)$_7$ se fija siguiendo el principio de parsimonia de Box y Jenkins (1976).
    - Prophet: se utiliza la misma serie agregada de CA_1, pero el modelo exige un formato de entrada específico con dos columnas (ds para las fechas e y para los valores observados). Dado que el dataset M5 no contiene fechas reales sino identificadores numéricos de día, se generan fechas sintéticas a partir del 29 de enero de 2011, día correspondiente a d_1 según el archivo calendar.csv (Taylor & Letham, 2018).

La partición temporal es idéntica para los cuatro modelos: entrenamiento sobre los días 1 a 1.913 y validación sobre los 28 días finales (1.914 a 1.941), coincidentes con el horizonte oficial de la competición M5 (Makridakis et al., 2022).

- XGBoost. A diferencia de los modelos base, para la aplicación de XGBoost resulta necesario preparar un dataset maestro en formato largo que integre toda la información disponible (ventas, calendario y precios), por el motivo de que los algoritmos de aprendizaje automático requieren una observación por fila (McKinney, 2017). Mediante la operación melt de pandas, las 30.490 filas y 1.941 columnas de día del archivo original se transforman en aproximadamente 58 millones de observaciones, sobre las que se incorporan las variables del calendario y los precios mediante dos fusiones relacionales.

A diferencia de SARIMA y Prophet, que ajustan un modelo independiente para cada serie temporal, XGBoost se utiliza como modelo global: se entrena una única vez sobre todas las series del dataset de forma conjunta (Chen & Guestrin, 2016; Montero-Manso & Hyndman, 2021) y aprende patrones comunes entre productos, tiendas, categorías y estados (Chen & Guestrin, 2016). Esta arquitectura tiene una implicación directa y fundamental: el modelo puede aprender que:

- los productos de la categoría FOODS tienen un ciclo semanal distinto al de los productos de HOBBIES,
- la tienda CA_1 vende más los sábados que la tienda TX_2,
- una subida de precio reduce la demanda de forma diferente según el departamento.

Ninguno de estos patrones cruzados es accesible para un modelo univariante que solo ve una serie a la vez (Montero-Manso & Hyndman, 2021). La contrapartida es que XGBoost no trabaja directamente con la estructura temporal de la serie: necesita que la información temporal se transforme en características (features) mediante un proceso de ingeniería previo, descrito en detalle en el apartado 5.4.1.

5.3 Resultados de los Modelos Base

Antes de aplicar cualquier algoritmo avanzado, conviene establecer un conjunto de modelos de referencia que cumpla dos funciones simultáneamente. La primera es proporcionar una cota mínima de rendimiento (baseline) sobre la que medir el valor añadido de los modelos más sofisticados: si un algoritmo complejo no supera a una heurística trivial, su coste computacional no resulta justificable (Hyndman & Athanasopoulos, 2021). La segunda es comprobar empíricamente las limitaciones de los modelos estadísticos clásicos (ARIMA y SARIMA) cuando se enfrentan a las particularidades del dataset M5 identificadas en el Hallazgo 3 del EDA. Todos los modelos de este apartado se evalúan sobre la serie agregada de la tienda CA_1, obtenida como la suma diaria de las ventas de los 3.049 productos de esa tienda. La elección de operar a nivel de tienda y no a nivel de SKU responde a una restricción técnica: tanto ARIMA como SARIMA son modelos univariantes que asumen series continuas y

regulares, supuesto incompatible con el 68,2 % de ceros observado a nivel de SKU individual (Box & Jenkins, 1970). A nivel de tienda agregada, los ceros individuales de cada producto se compensan entre sí y la serie resultante presenta valores positivos todos los días, una media estable y un ciclo semanal claramente visible. La partición temporal es idéntica para los tres modelos: se entrena sobre los primeros 1.885 días de la serie y se reservan los últimos 28 días (d_1886 a d_1913) como conjunto de validación. Esta ventana de 28 días coincide con el horizonte oficial de predicción de la competición M5 (Makridakis et al., 2022). La métrica de evaluación es el RMSE, según se justifica en el apartado 4.3.

El periodo de validación empleado en el código corresponde a los días 1914–1941. En consecuencia, cuando se menciona el modelo Naive Estacional, los días 1886–1913 deben entenderse únicamente como la ventana histórica copiada por el baseline para predecir el horizonte 1914–1941, no como el periodo evaluado.

5.3.1 Aplicación Naive Estacional

El modelo Naive Estacional constituye la referencia mínima de rendimiento del trabajo. Estrictamente hablando no es un modelo en el sentido algorítmico del término: no aprende, no estima parámetros y no incorpora ninguna variable explicativa. Se trata de una heurística directa basada en una premisa elemental: la mejor predicción de las ventas del día *t* es el valor observado exactamente 28 días antes. Formalmente:

$$\hat{y}_t = y_{t-28}$$

donde $\hat{y}_t$ es la predicción de las ventas para el día t e y_{t-28} es el valor real observado exactamente 28 días antes. El retardo de 28 días, alineado con el horizonte oficial de la competición M5, garantiza que la variable utilizada siempre esté disponible en el momento de la predicción y respeta el principio de causalidad temporal (Hyndman & Athanasopoulos, 2021; Makridakis et al., 2022).

La elección de un retardo de 28 días, en lugar de los siete que correspondería al ciclo semanal identificado en el EDA, responde a la restricción de causalidad temporal impuesta por el horizonte de la competición. En un escenario operativo real, cuando el sistema necesita pronosticar la demanda de los próximos 28 días, no dispone de las ventas de los siete días previos al primer día predicho porque aún no han ocurrido. El retardo de 28 días, en cambio, siempre está disponible en el momento de la predicción y respeta la condición de que ninguna variable utilizada contenga información del futuro (Hyndman & Athanasopoulos, 2021).

La razón por la que un modelo tan elemental merece ser evaluado de forma explícita es metodológica. En la literatura de pronóstico de series temporales constituye una práctica estándar establecer un modelo naive antes de aplicar cualquier algoritmo más complejo, dado que actúa como prueba de viabilidad: si un modelo sofisticado no consigue mejorar al naive, su complejidad adicional no aporta valor (Hyndman & Athanasopoulos, 2021). En el contexto específico de la competición M5, los organizadores incluyeron el Naive Estacional entre los modelos de referencia oficiales precisamente por este motivo (Makridakis et al., 2022).

El resultado obtenido sobre la serie de CA_1 es un RMSE de 433,53 unidades por día. Para interpretar esta magnitud conviene contextualizarla: la tienda CA_1 vende en promedio 4.016 unidades diarias considerando la suma de todos sus productos, por lo que

el error del Naive representa aproximadamente un 9,06 % del volumen medio. Este valor servirá como referencia comparativa para todos los modelos subsiguientes del trabajo.

5.3.2 Aplicación modelo ARIMA

El modelo ARIMA constituye el modelo estadístico clásico por excelencia para el pronóstico de series temporales univariantes con autocorrelación y tendencia, cuyos fundamentos quedan desarrollados en el Marco Teórico (apartado 3.X). En este apartado, el foco se traslada del modelo en sí al método empleado para seleccionar su configuración óptima, dado que la elección del orden (p, d, q) condiciona directamente la calidad predictiva del resultado.

Selección del orden mediante el criterio de información de Akaike (AIC). La determinación de los parámetros (p, d, q) de un modelo ARIMA presenta un problema práctico: existen muchas combinaciones posibles y no resulta evidente cuál de ellas funcionará mejor. Para resolver esta cuestión de forma automática se utiliza el criterio de información de Akaike (Akaike Information Criterion, AIC), formulado por Akaike (1974), que asigna a cada modelo una puntuación numérica y permite identificar el más adecuado.

La idea de fondo del AIC es sencilla. A la hora de comparar varios modelos compiten dos objetivos que tiran en direcciones opuestas. Por un lado, conviene que el modelo se ajuste bien a los datos, lo que normalmente requiere más parámetros. Por otro, un modelo con demasiados parámetros tiende a aprenderse los datos de memoria en lugar de captar el patrón general, fenómeno denominado sobreajuste (overfitting) que perjudica las predicciones sobre datos nuevos. El AIC equilibra ambos objetivos mediante una fórmula que premia el buen ajuste y penaliza la complejidad:

$$AIC = 2k - 2\ln(\hat{L})$$

donde k es el número de parámetros del modelo y $(\hat{L})$ es la verosimilitud, una medida de cómo de bien se ajusta el modelo a los datos. El primer término, penaliza la complejidad: cuantos más parámetros tiene el modelo, mayor es esta cantidad. El segundo término, disminuye cuando el modelo se ajusta mejor. La regla es directa: dentro del conjunto de modelos candidatos, se selecciona el que obtenga el AIC más bajo, porque es el que mejor equilibra ambos objetivos.

Conviene tener presente que el valor absoluto del AIC no aporta información por sí solo; lo único relevante es la comparación entre modelos ajustados sobre los mismos datos. Conviene puntualizar también que el AIC se calcula sobre los datos de entrenamiento, por lo que selecciona el mejor modelo dentro de la familia considerada, pero no garantiza que esa familia sea la más adecuada para el problema (validation).

Aplicación al modelo ARIMA sobre la serie de CA_1. La búsqueda automática del orden óptimo de ARIMA recorre todas las combinaciones posibles de (p, d, q) dentro de un rango previamente definido: $p \in \{0, 1, 2\}$, $d \in \{0, 1\}$ y $q \in \{0, 1, 2\}$. Esto da lugar a 18 configuraciones candidatas. Para cada una de ellas se ajusta el modelo sobre los 1.885 días de entrenamiento de la serie de CA_1, se calcula el AIC correspondiente y se selecciona la combinación con el valor más bajo. La configuración elegida por este procedimiento es con un AIC de 29.548,92.

Una vez seleccionado el modelo, se evalúa su rendimiento sobre los 28 días de validación. El resultado obtenido es un RMSE de 825,77 unidades por día, valor que duplica al del Naive Estacional (433,53). Que un modelo estadístico clásico, ajustado mediante un criterio matemático riguroso, obtenga peor resultado que una heurística trivial puede resultar sorprendente a primera vista, pero tiene una explicación estructural directa.

ARIMA es un modelo no estacional. Sus componentes capturan la dependencia entre días consecutivos (el valor de hoy depende del de ayer y del de anteayer) pero no incorpora ningún mecanismo para reconocer que el comportamiento de un sábado se parece al de los sábados anteriores. Cuando se aplica sobre la serie de CA_1, cuyo patrón dominante es precisamente el ciclo semanal identificado en el Hallazgo 2 del EDA, el modelo genera una predicción suavizada que ignora las oscilaciones del fin de semana. Pierde así la información más valiosa de la serie: la diferencia sistemática entre los picos del sábado y los valles del lunes.

El Naive Estacional, en cambio, captura precisamente esa información a coste cero. Su predicción consiste en copiar el valor del mismo día de la semana de cuatro semanas atrás, por lo que reproduce de forma directa el patrón semanal. La comparación entre ambos modelos cuantifica el coste de ignorar la estacionalidad: un modelo estadístico complejo y bien seleccionado pierde frente a una heurística trivial porque no incorpora la estructura dominante de los datos. La conclusión metodológica es clara: para series con estacionalidad fuerte, la extensión estacional del modelo (SARIMA, desarrollado en el siguiente apartado) no es opcional sino imprescindible.

