

Alix Partners: La Casa de Asterión

Predicción de demanda y optimización de precios

2025

Contenidos

1	Introducción	2
2	Análisis Económico	3
3	Análisis Técnico	6
3.1	Preprocesamiento de Datos	6
3.1.1	Análisis de Correlación entre Subgrupos	6
3.2	Modelo Predictivo	6
3.2.1	Exploración de modelos	6
3.2.2	Regresiones	6
3.2.3	Deep Learning	7
3.2.4	Boosting	8
3.3	Optimización de Precios	9
4	Consideraciones Finales	10

1 Introducción

En esta competencia, nos fue dado el siguiente desafío: un negocio de ventas minoristas, llamado "La Casa de Asterión", ha sufrido una reducción de ganancias en el último tiempo, y necesita mejorar su estrategia de precios para solucionar este problema. La empresa ha proporcionado los siguientes datos: las transacciones de ventas de los últimos 3 años, especificando las tiendas, los productos vendidos y sus precios. Además, también contamos con datos sobre las tiendas, como su ubicación, y sobre los productos, como su categoría y marca.

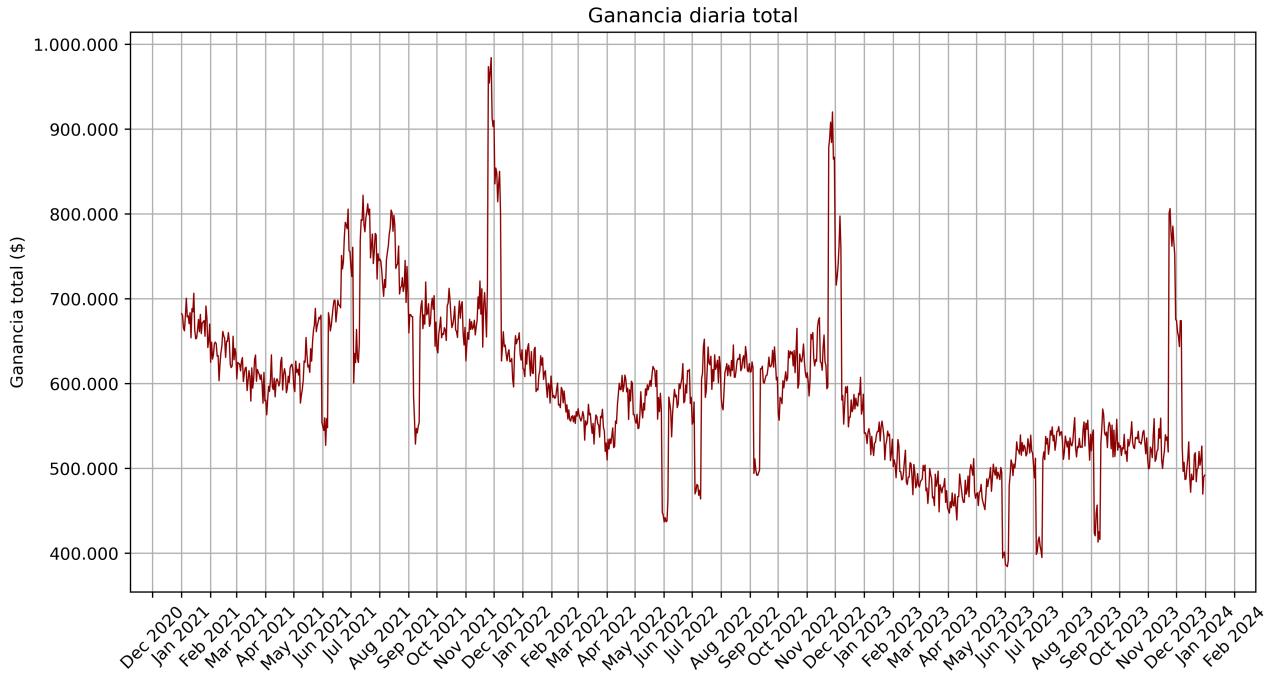
Entonces, valiéndonos de toda esta información, nuestro objetivo es construir un modelo capaz de predecir la demanda diaria de los productos de la próxima semana a partir del precio y otras variables, y con el mismo, optimizar los precios para maximizar las ganancias.