Este episodio ilustra además una de las limitaciones del AIC señaladas anteriormente: el criterio identifica el mejor modelo dentro de su familia, pero no advierte cuando la familia entera resulta inadecuada para el problema. Por muy bajo que fuera el AIC de la configuración seleccionada, ningún modelo ARIMA sin componente estacional podría haber capturado el ciclo semanal de la serie. La elección de la familia de modelos (ARIMA frente a SARIMA, en este caso) es una decisión previa que el AIC no puede tomar por sí solo y que debe apoyarse en el conocimiento del análisis exploratorio.

5.3.3 Aplicación modelo SARIMA

El modelo SARIMA (Seasonal Autoregressive Integrated Moving Average) extiende ARIMA incorporando un segundo conjunto de componentes diseñados específicamente para capturar la estacionalidad. Su notación completa es:

$$SARIMA\ (p,d,q)\ (P,D,Q)_m$$

donde el primer paréntesis recoge los parámetros no estacionales (idénticos a los de ARIMA) y el segundo los parámetros estacionales, que operan sobre los retardos del ciclo (Hyndman & Athanasopoulos, 2021). La diferencia fundamental entre ARIMA y SARIMA reside en que el segundo incorpora un ciclo temporal explícito mediante el parámetro m, que define cada cuántas observaciones se repite el patrón estacional. Por ejemplo, en una serie con datos diarios y estacionalidad semanal:

$m = 7$: indica al modelo que el comportamiento de un lunes guarda relación directa con el de los lunes anteriores, más allá de la dependencia con los días inmediatamente previos. Los parámetros estacionales (P, D, Q) replican la misma lógica que sus homólogos no estacionales, pero aplicados a estos retardos semanales.

$$SARIMA(1,1,1)(1,1,1)^7.$$

Configuración del modelo sobre CA_1. En este trabajo, el modelo se configura como La elección de estos parámetros no responde a una búsqueda automática por AIC sino al principio de parsimonia propuesto por Box y Jenkins (1970): ante la incertidumbre sobre la configuración óptima, conviene comenzar con la estructura más simple coherente con los hallazgos del análisis exploratorio.

Una primera búsqueda automática por AIC con $m = 7$ produjo un RMSE de 4.872,73 sobre el conjunto de validación, resultado que evidencia un fenómeno habitual en la optimización automática: el AIC puede identificar una combinación de parámetros que minimiza el error sobre el conjunto de entrenamiento, pero falla al generalizar a datos nuevos porque memoriza ruido en lugar de captar el patrón estructural. Este resultado motiva el cambio de estrategia hacia una configuración fija guiada por el EDA. La interpretación de los parámetros elegidos es la siguiente:

$d = 1$: aplica una diferenciación ordinaria para eliminar la tendencia creciente identificada en el Hallazgo 1 del EDA. La diferenciación consiste en restar a cada observación la del día anterior, transformando la serie creciente en una serie centrada en torno a cero donde solo quedan las oscilaciones.

$D = 1$ con $m = 7$: aplica una diferenciación estacional sobre el ciclo semanal. En lugar de restar el valor del día anterior, resta el valor del mismo día de la semana anterior, eliminando así el patrón semanal repetitivo confirmado por la descomposición STL del Hallazgo 2.

$p = 1, q = 1, P = 1, Q = 1$: los órdenes autorregresivos y de media móvil, tanto en la parte no estacional como en la estacional, se fijan en 1. Este es el punto de partida estándar recomendado por la literatura clásica para series económicas, dado que la mayoría de las series reales no requieren más de un retardo para capturar la dependencia temporal relevante.

La configuración del ajuste incluye además dos parámetros técnicos que conviene documentar: enforce_stationarity=False y enforce_invertibility=False. Ambos desactivan restricciones matemáticas que la biblioteca statsmodels aplica por defecto para garantizar que el modelo cumpla las propiedades teóricas de estacionariedad e invertibilidad. En series del mundo real con tendencia, ciclos estacionales y eventos atípicos, activar estas restricciones puede impedir que el algoritmo de optimización encuentre una solución válida. Desactivarlas es una práctica habitual en la literatura aplicada cuando se trabaja con datos complejos, y no compromete la validez predictiva de los resultados (Hyndman & Athanasopoulos, 2021).

Resultados y comparación. El resultado obtenido sobre los 28 días de validación es un RMSE de 388,26 unidades por día, lo que supone una mejora del 18% respecto al Naive Estacional (433,53) y reduce el error de ARIMA a menos de la mitad (825,77 frente a 388,26). Esta ganancia confirma empíricamente la importancia de la componente estacional: la diferencia de aproximadamente 437,51 unidades de RMSE entre ARIMA y SARIMA cuantifica el coste de ignorar el ciclo semanal en una serie donde dicho ciclo constituye el patrón dominante.

Para interpretar correctamente la magnitud del RMSE conviene contextualizarlo. La tienda CA_1 vende en promedio 4.787 unidades diarias en el horizonte de evaluación

sumando todos sus productos, por lo que el error de SARIMA (388,26) representa aproximadamente un 8,11 % del volumen medio. Este porcentaje resulta razonable para un modelo estadístico clásico aplicado sobre una serie tan volátil como las ventas minoristas, y se sitúa dentro del rango habitual reportado en la literatura para este tipo de problemas (Fildes et al., 2022).

La comparación entre los tres modelos evaluados en este apartado revela una jerarquía clara. ARIMA pierde frente al Naive porque no captura la estacionalidad. El Naive supera a ARIMA porque reproduce el ciclo semanal a coste cero, sin ningún aprendizaje. SARIMA supera al Naive porque combina la captura del ciclo con la estimación de parámetros que refinan la predicción. La progresión ARIMA, Naive ,SARIMA cuantifica el valor de cada componente y demuestra que incorporar la estructura estacional no es opcional sino imprescindible cuando la estacionalidad domina la serie.

Limitación de escalabilidad. A pesar de su rendimiento sólido sobre la serie agregada de CA_1, SARIMA comparte con ARIMA una limitación fundamental que lo excluye como solución principal para el dataset M5 completo: es un modelo univariante que debe ajustarse de forma independiente para cada serie. Aplicarlo a los 30.490 SKUs del dataset requeriría entrenar 30.490 modelos distintos, con el coste computacional que ello implica en un entorno de hardware convencional. Además, incluso si la escalabilidad no fuera un problema, el modelo seguiría enfrentándose a la intermitencia del 68,2 % de ceros identificada en el Hallazgo 3 del EDA: a nivel de SKU individual, muchas series no presentan valores positivos suficientes para estimar de forma estable los parámetros del modelo.

La evaluación de SARIMA sobre CA_1 agregada cumple en este trabajo un propósito metodológico específico: demostrar que los modelos univariantes funcionan correctamente en el nivel de agregación para el que fueron diseñados, donde la serie es continua, la señal es limpia y la estacionalidad es claramente visible. Su limitación no es de calidad predictiva sino de viabilidad a la escala que exige el problema real. Esta demostración empírica, junto con el argumento teórico desarrollado en el Marco Teórico, justifica la transición hacia modelos globales como XGBoost, capaces de operar sobre múltiples series simultáneamente; Prophet, en este trabajo, se mantiene como modelo univariante aplicado a series agregadas, tema que se desarrolla en los apartados siguientes.

5.4 Aplicación Modelos Prophet

Lo Prophet es una biblioteca de pronóstico desarrollada por Taylor y Letham (2018) en el equipo de ciencia de datos de Meta. Su diseño responde a una necesidad práctica: crear una herramienta de pronóstico que funcione bien sobre series temporales del mundo real sin requerir que el usuario tenga conocimientos avanzados de estadística. A diferencia de SARIMA, donde la selección de parámetros (p, d, q) exige un análisis previo de las funciones de autocorrelación y un diagnóstico de estacionariedad, Prophet solo requiere especificar el tipo de estacionalidad esperada y ajusta automáticamente el resto de la configuración.

Internamente, Prophet no modela la serie temporal como una secuencia de valores autocorrelacionados (que es el enfoque de ARIMA y SARIMA) sino como la suma de componentes funcionales independientes: una función de tendencia, una o varias

funciones de estacionalidad, y un término de error. La tendencia puede ser lineal o logística y detecta automáticamente los puntos de cambio estructural en la serie, momentos en los que el ritmo de crecimiento cambia de forma significativa. La estacionalidad se modela mediante series de Fourier, técnica matemática que permite representar patrones cíclicos complejos como la suma de funciones sinusoidales de distintas frecuencias (Taylor & Letham, 2018). Esta arquitectura permite capturar simultáneamente ciclos de distinta escala temporal (semanal, anual, e incluso el efecto puntual de festivos específicos) en un único modelo integrado.

Al igual que ARIMA y SARIMA, Prophet se evalúa en este trabajo sobre la serie agregada de la tienda CA_1 por las mismas razones estructurales: es un modelo univariante que ajusta un modelo independiente por serie, lo que impide aplicarlo simultáneamente a los 30.490 SKUs del dataset en un entorno computacional convencional. A diferencia de SARIMA, Prophet incorpora una evaluación adicional sobre la serie corporativa agregada (la suma diaria de ventas de las 10 tiendas, 3 estados y 30.490 productos), por una razón metodológica específica: Prophet está diseñado para series con múltiples estacionalidades simultáneas y horizontes largos (Taylor & Letham, 2018), perfil que la serie corporativa agregada cumple plenamente. Esta evaluación adicional permite contrastar la promesa teórica del modelo con su rendimiento empírico en el contexto para el que fue concebido.

Configuración del modelo sobre CA_1. El modelo se configura activando únicamente la estacionalidad semanal (weekly_seasonality=True) y desactivando tanto la estacionalidad anual (yearly_seasonality=False) como la diaria (daily_seasonality=False). La desactivación de la estacionalidad anual responde a una restricción práctica: el horizonte de predicción es de 28 días, periodo insuficiente para que el ciclo anual aporte información predictiva relevante. La desactivación de la estacionalidad diaria es coherente con la naturaleza de los datos: las observaciones del dataset M5 ya son agregadas diarias, por lo que no existe un ciclo intradiario que capturar.

Prophet requiere un formato de entrada específico: un DataFrame con exactamente dos columnas, (ds) para las fechas y (y) para los valores observados (Taylor & Letham, 2018). Dado que el dataset M5 no utiliza fechas reales sino identificadores numéricos de día (d_1, d_2, ...), se generan fechas sintéticas a partir del 29 de enero de 2011, que corresponde al día d_1 según el archivo calendar.csv del dataset. Esta conversión es necesaria para que Prophet pueda identificar correctamente los días de la semana al modelar la estacionalidad semanal.

El entrenamiento se realiza sobre los días 1 a 1.913 del dataset y la evaluación se efectúa sobre los días 1.914 a 1.941, que constituyen el horizonte ciego de 28 días reservado en el archivo sales_train_evaluation.csv de la competición M5 (Makridakis et al., 2022). El RMSE obtenido sobre este horizonte es de 500,15 unidades por día, resultado que se sitúa por encima tanto del Naive Estacional (433,53) como de SARIMA (388,26) y de XGBoost evaluado sobre la misma serie (397,10). Prophet queda, por tanto, un 15,37 % por debajo del Naive Estacional en esta serie concreta, lo que constituye un hallazgo que conviene analizar en detalle por su aparente contradicción con la expectativa teórica.

La explicación de este resultado tiene una base estructural identificada en la literatura. Prophet está diseñado como un modelo aditivo descomponible cuyas fortalezas se

manifiestan en series con múltiples ciclos estacionales simultáneos, cambios estructurales claros en la tendencia y un horizonte de predicción largo (Taylor & Letham, 2018; Triebe et al., 2021) (Taylor & Letham, 2018). La serie agregada de CA_1, sin embargo, presenta un perfil distinto: domina un único ciclo semanal regular sobre una tendencia suave y el horizonte de evaluación es relativamente corto (28 días). En estas condiciones, un modelo que se limita a reproducir literalmente el valor de hace siete días (el Naive Estacional) captura el patrón dominante de forma directa, mientras que Prophet, al estimar una tendencia subyacente y suavizarla mediante puntos de cambio, introduce un sesgo que penaliza la precisión punto a punto.

Esta observación es consistente con los hallazgos de Hewamalage et al. (2021), que documentan empíricamente cómo los baselines triviales pueden superar a modelos sofisticados en series con estacionalidad fuerte y horizontes cortos, especialmente cuando la métrica de evaluación es sensible a errores absolutos en lugar de a patrones globales. En la misma línea, (Triebe et al. ,2021) autores de la extensión NeuralProphet señalan que la arquitectura aditiva de Prophet sacrifica precisión local a cambio de interpretabilidad y robustez frente a cambios estructurales, sin que ambas propiedades sean igualmente valiosas en todos los contextos.

Conviene contextualizar la magnitud del RMSE. La tienda CA_1 vende en promedio 4.024 unidades diarias a lo largo del periodo de entrenamiento (días 1 a 1.913), volumen que asciende a 4.787 unidades diarias en el horizonte de evaluación (días 1.914 a 1.941), reflejo de la tendencia creciente identificada en el Hallazgo 1 del EDA. Tomando como referencia la media del periodo evaluado [criterio metodológicamente coherente con el del cálculo del RMSE (Hyndman & Athanasopoulos, 2021)] el error de Prophet (500,15) representa aproximadamente un 10,45 % del volumen medio. Este porcentaje se sitúa en el límite superior del rango habitual reportado en la literatura para pronóstico de demanda en retail (Fildes et al., 2022) y confirma que, si bien Prophet captura razonablemente la dinámica global de la serie, su precisión en este caso concreto no alcanza la de modelos que explotan la autocorrelación a corto plazo.

Escalado al dataset completo a nivel agregado. A diferencia de SARIMA, que no puede aplicarse a 30.490 series independientes en un entorno computacional convencional, Prophet sí permite una evaluación adicional a nivel corporativo agregado. Para ello se construye una única serie temporal sumando las ventas diarias de los 30.490 SKUs y se ajusta un único modelo Prophet sobre esa serie agregada, activando en este caso tanto la estacionalidad semanal como la anual (yearly_seasonality=True) dado que el volumen de datos a nivel corporativo permite estimar de forma estable los ciclos anuales.

La comparación a escala completa se interpreta en dos niveles distintos: XGBoost se evalúa primero a nivel SKU frente al Naive Estacional, mientras que Prophet se evalúa únicamente sobre la serie corporativa agregada. Por tanto, las métricas SKU y las métricas agregadas no deben mezclarse como si fueran equivalentes.

El RMSE obtenido a esta escala es de 4.627,87 unidades por día, frente a las 3.726,84 del Naive Estacional aplicado sobre las 30.490 series y posteriormente agregado. Prophet queda, por tanto, un 24,18 % por debajo del Naive también en este nivel. La media de ventas diarias agregadas en el horizonte de evaluación es de 43.992 unidades, por lo que el error relativo de Prophet asciende al 10,52 %, en línea con el observado a nivel de

CA_1. Esta consistencia entre escalas refuerza la interpretación estructural: la limitación de Prophet en este dataset no es un artefacto de la serie de CA_1 sino una característica reproducible del modelo cuando se aplica a series dominadas por estacionalidad semanal con horizontes cortos.

Limitación de escalabilidad compartida con SARIMA. Prophet comparte con SARIMA y ARIMA la limitación fundamental que lo excluye como solución principal para el dataset M5 a nivel SKU: aunque puede aplicarse sobre una serie agregada, sigue siendo un modelo univariante que debe ajustarse de forma independiente para cada serie cuando se pretende operar al nivel de granularidad que exige la competición. Aplicarlo a los 30.490 SKUs del dataset requeriría entrenar 30.490 modelos distintos, con el coste computacional asociado. Además, al igual que SARIMA, el modelo seguiría enfrentándose a la intermitencia del 68,2 % de ceros identificada en el Hallazgo 3 del EDA: a nivel de SKU individual, muchas series no presentan valores positivos suficientes ni estructura temporal estable para que los componentes de tendencia y estacionalidad de Prophet puedan estimarse de forma fiable (Montero-Manso & Hyndman, 2021).

La evaluación de Prophet sobre CA_1 agregada y sobre el dataset corporativo agregado cumple en este trabajo un propósito metodológico específico: documentar empíricamente que un modelo aditivo descomponible, aplicado en el nivel de agregación para el que fue diseñado, no garantiza por sí mismo una mejora sobre baselines triviales cuando la serie está dominada por un único ciclo estacional. Esta observación, junto con el argumento teórico desarrollado en el Marco Teórico, justifica la transición hacia un modelo global capaz de operar simultáneamente sobre múltiples series, incorporar variables explicativas más allá del historial de ventas y manejar la intermitencia mediante funciones de pérdida especializadas. Este modelo es XGBoost, que se desarrolla en el apartado siguiente.

5.5 Aplicación Modelo XGBoost

XGBoost (Chen & Guestrin, 2016) representa el núcleo técnico de este trabajo y el enfoque que da respuesta al problema que los modelos anteriores no podían resolver: predecir la demanda de los 30.490 SKUs del dataset M5 simultáneamente, incorporando todas las variables explicativas disponibles y operando a la escala granular que exige la competición.

El desarrollo de este apartado se estructura en cuatro bloques. En primer lugar, se describe la construcción del dataset maestro y la ingeniería de características temporales, que constituyen la base sobre la que opera el modelo. En segundo lugar, se justifica la elección de la función de pérdida Tweedie, diseñada específicamente para manejar la intermitencia del 68,2 % de ceros identificada en el Hallazgo 3 del EDA. En tercer lugar, se documentan los hiperparámetros del modelo y la partición temporal del experimento. Finalmente se presentan los resultados obtenidos, primero sobre la serie agregada de CA_1 para permitir la comparación directa con los modelos base, y posteriormente sobre el dataset completo de 30.490 series.

5.5.1 Construcción del dataset maestro e ingeniería de características

Construcción del dataset maestro. El archivo principal sales_train_evaluation.csv contiene las ventas históricas en formato ancho: 30.490 filas (una por SKU) y 1.941 columnas de días (d_1, d_2, …, d_1941). Esta estructura resulta incompatible con los

algoritmos de aprendizaje automático, que requieren una observación por fila (McKinney, 2010). La operación melt de pandas transforma el formato ancho en formato largo: cada fila representa la venta de un producto concreto en un día concreto.

Sobre este dataset en formato largo se ejecutan dos fusiones relacionales. La primera incorpora el archivo calendar.csv, que añade variables temporales como día de la semana (wday), mes (month), año (year), y variables binarias que indican si ese día permite compras con el programa SNAP (Supplemental Nutrition Assistance Program) en cada estado (snap_CA, snap_TX, snap_WI). La segunda incorpora el archivo sell_prices.csv, que contiene el precio de venta histórico de cada producto por tienda y semana. Tras aplicar las técnicas de optimización de memoria descritas en el apartado 4.2 (Harris et al., 2020), el dataset maestro se reduce a una estructura manejable en el entorno computacional disponible.

Variables de retardo temporal (lags). Las variables de retardo capturan las ventas de días anteriores. En términos simples, un lag responde a la pregunta: ¿cuánto vendió este producto hace X días? La lógica es intuitiva: si un producto vendió 10 unidades el lunes pasado, resulta razonable esperar que el próximo lunes venda una cantidad similar, salvo que haya cambiado alguna condición externa como el precio o la época del año.

Se construyen dos variables: lag_28 (ventas de hace 28 días) y lag_35 (ventas de hace 35 días). Ambos son múltiplos de 7, lo que significa que capturan las ventas del mismo día de la semana en semanas anteriores: lag_28 apunta al mismo día hace cuatro semanas, y lag_35 al mismo día hace cinco semanas. Esta propiedad resulta crítica porque el Hallazgo 2 del EDA demostró que el ciclo semanal es el patrón dominante del dataset. Al usar múltiplos de 7 se garantiza que el modelo compara días equivalentes, capturando el patrón estacional sin mezclar días de distinta naturaleza (Hyndman & Athanasopoulos, 2021).

La elección de retardos de 28 y 35 días (en lugar de retardos más cortos como 1 o 7) responde a una restricción estructural del problema: el horizonte de predicción de la competición M5 es de 28 días, por lo que cualquier retardo menor a 28 no estaría disponible en el momento real de la predicción. Utilizar lag_7 para predecir el día 1.914, por ejemplo, exigiría conocer las ventas del día 1.907; pero predecir el día 1.941 (último del horizonte) requeriría conocer las del día 1.934, valor que en el contexto operativo real no está disponible. La restricción a retardos iguales o superiores al horizonte garantiza que el modelo es operativamente viable y previene la data leakage, uno de los errores metodológicos más frecuentes en el pronóstico de series temporales (Bergmeir, Hyndman & Koo, 2018).

Medias móviles (rolling mean features). Junto a los retardos puntuales se construyen dos medias móviles que capturan la tendencia local del producto sin el ruido propio de un día concreto. rmean_28_7 es la media de las siete observaciones anteriores con origen en lag_28 (es decir, la media de los días de hace 28 a 34); rmean_28_28 es la media de las veintiocho observaciones anteriores con origen en lag_28 (la media del mes anterior al horizonte de predicción). El cálculo sobre lag_28 y no sobre sales directamente es crítico para evitar la data leakage: usar ventas más recientes que en producción real no estarían disponibles en el momento de la predicción invalidaría la evaluación del modelo (Bergmeir, Hyndman & Koo, 2018).