Dado que desconocemos si el cliente tiene conocimientos técnicos acerca de programación y desarrollo de modelos de AI, hemos decidido dividir este informe en dos secciones: por un lado, un análisis económico, el cual no necesita conocimientos técnicos para ser entendido, donde realizamos un análisis de datos básico y exponemos una estrategia de precios que, según las predicciones hechas, debería aumentar las ganancias; y por otro lado, realizamos un análisis técnico, en donde detallamos el preprocesamiento de datos, la construcción y entrenamiento del modelo predictivo y la metodología de optimización de precios.

Además, todo el código fuente utilizado para el trabajo está disponible adjunto al informe.

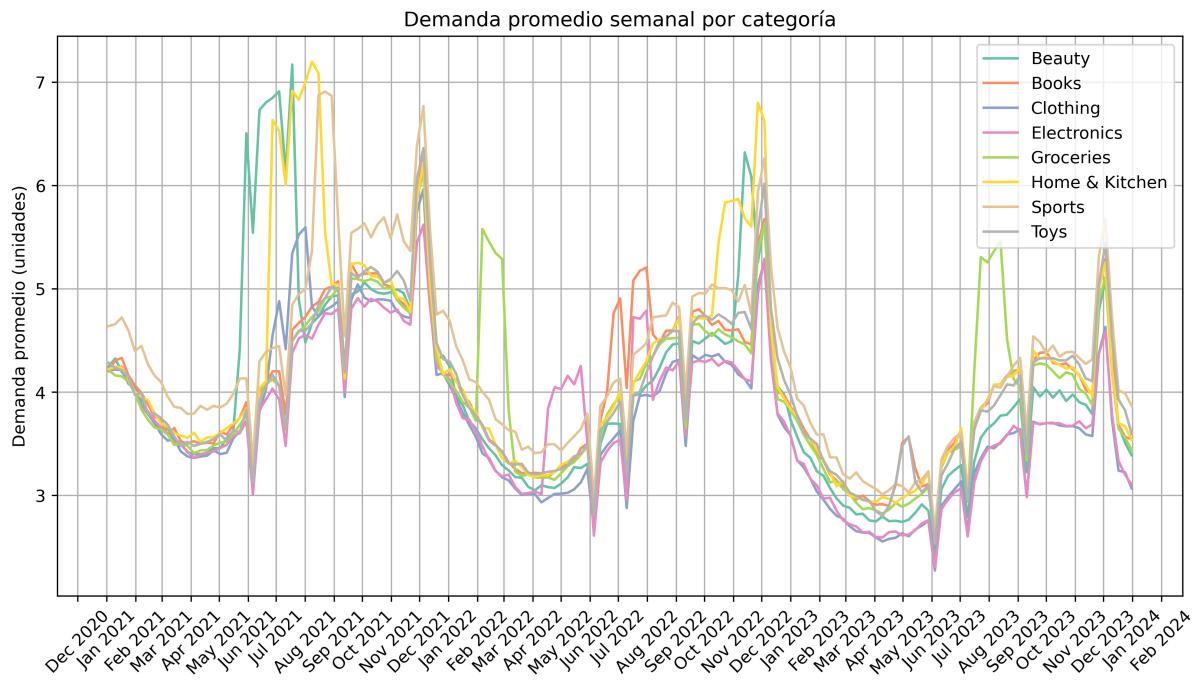
2 Análisis Económico

En primer lugar, nos preguntamos por qué se ha producido una caída de las ganancias y qué tan grave es. Para esto, se puede observar en el siguiente gráfico cómo fue evolucionando la ganancia total de la empresa a lo largo del tiempo, y cómo la misma ha ido disminuyendo de a poco en el transcurso del tiempo.



En concreto, con respecto al primer año, las ganancias totales se han reducido un 11.3% en el segundo año y un 23.7% en el tercer año, lo cual puede resultar preocupante.