Variables categóricas y temporales adicionales. El modelo entrenado sobre el dataset completo incorpora un total de 16 features: las dos variables de retardo (lag_28, lag_35), las dos medias móviles (rmean_28_7, rmean_28_28), las cinco variables categóricas que identifican la jerarquía del producto (item_id, dept_id, cat_id, store_id, state_id), tres variables temporales (wday, month, year), las tres variables binarias del programa SNAP (snap_CA, snap_TX, snap_WI) y el precio de venta semanal (sell_price). El tratamiento nativo de variables categóricas de XGBoost mediante el parámetro enable_categorical=True permite incorporar esta información sin necesidad de one-hot encoding (XGBoost Developers, 2024), evitando la explosión dimensional que un dataset con 30.490 SKUs distintos produciría con codificaciones binarias clásicas.

5.5.2 Abordaje de la Intermitencia: Función de Pérdida Tweedie

Una vez construido el dataset maestro con todas las variables explicativas y los features de ingeniería temporal, el siguiente paso es definir cómo XGBoost va a aprender de esos datos. Esta decisión se articula en torno a tres elementos: la función de pérdida, la partición temporal y los hiperparámetros del modelo.

Durante el entrenamiento, XGBoost construye árboles de decisión secuencialmente: cada árbol nuevo intenta corregir los errores del anterior. Para saber qué errores corregir necesita una función de pérdida, fórmula matemática que mide la diferencia entre lo que predice y lo que realmente ocurrió. La elección de esta función no es trivial porque define implícitamente qué tipo de distribución estadística asume el modelo para los datos (Chen & Guestrin, 2016).

La función de pérdida por defecto en problemas de regresión es el error cuadrático medio (MSE), que asume que los errores siguen una distribución normal simétrica. Este enfoque es razonable para muchos problemas, pero incompatible con los datos del dataset M5: una distribución con el 68 % de valores exactamente iguales a cero no puede aproximarse mediante una campana de Gauss. Aplicar MSE en este contexto conduce a que el modelo prediga sistemáticamente valores positivos pequeños incluso cuando la respuesta correcta es cero, sesgo documentado de forma reiterada en la literatura sobre pronóstico de demanda intermitente (Syntetos et al., 2009).

La función de pérdida Tweedie resulta adecuada para este problema porque modela distribuciones con masa en cero y valores positivos continuos (Smyth & Jørgensen, 2002; XGBoost Developers, 2024). La distribución Tweedie compuesta Poisson-Gamma es una familia de distribuciones de probabilidad que combina dos comportamientos estadísticos en un único modelo: una masa de probabilidad discreta concentrada en cero (modelada como un proceso de Poisson) y una distribución continua positiva para los valores no nulos (modelada como una distribución Gamma) (Jørgensen, 1997). Esta estructura matemática refleja exactamente el comportamiento de los datos del M5: muchos días sin venta intercalados con días de venta positiva de magnitud variable. La media y la varianza de la distribución Tweedie cumplen:

$$\mathrm{Var}(Y) = \phi\,\mu^{p}, 1 < p < 2$$

donde Yes la variable aleatoria modelada (ventas diarias por SKU en el contexto del trabajo), $\mu = E[Y]$ es la media de la distribución, ϕ es el parámetro de dispersión y p es el parámetro de potencia de la varianza (tweedie_variance_power en XGBoost), que determina la forma de la distribución. Para $1 < p < 2$ la distribución es una mezcla

compuesta Poisson-Gamma que combina una masa de probabilidad concentrada en el cero con una cola continua de valores positivos, estructura idónea para datos con inflación de ceros como los del dataset M5 (Smyth & Jørgensen, 2002; Tweedie, 1984).

El parámetro tweedie_variance_power controla la forma de la distribución dentro de la familia. Valores cercanos a 1 producen comportamientos próximos a una distribución de Poisson pura (apropiada para conteos enteros sin componente continua); valores cercanos a 2 producen comportamientos próximos a una Gamma pura (apropiada para datos continuos positivos sin ceros). El valor intermedio tweedie_variance_power=1.1 utilizado en este trabajo es estándar para datos con alta proporción de ceros y cola positiva moderada, y es el recomendado por la documentación oficial de XGBoost para problemas de pronóstico de demanda con intermitencia (Chen & Guestrin, 2016).

5.5.3 Partición temporal y prevención de la data leakage

La partición de los datos en conjuntos de entrenamiento y validación constituye una de las decisiones metodológicas más críticas en el pronóstico de series temporales. En problemas de aprendizaje supervisado convencionales, la práctica habitual es asignar las observaciones a los conjuntos de forma aleatoria. Esta estrategia no resulta válida en series temporales: separar aleatoriamente los datos rompería la estructura temporal del problema y permitiría que el modelo aprendiese a partir de información futura para predecir el pasado, situación conocida como data leakage (Bergmeir et al., 2018).

La estrategia adoptada en este trabajo es la partición temporal estricta: los datos se dividen mediante un único corte cronológico que separa el pasado (entrenamiento) del futuro (validación). En el experimento sobre la serie agregada de CA_1, el conjunto de entrenamiento abarca los días 1 a 1.913 (1.878 observaciones tras eliminar las filas iniciales sin lag_35) y el conjunto de validación los días 1.914 a 1.941 (28 observaciones). En el experimento sobre el dataset completo se aplica la misma partición temporal, pero descartando además los primeros 1.000 días para reducir el coste de memoria sin pérdida de información relevante: el conjunto de entrenamiento queda con 27.867.860 observaciones (productos × días) y el de validación con 853.720 observaciones (30.490 productos × 28 días).

El horizonte de evaluación coincide exactamente con los 28 días reservados como horizonte de validación en el archivo sales_train_evaluation.csv de la competición oficial (Makridakis et al., 2022). Esta coincidencia permite que los resultados obtenidos sean directamente comparables, en términos metodológicos, con los publicados por los participantes oficiales de la competición M5.

Hiperparámetros del modelo. Los principales hiperparámetros configurados son learning_rate=0.05, max_depth=6 para CA_1 y max_depth=8 para el dataset completo, subsample=0.8, colsample_bytree=0.8 y min_child_weight=10 para CA_1 (50 para el dataset completo). Los dos parámetros de subsampling implementan la estocasticidad introducida por Friedman (2002) como mecanismo de regularización: cada árbol se construye utilizando solo el 80 % de las filas y el 80 % de las columnas del dataset disponible, lo que reduce el sobreajuste impidiendo que el modelo memorice patrones específicos del entrenamiento. El parámetro min_child_weight actúa como regularizador estructural: valores altos obligan al modelo a crear particiones más conservadoras,

evitando dividir el dataset en grupos demasiado pequeños que podrían reflejar ruido en lugar de patrones generalizables (Chen & Guestrin, 2016). El incremento de este parámetro al pasar de CA_1 al dataset completo refleja el aumento del tamaño del dataset: en un conjunto de 27 millones de observaciones, una hoja con menos de 50 observaciones tiene una probabilidad mucho mayor de capturar ruido que una hoja con el mismo umbral en un conjunto de 1.878 observaciones.

Mecanismo de early stopping. XGBoost incorpora un mecanismo de detención temprana que monitoriza el error sobre el conjunto de validación después de cada árbol construido. Si tras construir un nuevo árbol el error de validación no mejora durante un número consecutivo de iteraciones (fijado en early_stopping_rounds=50 en este trabajo), el algoritmo detiene el entrenamiento automáticamente y devuelve el modelo correspondiente a la mejor iteración observada. Este mecanismo resuelve un problema clásico del aprendizaje automático: durante el entrenamiento, el error sobre el conjunto de entrenamiento disminuye monotónicamente a medida que se añaden árboles, pero el error sobre el conjunto de validación primero disminuye y luego comienza a aumentar cuando el modelo empieza a memorizar ruido específico del entrenamiento (Hastie, Tibshirani & Friedman, 2009). El early stopping identifica automáticamente el punto donde el modelo deja de generalizar y detiene el proceso antes de que el sobreajuste se consolide.

5.5.4 Resultados sobre la serie agregada de CA_1

El modelo XGBoost entrenado sobre la serie agregada de CA_1 utiliza únicamente siete variables: lag_28, lag_35, rmean_28_7, rmean_28_28, wday, month y year. Se excluyen las variables categóricas (item_id, dept_id, etc.) porque al trabajar sobre una serie agregada toda la información jerárquica colapsa en un único valor. El modelo se entrena sobre los 1.878 días disponibles tras eliminar las filas iniciales sin lag_35 y predice los días 1.914 a 1.941.

El RMSE obtenido es de 397,10 unidades por día, resultado que se sitúa en segunda posición entre los cinco modelos evaluados, por detrás únicamente de SARIMA (388,26) y por encima del Naive Estacional (433,53), Prophet (500,15) y ARIMA (825,77). La mejora respecto al Baseline Naive Estacional es del 8,40 %. Para contextualizar la magnitud del error: dado que la tienda CA_1 vende en promedio 4.787 unidades diarias durante el horizonte de evaluación, el RMSE de 397,10 representa aproximadamente un 8,30 % del volumen medio, porcentaje que se sitúa dentro del rango habitual reportado en la literatura para pronóstico de demanda en retail (Fildes et al., 2022).

El hecho de que SARIMA supere ligeramente a XGBoost en este escenario concreto admite una explicación estructural. Sobre una única serie agregada (limpia, continua y dominada por una autocorrelación semanal clara) los componentes autorregresivos explícitos de SARIMA capturan las dependencias a corto plazo de forma directa, mientras que XGBoost debe inferir esa estructura indirectamente a través de los lags introducidos como features. La ventaja arquitectónica de XGBoost no se manifiesta a esta escala porque las condiciones del problema reducido (una sola serie, sin variables exógenas, sin patrones cruzados entre productos) anulan precisamente las características que distinguen a un modelo global de un modelo univariante (Montero-Manso & Hyndman, 2021). La

verdadera ventaja de XGBoost emerge al escalar el experimento al dataset completo, como se demuestra a continuación.

5.5.5 Escalado al dataset completo

El verdadero potencial de XGBoost se manifiesta cuando se aplica al dataset completo de 30.490 series. A esta escala, el modelo incorpora el conjunto completo de 16 variables descrito en el apartado 5.4.1: las cuatro de retardo y medias móviles, las cinco categóricas que identifican la jerarquía del producto, las tres temporales, las tres variables SNAP y el precio de venta.

El entrenamiento se detiene automáticamente en la iteración 107 mediante el mecanismo de early stopping, de un máximo configurado de 1.000 iteraciones. El modelo alcanzó su mejor rendimiento sobre el conjunto de validación tras construir 107 árboles, lo que indica que el modelo encuentra rápidamente una representación estable de los patrones del dataset y que iteraciones adicionales no aportan mejora generalizable. Este comportamiento es consistente con datasets de gran volumen donde la abundancia de observaciones permite a cada árbol extraer información sustancial en pocas iteraciones (Chen & Guestrin, 2016).

Evaluación a nivel SKU. El RMSE obtenido sobre los 853.720 puntos del conjunto de validación (30.490 productos × 28 días) es de 2,1988 unidades por SKU por día. Aplicado a los mismos puntos, el Baseline Naive Estacional (que predice las ventas del día t como las del día t - 28) obtiene un RMSE de 5,0985 unidades por SKU por día. La mejora de XGBoost respecto al Baseline Naive es, por tanto, del 56,87 %. En términos prácticos, el modelo global se equivoca de media en algo más de dos unidades por producto y día, frente a las cinco unidades del modelo ingenuo.

Esta magnitud es comparable a las mejoras reportadas por el equipo ganador de la competición M5 en su descripción técnica oficial: el modelo de Makridakis et al. (2022) (basado en LightGBM con función Tweedie e ingeniería de features similar a la empleada en este trabajo) obtuvo mejoras del mismo orden de magnitud sobre los benchmarks oficiales de la competición. La consistencia entre ambos resultados confirma que el enfoque metodológico adoptado es coherente con el estado del arte en pronóstico de demanda retail a nivel SKU.

Evaluación a nivel agregado corporativo. Para permitir la comparación con Prophet (que opera solo a nivel agregado), se suman las 30.490 predicciones individuales del modelo XGBoost para cada uno de los 28 días de validación, obteniendo así la venta diaria total a nivel corporativo. El RMSE agregado resultante es de 3.352,97 unidades por día. El Naive Estacional aplicado a las 30.490 series y posteriormente agregado produce un RMSE de 3.726,84 (mejora de XGBoost del 10,03 %), y Prophet, ajustado directamente sobre la serie corporativa agregada, produce un RMSE de 4.627,87 (XGBoost mejora un 27,57 % respecto a Prophet). La media de ventas diarias agregadas en el horizonte de evaluación es de 43.992 unidades, por lo que el error relativo de XGBoost en este nivel es del 7,62 %.

Síntesis del rendimiento. La mejora del 56,87 % de XGBoost sobre el Naive a nivel SKU no se explica únicamente por la sofisticación del algoritmo. Se explica por la combinación de tres decisiones de diseño directamente derivadas del análisis exploratorio: la incorporación de retardos múltiplos de siete que capturan el ciclo dominante del dataset,

la función de pérdida Tweedie que trata la inflación de ceros como una característica estructural y no como una anomalía, y la arquitectura de modelo global que permite aprender patrones cruzados entre productos, tiendas y categorías que ningún modelo univariante puede explotar (Januschowski et al., 2020). Cada una de estas decisiones tiene su origen en un hallazgo concreto del EDA, no son elecciones arbitrarias sino conclusiones del análisis previo de los datos.

5.6 Contraste de Resultados

El contraste de resultados constituye el punto de convergencia de todo el trabajo: es el momento en que los cuatro modelos evaluados se presentan de forma conjunta y se extraen las conclusiones que dan respuesta al objetivo principal de la investigación. Sin embargo, antes de presentar los números, resulta imprescindible establecer el marco interpretativo correcto, porque una lectura superficial de los resultados podría llevar a conclusiones erróneas.

5.6.1 Marco interpretativo: la comparación entre escalas

Los cuatro modelos evaluados no operan todos sobre el mismo nivel de agregación del dataset M5, lo que significa que sus métricas de error no son directamente comparables entre sí. Esta heterogeneidad no es un defecto del diseño experimental sino una consecuencia inevitable de las distintas características de cada modelo y, como se argumenta a continuación, constituye en sí misma uno de los hallazgos más relevantes del trabajo.

El Naive Estacional y XGBoost operan sobre el nivel 12 de la jerarquía M5 (es decir, predicen las ventas de cada uno de los 30.490 SKUs individualmente, día a día). Sus RMSEs expresan el error medio en unidades por producto por día, una magnitud directamente interpretable en el contexto operativo de gestión de inventarios. SARIMA y Prophet, en cambio, operan sobre niveles superiores de la jerarquía (concretamente sobre la serie agregada de la tienda CA_1 en el caso de SARIMA, y sobre la serie corporativa total en el caso de Prophet evaluado a escala completa). Sus RMSEs expresan el error medio en unidades totales por día, una magnitud entre 100 y 1.000 veces mayor simplemente porque el volumen del fenómeno que predicen es radicalmente distinto.

Comparar el RMSE de XGBoost (2,1988 unidades por SKU por día) con el de SARIMA (388,26 unidades por tienda por día) como si midieran lo mismo sería tan incorrecto como comparar el error al predecir el peso de una persona con el error al predecir el peso total de un edificio entero: ambos se expresan en kilogramos, pero la escala del fenómeno hace la comparación directa completamente inválida (Hyndman & Koehler, 2006).

Para resolver esta limitación, los resultados se presentan en dos planos separados. En el primer plano (denominado nivel SKU), se comparan únicamente los modelos que operan a la granularidad mínima del dataset: Naive Estacional y XGBoost. En el segundo plano (denominado nivel agregado corporativo), se comparan los modelos evaluados sobre la serie de ventas diarias totales: Naive Estacional escalado y agregado, XGBoost escalado y agregado, y Prophet ajustado directamente sobre la serie agregada. El modelo SARIMA queda excluido de la comparativa agregada por las razones de escalabilidad documentadas en el apartado 5.2.3, y se compara únicamente con los modelos evaluados sobre la serie de CA_1 (un nivel intermedio de la jerarquía).

5.6.2 Resultados por grupo de comparación

Plano del nivel SKU individual. En este plano (el único nivel en que la competición M5 exige operar) la comparación entre Naive Estacional y XGBoost es metodológicamente válida y directamente interpretable. El Naive Estacional obtuvo un RMSE de 5,0985 unidades por SKU por día, estableciendo el listón mínimo de calidad. XGBoost obtuvo un RMSE de 2,1988, lo que representa una reducción del error del 56,87 %. En términos prácticos, esto significa que XGBoost se equivoca de media en algo más de dos unidades por producto y día, frente a las más de cinco unidades del modelo ingenuo. Para un retailer del tamaño de Walmart (con decenas de miles de productos y cientos de tiendas), una reducción del 56,87 % en el error de predicción tiene implicaciones operativas y económicas de gran magnitud: menos sobreabastecimiento, menos roturas de stock y menos desperdicio alimentario.

Plano del nivel agregado corporativo. En este plano se comparan tres modelos sobre la misma magnitud (la venta diaria total agregada de los 30.490 SKUs). El Naive Estacional, aplicado a cada serie y posteriormente agregado, obtuvo un RMSE de 3.726,84 unidades por día. XGBoost, también aplicado a cada serie y posteriormente agregado, obtuvo un RMSE de 3.352,97 unidades por día, lo que supone una mejora del 10,03 % respecto al Naive. Prophet, ajustado directamente sobre la serie agregada, obtuvo un RMSE de 4.627,87 unidades por día, lo que lo sitúa un 24,18 % por debajo del Naive. La media de ventas diarias agregadas en el horizonte de evaluación es de 43.992 unidades, por lo que los errores relativos correspondientes son del 7,62 % para XGBoost, del 8,47 % para Naive y del 10,52 % para Prophet.

Modelo	RMSE (unidades/día)	Mejora respecto al Naive (%)	Error relativo (%)
XGBoost	3.352,97	+10,03 %	7,62 %
Naive Estacional	3.726,84	0,00 %	8,47 %
Prophet	4.627,87	-24,18 %	10,52 %

Tabla 3: Comparativa de modelos en el nivel agregado corporativo. Fuente: elaboración propia a partir de la ejecución del notebook adjunto.

Estos resultados a nivel agregado revelan dos observaciones importantes. La primera es que Prophet, evaluado a la escala donde un modelo univariante con descomposición aditiva debería destacar (una única serie continua con tendencia clara y estacionalidad bien definida), no consigue superar al Baseline más sencillo. Esta observación es consistente con los hallazgos de Hewamalage et al. (2021) sobre la dificultad estructural de los modelos aditivos para superar baselines triviales en series con un único ciclo estacional dominante y horizonte corto. La segunda observación es que XGBoost, aunque no ha sido diseñado para operar sobre series agregadas, supera a Prophet por un margen amplio (un 27,57 % de mejora) gracias a la información cruzada entre las 30.490 series que su arquitectura le permite aprovechar.

Plano del nivel de tienda agregada (CA_1). Sobre la serie agregada de la tienda CA_1 (un nivel intermedio de la jerarquía que permite la comparación directa de los cinco

modelos evaluados en condiciones controladas) los resultados se ordenan según muestra la Tabla X. SARIMA obtiene el menor error con un RMSE de 388,26, seguido por XGBoost (397,10), Naive Estacional (433,53), Prophet (500,15) y ARIMA (825,77). La diferencia entre SARIMA y XGBoost en este nivel es relativamente pequeña (aproximadamente un 2,3 % a favor de SARIMA) y refleja la ventaja estructural que los modelos autorregresivos tienen sobre una serie agregada limpia, sin intermitencia y con autocorrelación a corto plazo dominante.

Modelo	RMSE	Mejora vs Naive (%)	Error relativo (%)
SARIMA (1,1,1)(1,1,1)7	388,26	+10,44	8,11
XGBoost	397,10	+8,40	8,30
Naive Estacional	433,53	0,00	9,06
Prophet	500,15	-15,37	10,45
ARIMA (2,1,2)	825,77	-90,48	17,25

Tabla 4: Comparación de los cinco modelos sobre la serie agregada de la tienda CA_1 en el horizonte de 28 días. Media de ventas diarias en el horizonte de evaluación: 4.787 unidades. Fuente: elaboración propia a partir de la ejecución del notebook adjunto.

Síntesis visual de la evaluación. Los resultados expuestos sobre los dos planos de comparación admiten una representación gráfica conjunta que facilita la lectura comparativa de los modelos. La Figura 13 sintetiza, en su panel superior izquierdo, el rendimiento de los cinco modelos sobre la serie agregada de la tienda CA_1; en el panel superior derecho, el rendimiento de los tres modelos directamente comparables sobre el dataset completo agregado a nivel corporativo (XGBoost, Naive Estacional y Prophet); y, en el panel inferior, la trayectoria diaria de las predicciones de cada modelo frente a las ventas reales observadas durante los 28 días del horizonte de evaluación. En todos los gráficos de barras se incluye una línea discontinua roja que señala el nivel de error del Baseline Naive Estacional, lo que permite identificar visualmente qué modelos superan dicho umbral de referencia (Hyndman & Athanasopoulos, 2021).