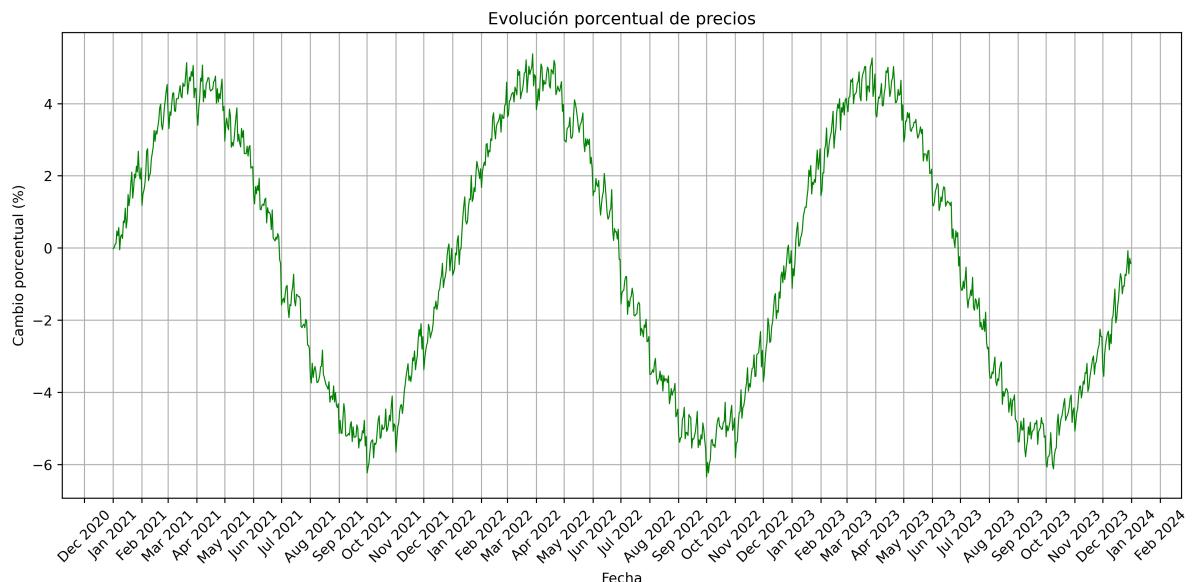
Este fenómeno puede deberse en gran parte a la reducción de la demanda total, que ha disminuido 10% el primer año y un 21.2% en el segundo año. Esto se puede apreciar en el siguiente gráfico, además de cómo varía la demanda según la categoría de los productos:



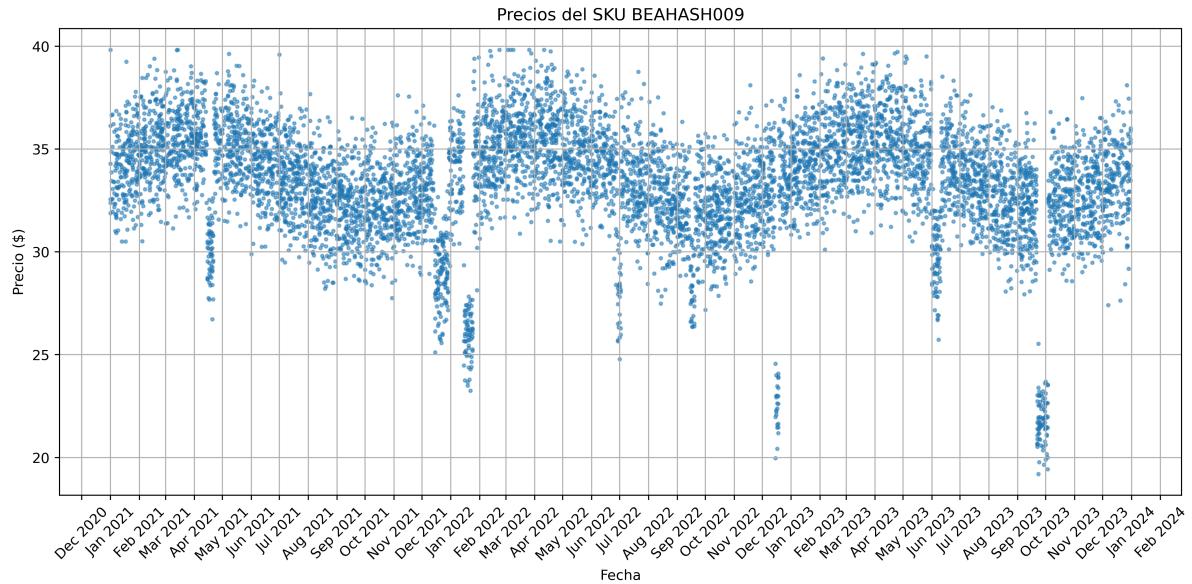
Al comparar este último gráfico con el de las ganancias totales, se pueden observar 3 comportamientos similares:

- A nivel global, hay una reducción gradual de las ganancias y la demanda
- En cada año, en general hay un incremento a medida que transcurre el año y un descenso a principios de cada año.
- En Mayo, Junio y Agosto hay caídas abruptas, mientras que en Diciembre hay un incremento sustancial, seguramente debido a las festividades.

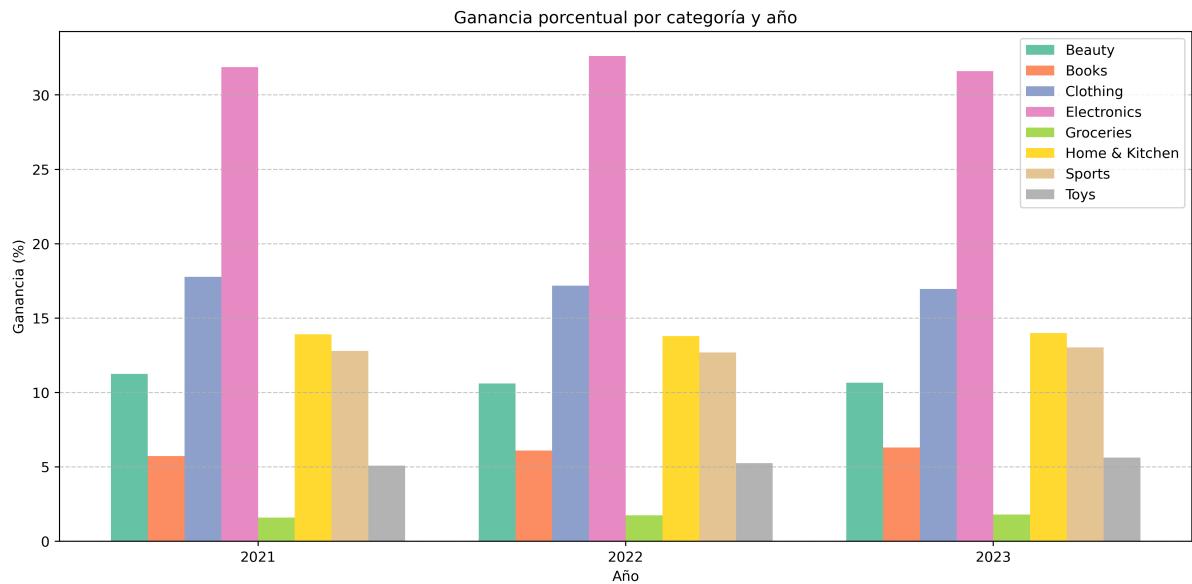
Decidimos analizar si estos descensos en la demanda se debían a cambios en los precios, sin embargo, en el siguiente gráfico se aprecia que en promedio los precios han seguido un mismo patrón:



Dado que los precios en general no han variado pero la demanda sí, nos lleva a pensar que es posible que un factor importante en la reducción sea la estrategia de precios actual, la cual no aprovecha correctamente la estacionalidad de la demanda. De hecho, en cada uno de los productos podemos observar un comportamiento similar: los precios oscilan de manera periódica a lo largo del año, pero en ciertos momentos, hay caídas abruptas en los precios, como se muestra a continuación.



Por otro lado, la disminución de la demanda de diferentes productos no afecta a las ganancias de la misma manera, puesto que las ganancias están constituidas mayormente por productos de la categoría de electrónica:



Antes de continuar, es importante aclarar que el subgrupo de productos de "Basketball" no registró ninguna venta en los últimos 3 años, lo que puede deberse a una falla en la carga de datos o a que el producto no se comercializa en las tiendas.