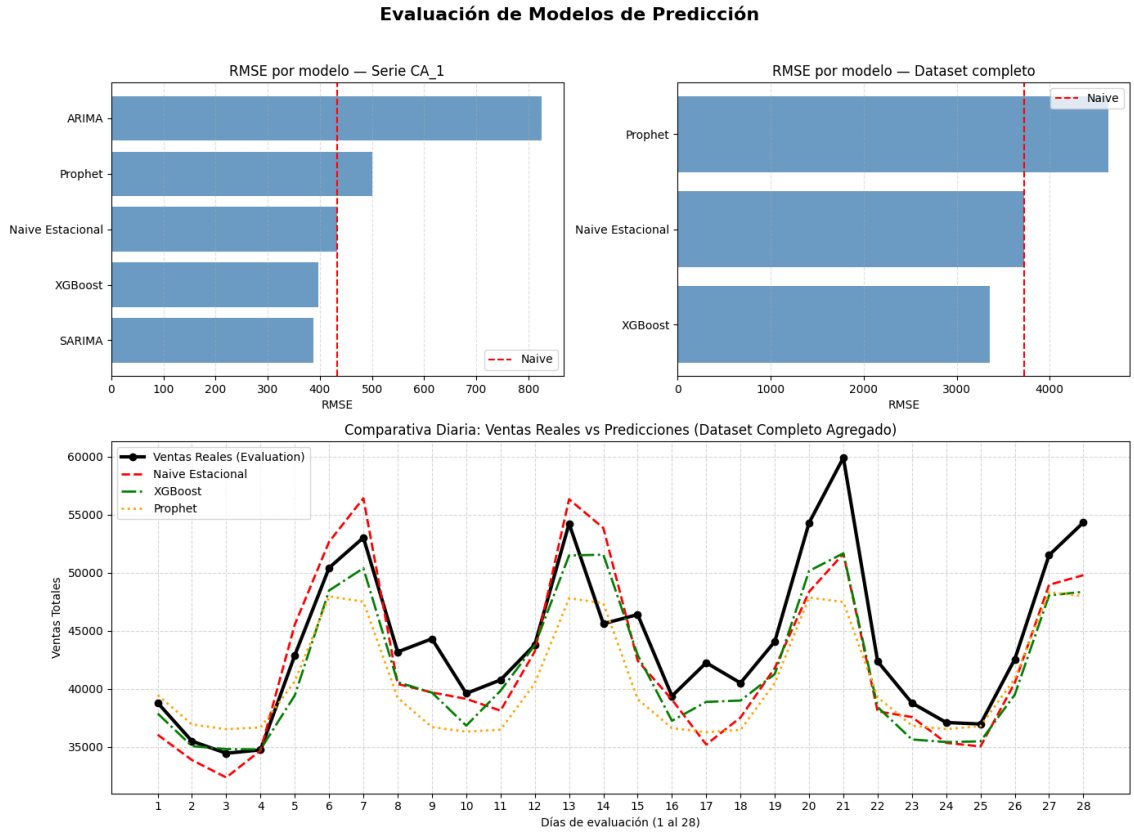

Figura 13: Evaluación comparativa de los modelos de predicción. Panel superior izquierdo: RMSE por modelo sobre la serie agregada de la tienda CA_1. Panel superior derecho: RMSE por modelo sobre el dataset completo agregado a nivel corporativo. Panel inferior: comparativa diaria entre las ventas reales observadas (línea negra) y las predicciones de los tres modelos directamente comparables sobre la serie corporativa agregada durante los 28 días del horizonte de evaluación. Fuente: elaboración propia a partir de la ejecución del notebook adjunto.

La representación gráfica refuerza tres observaciones cuantitativas ya recogidas en los apartados anteriores. En primer lugar, sobre la serie de CA_1, SARIMA y XGBoost se sitúan a la izquierda de la referencia Naive, mientras que Prophet y, sobre todo, ARIMA quedan claramente por encima de dicho umbral; la magnitud del error de ARIMA (RMSE de 825,77) evidencia la imposibilidad de capturar el patrón semanal sin componente estacional explícita (Box & Jenkins, 1970). En segundo lugar, sobre el dataset completo agregado a nivel corporativo, únicamente XGBoost logra superar al Baseline Naive Estacional, lo que confirma la limitación estructural de Prophet en series dominadas por estacionalidad semanal con horizonte corto (Hewamalage, Bergmeir & Bandara, 2021; Triebe et al., 2021). En tercer lugar, la trayectoria diaria reproducida en el panel inferior muestra que las tres predicciones siguen razonablemente el ciclo semanal observado en las ventas reales, pero también que ninguno de los modelos logra anticipar adecuadamente los picos extremos de demanda (en torno a los días 13 y 21 del horizonte), fenómeno que en la literatura se asocia a eventos exógenos de calendario no capturados por la estructura univariante de los modelos (Fildes et al., 2022).

5.6.3 El hallazgo metodológico central

Más allá de los números concretos, el resultado más valioso de este contraste es de naturaleza metodológica: la evidencia empírica de que el nivel de agregación al que se aplica un modelo determina completamente su viabilidad. SARIMA y Prophet funcionan correctamente (es decir, alcanzan o superan al baseline) únicamente cuando se aplican en

niveles de agregación donde las condiciones que sus supuestos matemáticos requieren se cumplen: la serie debe ser continua, la intermitencia debe haber desaparecido por efecto de la agregación, y el ciclo estacional debe ser claramente identificable. Pero incluso cumplidas estas condiciones, la ventaja sobre baselines triviales puede ser nula o negativa, como demuestra el caso de Prophet sobre CA_1 (Hewamalage, Bergmeir & Bandara, 2021). En el nivel SKU individual (que es precisamente donde el problema real de gestión de inventarios ocurre) estas condiciones no se cumplen y los modelos univariantes simplemente no son aplicables.

XGBoost resuelve esta tensión de forma estructural, no circunstancial. Su arquitectura de modelo global (un único modelo entrenado sobre todas las series simultáneamente) le permite operar nativamente en el nivel más granular sin necesidad de ajustar miles de modelos independientes. Su función de pérdida Tweedie le permite manejar la inflación de ceros que invalida a los modelos estadísticos clásicos (Smyth & Jørgensen, 2002). Y su capacidad para incorporar variables exógenas (precio, festivos, SNAP) le proporciona información contextual que ningún modelo univariante puede explotar (Januschowski et al., 2020). Estas tres ventajas estructurales, demostradas empíricamente a través de una reducción del 56,87 % en el RMSE respecto al Baseline a nivel SKU, son la respuesta al objetivo principal planteado al inicio de este trabajo: en el contexto del pronóstico de demanda retail a escala granular, los algoritmos de aprendizaje automático superan a los modelos estadísticos clásicos, y XGBoost en particular es el enfoque más adecuado para las características específicas del dataset M5.

5.6.4 Limitación de la comparativa y trabajo futuro

Es importante señalar dos limitaciones del experimento comparativo realizado. La primera afecta a la métrica empleada: este trabajo utiliza el RMSE como métrica principal de comparación, mientras que la métrica oficial de la competición M5 es el WRMSSE (Weighted Root Mean Squared Scaled Error), que escala el error por la variabilidad histórica de cada serie y pondera cada SKU según su importancia en la jerarquía de ventas (Makridakis et al., 2022). El cálculo del WRMSSE requiere información adicional sobre la escala de cada serie individual que excede el alcance de este trabajo, pero su inclusión en una versión extendida del análisis permitiría una comparación más rigurosa con los resultados publicados por los participantes de la competición. Esta limitación no invalida las conclusiones obtenidas (la superioridad de XGBoost sobre el Baseline es robusta independientemente de la métrica utilizada) pero sí representa una línea de trabajo futuro de interés directo.

La segunda limitación afecta a la heterogeneidad de niveles de agregación. Aunque el experimento se ha diseñado para que cada modelo opere en el nivel para el que es metodológicamente adecuado (lo cual es metodológicamente correcto), la consecuencia inevitable es que la comparación numérica directa entre todos los modelos solo es posible sobre la serie agregada de CA_1. La comparación a nivel SKU está restringida a Naive y XGBoost, y la comparación a nivel corporativo agregado a Naive, XGBoost y Prophet. Esta fragmentación de la comparativa es una consecuencia de las distintas restricciones de escalabilidad de cada modelo, y su reconocimiento explícito forma parte del argumento central del trabajo, pero impide una tabla numérica unificada con los cinco modelos sobre todas las escalas. Tal comparativa unificada solo sería posible mediante una

reconciliación jerárquica completa (Hyndman et al., 2011), técnica que también excede el alcance de este trabajo y queda como línea futura.

6. Objetivos de desarrollo sostenible (ODS)

6.1 Impacto en el ODS 12: Producción y Consumo Responsables

El presente trabajo no se limita a una contribución técnica en el ámbito del pronóstico de series temporales: tiene una dimensión de impacto real sobre uno de los grandes desafíos del sistema alimentario global. El Objetivo de Desarrollo Sostenible número 12 de la Agenda 2030 de las Naciones Unidas (Producción y Consumo Responsables) establece entre sus metas específicas reducir a la mitad el desperdicio de alimentos per cápita a nivel mundial para 2030 (Naciones Unidas, 2015) y disminuir las pérdidas de alimentos a lo largo de las cadenas de producción y suministro (Naciones Unidas, 2015, meta 12.3). La conexión entre esta meta y la precisión en el pronóstico de demanda retail es directa, cuantificable y documentada.

6.1.1 El problema del desperdicio en la cadena de suministro retail

Según la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO, 2019), aproximadamente un tercio de todos los alimentos producidos para consumo humano (alrededor de 1.300 millones de toneladas anuales) se pierde o desperdicia a lo largo de la cadena de suministro. En los países desarrollados, una parte significativa de este desperdicio se concentra en la fase de distribución minorista: los supermercados y grandes superficies generan pérdidas por dos mecanismos opuestos, pero igualmente costosos, ambos consecuencia directa de una gestión deficiente del inventario.

El primero es el sobreabastecimiento: cuando el sistema de gestión pide más producto del que el mercado va a demandar (Silver et al., 2016; Fildes et al., 2022), el excedente de productos frescos con fecha de caducidad corta (frutas, verduras, lácteos, carne) no puede venderse y debe desecharse. Estas mermas representan no solo una pérdida económica directa para el retailer sino también un desperdicio de todos los recursos empleados en la producción, el transporte y el almacenamiento de ese producto: agua, energía, trabajo y emisiones de CO_2 (FAO, 2019). El segundo mecanismo es la rotura de stock: cuando el sistema infraestima la demanda y el producto se agota antes de que llegue el siguiente pedido (Silver et al., 2016; Fildes et al., 2022), el retailer pierde la venta y el cliente queda insatisfecho, pero además el pedido de reposición de emergencia implica un coste logístico y una huella de carbono significativamente mayores que un pedido planificado en el ciclo normal (Eriksson et al., 2014).

Ambos mecanismos tienen la misma causa raíz: la incertidumbre sobre la demanda futura. Un sistema de pronóstico más preciso reduce esa incertidumbre y, con ella, tanto el sobreabastecimiento como las roturas de stock.

6.1.2 La conexión cuantitativa con los resultados del trabajo

La mejora del 56,87 % en el RMSE obtenida por XGBoost respecto al Baseline Naive a nivel SKU no es solo un resultado académico, sino que tiene una traducción operativa concreta. En la gestión de inventarios, el nivel de stock de seguridad que un retailer debe mantener para cubrir la incertidumbre de la demanda es directamente proporcional al

error de predicción durante el tiempo de reposición: cuanto mayor el error, mayor el stock de seguridad necesario para evitar roturas con una determinada probabilidad objetivo (Silver et al., 2016). Conviene matizar que la relación entre RMSE y stock de seguridad no es estrictamente lineal (el stock de seguridad escala como $z \cdot \sigma_{error}$, donde z es el factor asociado al nivel de servicio objetivo), pero el sentido cualitativo es claro: reducciones sustanciales del error de predicción se traducen en reducciones sustanciales del capital inmovilizado en inventario y, por extensión, de las mermas asociadas al producto perecedero.

En el contexto concreto de Walmart (la mayor cadena de supermercados del mundo por ingresos, con más de 10.000 establecimientos a nivel global), incluso una mejora marginal en la precisión predictiva tiene implicaciones a una escala difícilmente comparable con cualquier otra intervención. La propia existencia de la competición M5, organizada por Walmart y los investigadores de la Universidad de Nicosia con un premio total de 100.000 dólares, refleja la valoración que la industria otorga a las mejoras en este ámbito (Makridakis et al., 2022).

McKinsey & Company (2022) estima que la optimización de la cadena de suministro mediante técnicas de análisis predictivo puede reducir el desperdicio alimentario en el retail entre un 20 % y un 50 %, dependiendo del contexto y el nivel de implementación. Aunque este trabajo no implementa el modelo en un entorno de producción real (lo que constituye una de sus limitaciones explícitas y una línea de trabajo futuro prioritaria), los resultados obtenidos demuestran que la tecnología necesaria para alcanzar esas mejoras es accesible, implementable con herramientas de código abierto y reproducible en un entorno académico estándar.

6.1.3 Sostenibilidad económica y ambiental como objetivos complementarios

La reducción del desperdicio alimentario no es solo una cuestión medioambiental, sino también un imperativo económico para el sector retail (FAO, 2019; Aiello et al., 2017). Las mermas representan uno de los principales costes operativos de los supermercados, con impacto directo en los márgenes de beneficio en un sector caracterizado por márgenes muy ajustados (Aiello et al., 2017). En este sentido, el ODS 12 y la rentabilidad empresarial no son objetivos contrapuestos sino perfectamente alineados: mejorar la precisión del pronóstico de demanda reduce simultáneamente el desperdicio ambiental y el coste operativo.

Este alineamiento entre sostenibilidad y eficiencia económica es precisamente el argumento que justifica la inversión del sector en tecnologías de pronóstico avanzado. El dataset M5, al proporcionar un entorno de pruebas empírico con datos reales de Walmart, permite validar esas tecnologías en condiciones representativas del problema real antes de su despliegue operativo, reduciendo el riesgo de la inversión y acelerando la adopción de soluciones que contribuyen directamente a los objetivos de la Agenda 2030 (Naciones Unidas, 2015).

7. Conclusión

Este trabajo de investigación ha abordado un problema real y de alta relevancia práctica: determinar qué enfoque de análisis de series temporales ofrece menor error predictivo en el contexto del pronóstico de demanda retail a escala granular, utilizando como banco de pruebas el dataset M5 Forecasting – Accuracy de Walmart. A lo largo del desarrollo se han evaluado cuatro modelos de distinta naturaleza y complejidad, y los resultados obtenidos permiten extraer conclusiones sólidas, respaldadas tanto por la evidencia empírica del experimento como por la literatura académica consultada.

7.1 En relación con el objetivo principal

La hipótesis central del trabajo (que los algoritmos de aprendizaje automático superan a los modelos estadísticos clásicos en el contexto del pronóstico de demanda retail con alta intermitencia y múltiples variables exógenas) queda confirmada por los resultados. XGBoost, como modelo global de Gradient Boosting, es el único enfoque evaluado que opera a la escala que exige el dataset M5 y supera al Baseline Naive Estacional con una reducción del error del 56,87 % (RMSE de 2,1988 frente a 5,0985 unidades por SKU por día sobre el horizonte de evaluación oficial).

Esta superioridad no se explica únicamente por la sofisticación del algoritmo. Se explica por la combinación de tres decisiones de diseño directamente derivadas del análisis exploratorio: la incorporación de retardos múltiplos de siete que capturan el ciclo dominante del dataset, la función de pérdida Tweedie que trata la inflación de ceros como una característica estructural y no como una anomalía (Smyth & Jørgensen, 2002), y la arquitectura de modelo global que permite aprender patrones cruzados entre productos, tiendas y categorías que ningún modelo univariante puede explotar (Montero-Manso & Hyndman, 2021). Cada una de estas decisiones tiene su origen en un hallazgo concreto del EDA, no son elecciones arbitrarias sino conclusiones del análisis previo de los datos.

7.2 En relación con los modelos estadísticos clásicos

SARIMA y Prophet han demostrado un comportamiento dispar cuando se aplican en las condiciones para las que fueron diseñados. SARIMA, evaluado sobre la serie agregada de la tienda CA_1, capturó correctamente los patrones estructurales del dataset (tendencia creciente, ciclo semanal y efectos de calendario) y obtuvo el menor error de todos los modelos en ese nivel concreto (RMSE de 388,26 unidades por día, un 10,44 % por debajo del Naive). Prophet, en cambio, obtuvo un resultado peor que el Baseline tanto sobre CA_1 (RMSE 500,15; un 15,37 % por debajo del Naive) como sobre la serie corporativa agregada (RMSE 4.627,87; un 24,18 % por debajo del Naive). Este último resultado, que contradice la expectativa de partida, encuentra explicación en la literatura sobre modelos aditivos descomponibles: en series dominadas por un único ciclo estacional con horizonte corto, la arquitectura aditiva de Prophet penaliza la precisión punto a punto a cambio de interpretabilidad y robustez frente a cambios estructurales, sin que esta segunda propiedad sea explotable en un horizonte de 28 días (Hewamalage, Bergmeir & Bandara, 2021; Triebe et al., 2021).

Esta evaluación también demostró empíricamente la limitación fundamental de ambos modelos univariantes: ninguno es viable al nivel de granularidad que exige el problema real. Aplicarlos a los 30.490 SKUs del dataset M5 requeriría ajustar 30.490 modelos

independientes (computacionalmente inviable en un entorno convencional) y además seguiría sin resolver el problema de la intermitencia, que invalida sus supuestos estadísticos en el nivel más granular. Esta demostración empírica refuerza y complementa el argumento teórico desarrollado en el Marco Teórico, convirtiendo la argumentación académica en una evidencia experimental directa.

7.3 En relación con el proceso metodológico

Una de las conclusiones más importantes de este trabajo es de naturaleza metodológica y transversal: el análisis exploratorio de datos no es un paso preliminar decorativo sino la base sobre la que se construyen todas las decisiones técnicas relevantes. Los tres hallazgos del EDA (tendencia creciente, estacionalidad semanal dominante e intermitencia del 68,2 % de ceros) tuvieron consecuencias directas y verificables sobre el diseño del modelo.

El hallazgo de la estacionalidad semanal justificó la elección de retardos múltiplos de siete (lag_28 y lag_35), garantizando que el modelo compare siempre días equivalentes y previniendo simultáneamente la data leakage que retardos más cortos habrían introducido (Bergmeir et al., 2018).

El hallazgo de la intermitencia justificó la elección de la función de pérdida Tweedie en lugar del error cuadrático estándar. Sin esta elección, el modelo habría intentado minimizar un error calculado sobre una distribución que no representa la naturaleza real de los datos, produciendo predicciones sesgadas hacia valores positivos donde la respuesta correcta es cero (Syntetos et al., 2009).

El hallazgo de la tendencia creciente justificó la inclusión de las variables year y month como features, permitiendo al modelo distinguir el comportamiento de un lunes de 2011 del comportamiento de un lunes de 2016, cuando el volumen de ventas era significativamente superior.

Este proceso de ida y vuelta entre el análisis exploratorio y las decisiones de modelado ilustra una práctica fundamental en ciencia de datos: los modelos no deben diseñarse de forma independiente de los datos sobre los que operan. El conocimiento del dominio y el análisis previo de los datos son tan determinantes para el resultado final como la elección del algoritmo (Hyndman & Athanasopoulos, 2021).

7.4 Limitaciones del trabajo

Este trabajo presenta tres limitaciones principales que deben tenerse en cuenta al interpretar los resultados.

La primera es la heterogeneidad de niveles de agregación entre modelos. La comparación entre los RMSEs de SARIMA y Prophet (calculados a nivel de tienda agregada) y los de Naive y XGBoost (calculados a nivel de SKU individual) no es directamente válida. Esta heterogeneidad es una consecuencia inevitable de las distintas restricciones de escalabilidad de cada modelo, y su reconocimiento explícito forma parte del argumento central del trabajo. La incorporación de Prophet sobre el dataset agregado corporativo (RMSE 4.627,87) y de XGBoost agregado equivalente (RMSE 3.352,97) ha permitido una comparativa parcial a escala corporativa que mitiga (sin eliminar) esta limitación.

La segunda es la métrica de evaluación utilizada. Este trabajo emplea el RMSE como métrica principal de comparación, mientras que la métrica oficial de la competición M5 es el WRMSSE, que escala el error por la variabilidad histórica de cada serie y pondera

los SKUs según su importancia en la jerarquía (Makridakis et al., 2022). El cálculo del WRMSSE requiere información adicional sobre la escala de cada serie que excede el alcance de este trabajo. Su implementación en una versión extendida permitiría una comparación más rigurosa con los resultados publicados por los participantes oficiales de la competición.

La tercera es la ausencia de implementación en producción. Los resultados obtenidos demuestran la superioridad de XGBoost en un entorno experimental controlado, pero no validan su comportamiento en un sistema de gestión de inventarios real. La transición de un modelo de investigación a un sistema operativo implica desafíos adicionales (latencia de inferencia, actualización continua del modelo, integración con los sistemas de gestión existentes) que quedan fuera del alcance de este TFG.

7.5 Líneas de investigación futura

Los resultados de este trabajo abren varias líneas de continuación de alto interés académico y práctico.

La más inmediata es la comparación con arquitecturas de Deep Learning, en particular con redes LSTM (Long Short-Term Memory) y variantes recientes como N-BEATS o Temporal Fusion Transformers. Como se argumentó en el Marco Teórico, las redes recurrentes son capaces de capturar dependencias temporales a muy largo plazo sin necesidad de especificar manualmente la estructura del modelo, lo que las convierte en candidatos naturales para superar a XGBoost en datasets con patrones temporales complejos (Lim, Arik, Loeff & Pfister, 2021). La comparación directa entre XGBoost y una arquitectura de Deep Learning entrenada sobre el mismo dataset M5, con la misma partición temporal y la misma métrica de evaluación, constituiría una extensión natural y de gran valor de este trabajo.

La segunda línea es la implementación de reconciliación jerárquica óptima. Este trabajo opera exclusivamente en el nivel 12 de la jerarquía M5 (el nivel de SKU individual). La aplicación de técnicas de reconciliación jerárquica óptima (Wickramasuriya et al., 2019) garantizaría la coherencia entre todos los niveles de agregación simultáneamente, alineando las predicciones individuales con los totales de departamento, tienda y estado. Este es uno de los problemas abiertos más activos en la literatura de forecasting jerárquico.

La tercera línea es el cálculo del WRMSSE para situar este trabajo dentro del ranking oficial de la competición M5 (Makridakis et al., 2022). Aunque la superioridad de XGBoost sobre el Naive es robusta independientemente de la métrica, el cálculo del WRMSSE permitiría comparar el modelo desarrollado en este trabajo con las soluciones publicadas por los 5.582 equipos participantes de la competición.