Teniendo en cuenta todo lo anterior,

3 Análisis Técnico

3.1 Preprocesamiento de Datos

Comenzamos el trabajo explorando los datos proporcionados y completando los datos faltantes. Después, nos quedamos con aquellas columnas que consideramos relevantes para el modelo e intentamos unir los 6 archivos csv en un mismo dataframe, sin embargo, los datos de los clientes no pudieron ser utilizados dado que las transacciones no contaban con el ID de cliente, por lo que no fue posible hacer unirlos.

Entonces, nos quedamos con los siguientes datos para cada transacción: fecha, producto (SKU), tienda, precio, costo, cantidad vendida (en unidades y en total), categorías del producto y ubicación de la tienda, entre otros.

Por otro lado, descartamos ciertos outliers que consideramos que podían llegar a cesgar al modelo: transacciones con precios, cantidad vendida unitaria o cantidad vendida total fuera del 99.9% quantile.

3.2 Análisis de Correlación entre Subgrupos

En un principio, consideramos entrenar un modelo especializado para cada serie SKU-tienda o subgrupo-tienda. Esto tendría como ventaja modelos más simples, pero la desventaja era perder información cruzada entre grupos. Esto nos llevó a preguntarnos si existía correlación entre dichos grupos. Si los grupos fueran suficientemente independientes, no habría relación cruzada y tendría sentido entrenar un modelo especializado por serie. Para evaluarlo, primero calculamos la correlación de Pearson, la cual mide relaciones lineales, y vimos que pocos subgrupos estaban fuertemente correlacionados. Sin embargo, como Pearson solo captura correlaciones lineales, decidimos calcular también la correlación de Spearman y Kendall, métricas que permiten detectar relaciones monótonas entre variables, es decir, si tienden a moverse en la misma dirección aunque no sean lineales. Este análisis mostró que en gran cantidad de grupos sí existía correlación, ya sea directa o inversa. Por lo tanto, concluimos que ignorar estas interacciones no sería lo más adecuado, y optamos por entrenar un modelo global capaz de captarlas y generalizarlas.

3.3 Modelo Predictivo

3.3.1 Exploración de modelos

En una primera instancia, realizamos una etapa de investigación en la cual buscamos qué modelos se implementan en la industria para abordar la problemática de pronóstico de demanda y estrategia de precios. Encontramos distintos artículos que mencionaban enfoques experimentales y tradicionales en mayor o menor medida. Decidimos probar al menos un modelo de cada una de las siguientes familias: - Boosting - Regresiones - Deep Learning

A continuación, se describen los distintos enfoques y pruebas realizadas durante esta etapa de exploración.

3.3.2 Regresiones

Como primera aproximación, entrenamos dos regresiones: una lineal y una polinómica. La regresión lineal no nos proporcionó un buen ajuste y presentó un R^2 muy bajo, ya que es un modelo simple incapaz de captar relaciones no lineales entre las variables y el target. Intentamos reducir la dimensionalidad evaluando la correlación de Pearson de las variables numéricas con el target ademas de la varianza del target explicada por las variables categoricas para ver cuan "unidas" estaban estas. Esto mostró que muy pocas variables numericas mantenían una relación lineal superior a 0.6, por lo que entrenar un modelo lineal únicamente con esas variables le habría restado información relevante del problema. Y las variables categoricas que tenian una relacion mas fuerte con el target, no se trataba de una relacion precisamente lineal, por ende el modelo no podia capturar bien estas dinamicas.

Por otro lado, entrenamos una regresión polinómica de grado 2, elegida para minimizar el riesgo de sobreajuste. Se utilizó un 80% de los datos para entrenamiento y un 20% para test, obteniendo un R^2 de 0.94 en el entrenamiento. Sin embargo, esto nos generó sospechas, ya que era extraño que un modelo tan simple explicara tan bien la varianza del target. Realizando un entrenamiento walk-forward con expanding window, observamos que las métricas de entrenamiento fluctuaban mucho en cada iteración, confirmando que el modelo no era robusto y que el alto R^2 inicial era señal de sobreajuste. Estos modelos, al no ser estrictamente de series temporales, permitían trabajar con datos faltantes. Aun así, probamos agregar rolling features para que el modelo captara mejor la dinámica temporal: medias móviles y desviaciones estándar calculadas sobre datos anteriores, en ventanas de 7, 90 y 180 días, agrupadas por subgrupo, por serie SKU-tienda, y considerando o no los datos faltantes. También evaluamos distintas codificaciones para variables categóricas, incluyendo One-Hot Encoding y Target Encoding, siendo esta última preferida por reducir el tamaño del dataset codificado. Todas estas aproximaciones no mejoraron el desempeño del modelo.

3.3.3 Deep Learning

Para la familia de modelos de Deep Learning pasamos por 2 opciones:

En un principio, evaluamos la posibilidad de implementar una Graph Neural Network. Esta idea nos resultaba atractiva, ya que con los grafos podríamos modelar dinámicas espaciales entre las features, como las relaciones entre tiendas. Sin embargo, decidimos descartarla debido a su complejidad: era necesario calcular una matriz de adyacencia que representara adecuadamente el grafo y no encontramos suficiente material sobre la aplicación de estas redes a nuestro problema, ni implementaciones disponibles. Además, trabajar con este modelo requería operar a un nivel más bajo y definir cuidadosamente cómo manejar las variables categóricas para que contribuyeran al entrenamiento.

También experimentamos con algunos modelos de la librería PyTorch Forecasting, que ofrece modelos muy potentes basados en diferentes filosofías, como la atención y la descomposición de series temporales. Además, permite el manejo de embeddings aprendidos para variables categóricas, lo cual resultaba ideal para nuestro problema. De esta librería probamos el modelo NHITS, que se basa en la descomposición de series para realizar predicciones. Sin embargo, esta no fue nuestra solución final debido a algunas problemáticas. Al tratarse de modelos para series temporales, el modelo necesitaba ver todas las series completas (SKU-STORE_ID-DATE-DATE) para predecir correctamente.

Para afrontar esto, intentamos un llenado de batches "on the fly", ya que llenar todo el dataset con todas las combinaciones posibles era inviable computacionalmente: nuestro dataset sería demasiado pesado para la RAM y aún así requeriría recursos adicionales para el entrenamiento. El llenado "on the fly" consistía en crear nuestra propia clase, que heredaba de la clase original pero sobreescribía uno de sus métodos. De este modo, cada vez que se tomaba un batch, se generaba un rango de fechas y se completaban con ceros los días sin ventas, permitiendo que el modelo viera las series completas sin saturar la memoria. Además, esta clase nos permitía entrenar cualquier modelo de la librería con la misma estrategia. Sin embargo, esta aproximación presentó varios problemas debido a la interacción de métodos entre nuestra clase personalizada y la original, sumado a nuestro desconocimiento del funcionamiento interno de la librería y la estructura de datos. Por estas razones, decidimos volver a un modelo más simple.

3.3.4 Boosting

El modelo que mejores resultados nos proporcionó fue el LightGBM, el cual es un modelo de boosting que utiliza árboles de decisión y es capaz de manejar grandes volúmenes de datos de manera eficiente.

En primera instancia, utilizamos las transacciones dadas para entrenar el modelo, pero esto provocaba que después a la hora de predecir la demanda por producto y tienda, el modelo no tuviera en cuenta los días en que no hubo ventas y sobreestimara por mucho la demanda. Por lo tanto, decidimos crear un dataframe con todas las combinaciones posibles de fechas, productos y tiendas, y completar los días en que no hubo ventas con un valor 0 en la cantidad vendida total.

Además, a partir del dataframe completo, para que el modelo pueda entender mejor la estacionalidad de la demanda, creamos nuevos lagging features: agrupamos las transacciones por producto y calculamos una media móvil de la cantidad vendida total y su desviación estándar de los últimos días, para distintas ventanas temporales. Hicimos lo mismo pero agrupando por tienda, como también por subgrupo de productos.

Por otro lado, para que el modelo pueda relacionar mejor el precio y la demanda, creamos otros lagging features sobre los precios: agrupando las transacciones por producto, calculamos la media móvil y desviación estándar del cambio porcentual del precio de los últimos días, para distintas ventanas temporales. También hicimos lo mismo, pero agrupando por subgrupo de productos.

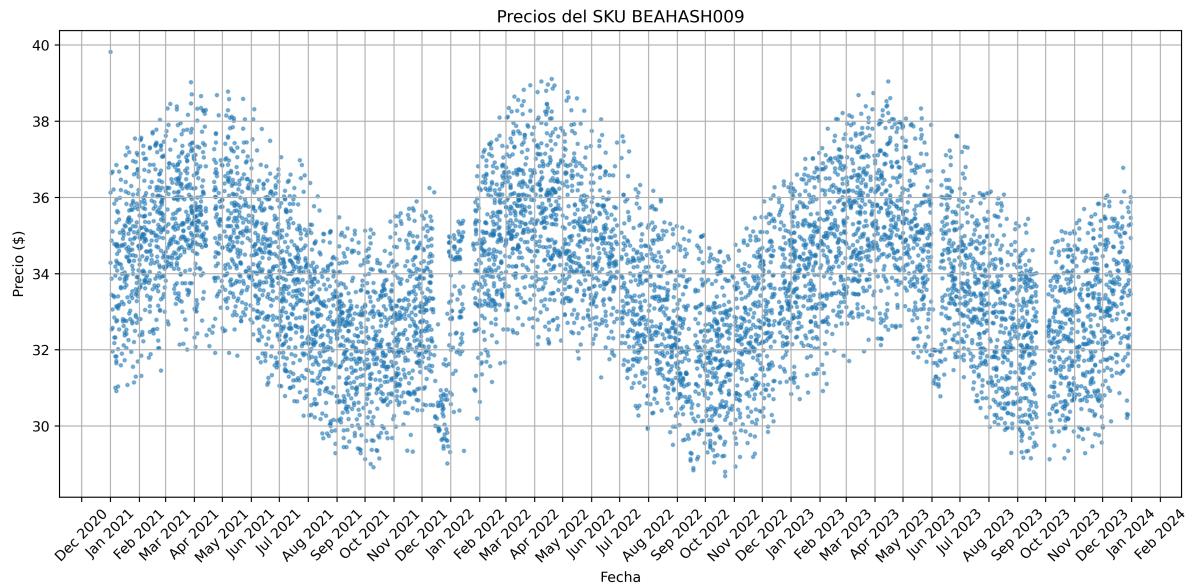
Entonces, con el dataframe completo y los features agregados, comenzamos a probar el modelo. Para esto, no podíamos emplear un cross-validation convencional, dado que los lagging features hubieran generado data leakage, por lo que decidimos hacer un walk-forward validation: entrenar el modelo con los datos del primer año, predecir la demanda de la próxima semana, guardar las predicciones y compararlas con las ventas reales, y luego sumar 30 días a los datos de entrenamiento y repetir el proceso. De esta manera, podemos evaluar la performance del modelo prediciendo la demanda de la próxima semana, tal como se haría si tuviéramos que hacerlo en tiempo real.

El fine-tuning del modelo se debió hacer de manera manual, puesto que el entrenamiento de cada modelo tenía un gran costo computacional y, por lo tanto, un Grid Search no era viable con los recursos disponibles.

3.4 Optimización de Precios

Después de validar el modelo, elegimos los hiperparámetros que mejor performance nos dieron y entrenamos el modelo con todos los datos disponibles. Luego, creamos un dataframe template con todas las combinaciones posibles de fechas, productos y tiendas de la próxima semana, excluyendo las tiendas que cerraron.

Debido a que probar todas las configuraciones de precios posibles para predecir cuál maximiza las ventas totales era computacionalmente inviable, primero redujimos el espacio de búsqueda: para cada producto, quitamos los precios que se apartaban demasiado del rango de los últimos días. Tomando como ejemplo los precios de BEAHASH009, ya vistos anteriormente, podemos ver a continuación los precios filtrados:



Entonces, tomamos como rango de precios posibles para cada producto el máximo y el mínimo de los precios filtrados en los últimos 30 días. Cabe aclarar que aún así los datos con los precios no filtrados sí fueron utilizados para entrenar el modelo.

Además, para simplificar la implementación de la estrategia de precios, decidimos que los precios de los productos serán los mismos en todas las tiendas y para toda la semana.

Así, con un espacio de búsqueda más manejable, aplicamos una optimización bayesiana (con la librería Optuna) para encontrar la configuración de precios que maximiza las ganancias totales según las predicciones del modelo.

4 Consideraciones Finales