La cuarta línea es la integración operativa del modelo en un sistema real de gestión de inventarios, con el objetivo de cuantificar el impacto real sobre las métricas de negocio que importan al retailer: reducción de mermas, reducción de roturas de stock y ahorro en costes logísticos. Esta integración cerraría el círculo entre la investigación académica y el impacto práctico sobre el ODS 12, transformando un resultado experimental en una herramienta con valor operativo demostrable.

7.6 Reflexión final

Este trabajo ha demostrado que es posible construir un sistema de pronóstico de demanda retail competitivo utilizando exclusivamente herramientas de código abierto, en un entorno de hardware convencional y con un proceso metodológico riguroso pero accesible para un estudiante de grado. La clave no ha residido en la complejidad del algoritmo elegido sino en la comprensión profunda de los datos (sus patrones, sus limitaciones y sus características estadísticas) y en la traducción de ese conocimiento en decisiones de diseño justificadas y verificables.

El dataset M5 de Walmart, con sus 58 millones de observaciones, su estructura jerárquica de 12 niveles y su extrema intermitencia, representa uno de los benchmarks más exigentes de la literatura de forecasting (Makridakis et al., 2022). Haber obtenido resultados coherentes con la literatura y haber superado el Baseline oficial en un 56,87 % a nivel SKU es, en ese contexto, un resultado satisfactorio que valida tanto el proceso metodológico seguido como las decisiones técnicas tomadas a lo largo del desarrollo.

8. Anexo Disponibilidad de Datos y Código

Con el objetivo de garantizar la trazabilidad y reproducibilidad del proyecto, el código fuente completo empleado en el análisis exploratorio de datos, la ingeniería de características y el entrenamiento de modelos se encuentra disponible públicamente en GitHub.

- Repositorio: TFG-Series-Temporales-Retail
- Última revisión consultada: 14/05/2026 (*commit acb2e36)*
- Licencia: MIT

El proyecto utiliza el conjunto de datos M5 Forecasting Accuracy, disponible en Kaggle:

- Fuente oficial: M5 Forecasting Accuracy (Kaggle)

El repositorio incluye el cuaderno principal del proyecto: (*TFG_M5_Forecasting_Tebar.ipynb*), el archivo de dependencias (*requirements.txt*) y documentación adicional sobre los datos empleados (*docs/Datos.md*).

Los datos originales no se incluyen en el repositorio debido a restricciones de tamaño y a las condiciones de uso establecidas por Kaggle.

9. Referencias

Aiello, G., Giovino, I., Vallone, M., & Catania, P. (2017). A multi objective approach to short food supply chain management. Chemical Engineering Transactions, 58, 313–318. https://doi.org/10.3303/CET1758053

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

Bergmeir, C., Hyndman, R. J., & Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. Computational

Statistics and Data Analysis, 120, 70–83. https://doi.org/10.1016/j.csda.2017.11.003

Box, G. E. P., & Jenkins, G. M. (1970). Time series analysis: Forecasting and control. Holden-Day.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: Forecasting and control (5th ed.). John Wiley & Sons.

Bracewell, R. N. (2000). The Fourier transform and its applications (3rd ed.). McGraw-Hill.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. Wadsworth.

Brooks, C. (2019). Introductory econometrics for finance (4th ed.). Cambridge University Press.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. Journal of Statistical Software, 76(1), 1–32. https://doi.org/10.18637/jss.v076.i01

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Cleveland, R. 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), 3–33.

Croston, J. D. (1972). Forecasting and stock control for intermittent demands. Operational Research Quarterly, 23(3), 289–303. https://doi.org/10.2307/3007885

Eriksson, M., Strid, I., & Hansson, P. A. (2014). Waste of organic and conventional meat and dairy products—A case study from Swedish retail. Resources, Conservation and Recycling, 83, 44-52.

FAO. (2019). The state of food and agriculture 2019: Moving forward on food loss and waste reduction. Food and Agriculture Organization of the United Nations. https://www.fao.org/publications/sofa/2019/en/

Fildes, R., Ma, S., & Kolassa, S. (2022). Retail forecasting: Research and practice. International Journal of Forecasting, 38(4), 1283–1318. https://doi.org/10.1016/j.ijforecast.2019.06.004

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Friedman, J. H. (2002). Stochastic gradient boosting. Computational Statistics & Data Analysis, 38(4), 367–378. https://doi.org/10.1016/S0167-9473(01)00065-2

Groen, A. (2017). Reducing DataFrame memory size by ~65% [Notebook]. Kaggle. https://www.kaggle.com/code/arjanso/reducing-dataframe-memory-size-by-65

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press. https://www.deeplearningbook.org/

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S.,

van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer. https://hastie.su.domains/ElemStatLearn/

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting, 37(1), 388–427. https://doi.org/10.1016/j.ijforecast.2020.06.008

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and practice (2nd ed.). OTexts. https://otexts.com/fpp2/

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

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. International Journal of Forecasting, 22(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

Hyndman, R. J., Ahmed, R. A., Athanasopoulos, G., & Shang, H. L. (2011). Optimal combination forecasts for hierarchical time series. Computational Statistics & Data Analysis, 55(9), 2579–2589. https://doi.org/10.1016/j.csda.2011.03.006

Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. International Journal of Forecasting, 36(1), 167–177. https://doi.org/10.1016/j.ijforecast.2019.05.008

Jørgensen, B. (1997). The theory of dispersion models. Chapman & Hall.

Kaggle. (2020). M5 forecasting—Accuracy. https://www.kaggle.com/c/m5-forecasting-accuracy

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.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter notebooks—A publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), Positioning and power in academic publishing: Players, agents and agendas (pp. 87–90). IOS Press. https://doi.org/10.3233/978-1-61499-649-1-87

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. Philosophical Transactions of the Royal Society A, 379(2194), Article 20200209. https://doi.org/10.1098/rsta.2020.0209

Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting, 37(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering, 31(12), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. International Journal of Forecasting, 38(4), 1346–1364. https://doi.org/10.1016/j.ijforecast.2021.11.013

Mauricio, J. A. (2007). Introducción al análisis de series temporales. Universidad Complutense de Madrid. https://www.ucm.es/data/cont/docs/518-2013-11-11-JAM-IAST-Libro.pdf

McKinney, W. (2010). Data structures for statistical computing in Python. Proceedings of the 9th Python in Science Conference, 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

McKinsey & Company. (2022). Reducing food loss and waste: Getting consumer companies and governments to act. https://www.mckinsey.com/industries/consumer-packaged-goods/our-insights/reducing-food-loss-and-waste-getting-consumer-companies-and-governments-to-act

Meldgaard, C. S., & Iversen, L. (2021). Learnings from Kaggle's forecasting competitions. arXiv. https://arxiv.org/abs/2009.07701

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. https://doi.org/10.1016/j.ijforecast.2021.03.004

Naciones Unidas. (2015). Transformar nuestro mundo: La Agenda 2030 para el Desarrollo Sostenible (A/RES/70/1). https://unctad.org/system/files/official-document/ares70d1_es.pdf

Pandas Development Team. (2024). pandas documentation. https://pandas.pydata.org/docs/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

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.

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. Proceedings of the 9th Python in Science Conference, 92–96. https://doi.org/10.25080/Majora-92bf1922-011

Silver, E. A., Pyke, D. F., & Thomas, D. J. (2016). Inventory and production management in supply chains (4th ed.). CRC Press. https://doi.org/10.1201/9781315374406

Smyth, G. K., & Jørgensen, B. (2002). Fitting Tweedie's compound Poisson model to insurance claims data: Dispersion modelling. ASTIN Bulletin, 32(1), 143–157. https://doi.org/10.2143/AST.32.1.1020

Syntetos, A. A., & Boylan, J. E. (2005). The accuracy of intermittent demand estimates. International Journal of Forecasting, 21(2), 303–314. https://doi.org/10.1016/j.ijforecast.2004.10.001

Syntetos, A. A., Boylan, J. E., & Disney, S. M. (2009). Forecasting for inventory planning: A 50-year review. Journal of the Operational Research Society, 60(S1), S149–S160. https://doi.org/10.1057/jors.2008.173

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

Triebe, O., Hewamalage, H., Pilyugina, P., Laptev, N., Bergmeir, C., & Rajagopal, R. (2021). NeuralProphet: Explainable forecasting at scale. arXiv. https://arxiv.org/abs/2111.15397

Tweedie, M. C. K. (1984). An index which distinguishes between some important exponential families. In J. K. Ghosh & J. Roy (Eds.), Statistics: Applications and new directions: Proceedings of the Indian Statistical Institute Golden Jubilee International Conference (pp. 579–604). Indian Statistical Institute.

UNIC. (2020). The M5 competition: Competitors' guide. University of Nicosia. https://www.unic.ac.cy/iff/research/forecasting/m-competitions/m5/

Van Rossum, G., & Drake, F. L. (2009). Python 3 reference manual. CreateSpace.

Viñals, M. P. (2009). Series temporales. Universitat Politècnica de Catalunya.

Wickramasuriya, S. L., Athanasopoulos, G., & Hyndman, R. J. (2019). Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. Journal of the American Statistical Association, 114(526), 804–819. https://doi.org/10.1080/01621459.2018.1448825

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. Management Science, 6(3), 324–342. https://doi.org/10.1287/mnsc.6.3.324

XGBoost Developers. (2024). XGBoost documentation. https://xgboost.readthedocs.io/

Universidad Europea VALENCIA

ANEXO - DECLARACIÓN DEL USO DE INTELIGENCIA ARTIFICIAL

En la elaboración del presente TFG se han utilizado herramientas de Inteligencia Artificial (IA) como apoyo en distintas fases del trabajo académico.

La herramienta utilizada ha sido **Claude** (Anthropic), modelo Claude Opus 4.7, accedido a través de la interfaz web claude.ai mediante una cuenta de pago con la funcionalidad *Projects* activada para mantener el contexto del notebook, el documento del TFG y la guía docente a lo largo de las distintas sesiones de trabajo.

Estas herramientas se han utilizado para las siguientes tareas:

- Búsqueda preliminar y síntesis de información.
- Generación y organización de ideas.
- Generación de esquemas o estructuras de contenido.
- Apoyo a la redacción y mejora de estilo académico.
- Reformulación de textos hacia la forma impersonal exigida por la guía docente.
- Generación, revisión o depuración de código software.
- Localización y formateo de referencias bibliográficas según las normas APA 7.
- Apoyo en análisis de datos o interpretación de resultados.
- Otras finalidades específicas: verificación de la coherencia entre las decisiones metodológicas implementadas en el notebook y su descripción en la memoria escrita, revisión cruzada entre las celdas Markdown del notebook y los apartados correspondientes del documento Word, y aplicación del feedback del tutor a apartados concretos del borrador.

DECLARO QUE:

- El contenido final presentado es resultado de mi trabajo personal y académico.
- He revisado y validado críticamente toda la información obtenida mediante herramientas de IA.
- El uso de IA no ha sustituido el análisis, la reflexión ni las conclusiones propias exigidas en el TFG.

Universidad Europea VALENCIA

- El trabajo cumple las normas de integridad académica y propiedad intelectual establecidas por la universidad.

Nombre del estudiante: Álvaro Tébar Folch

Firma:

