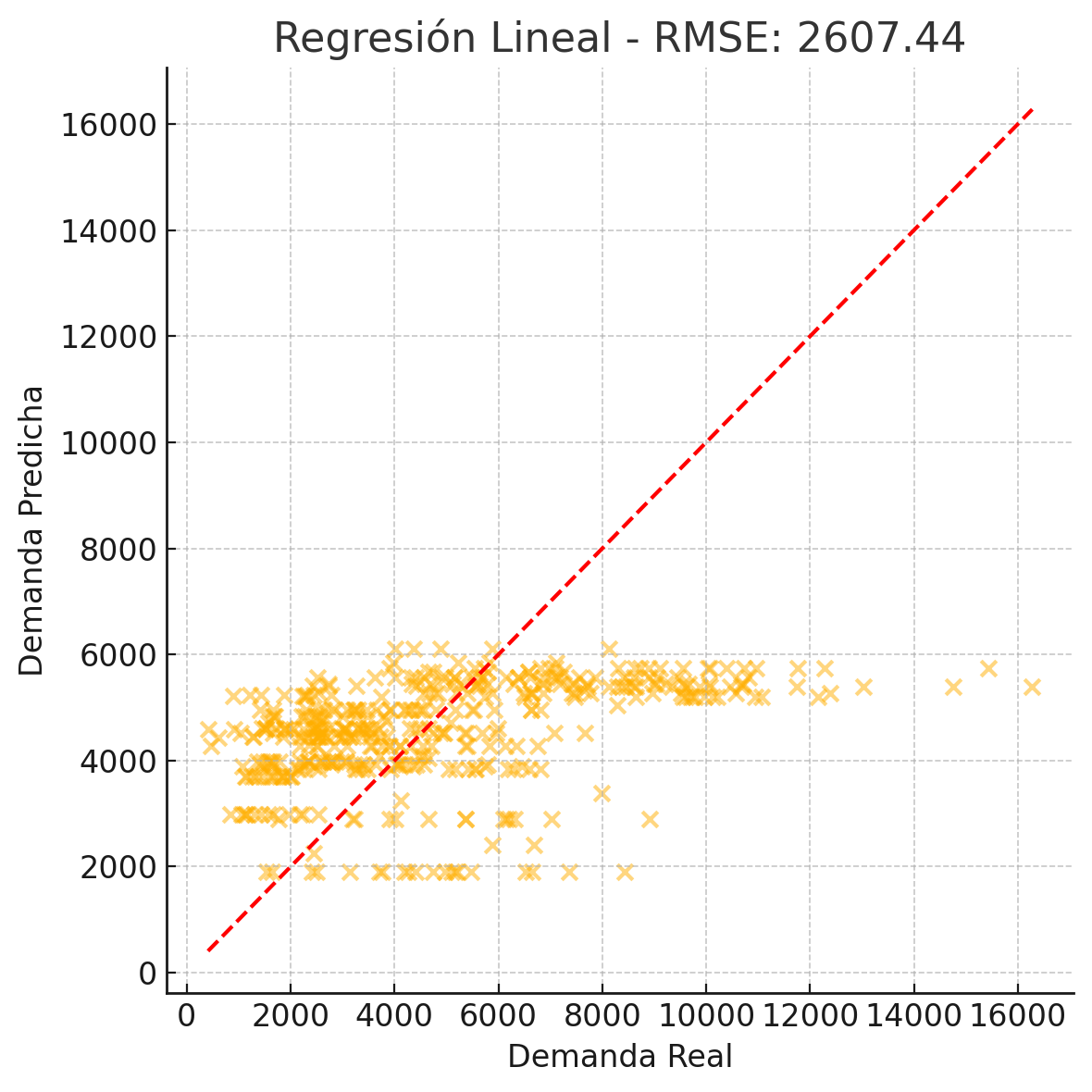
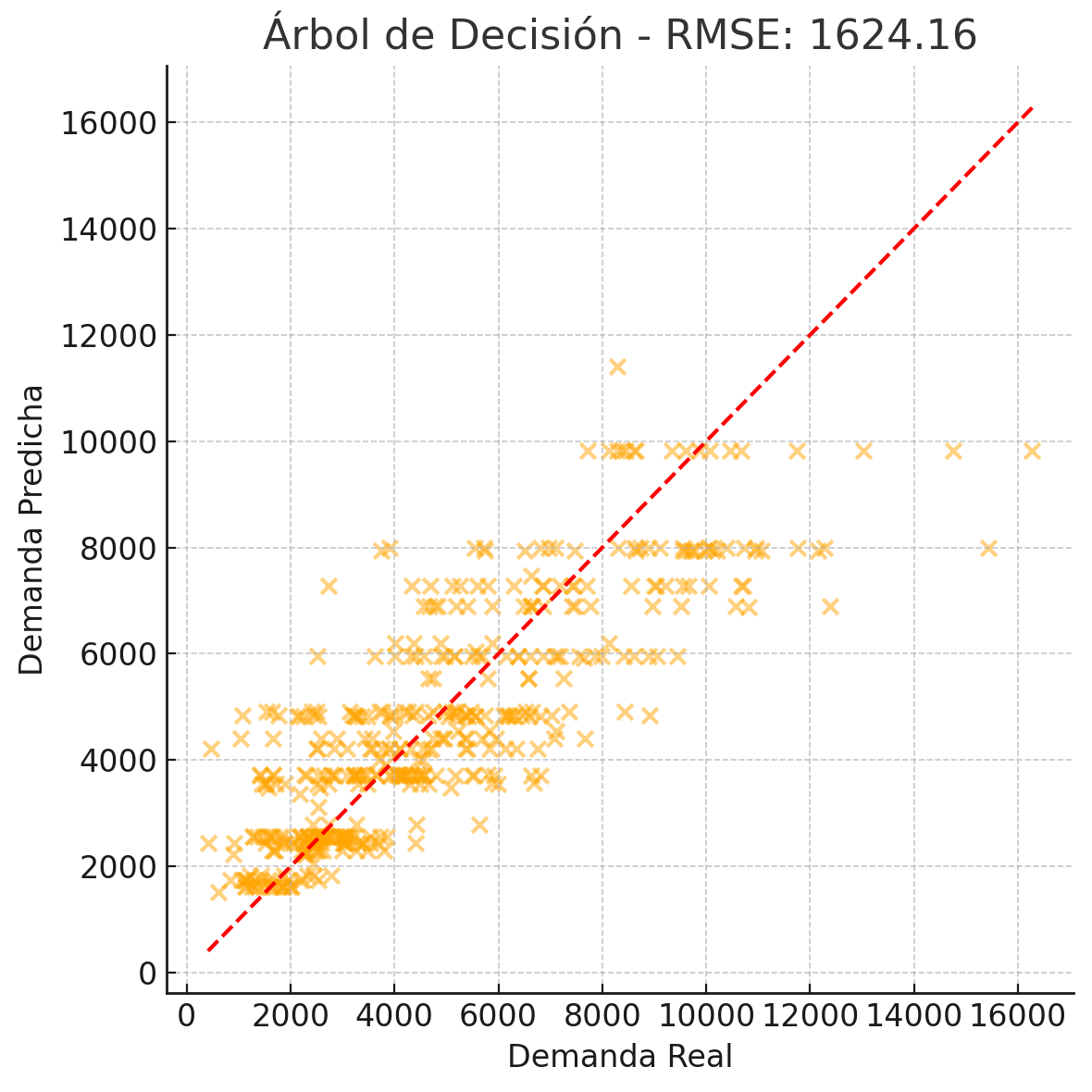
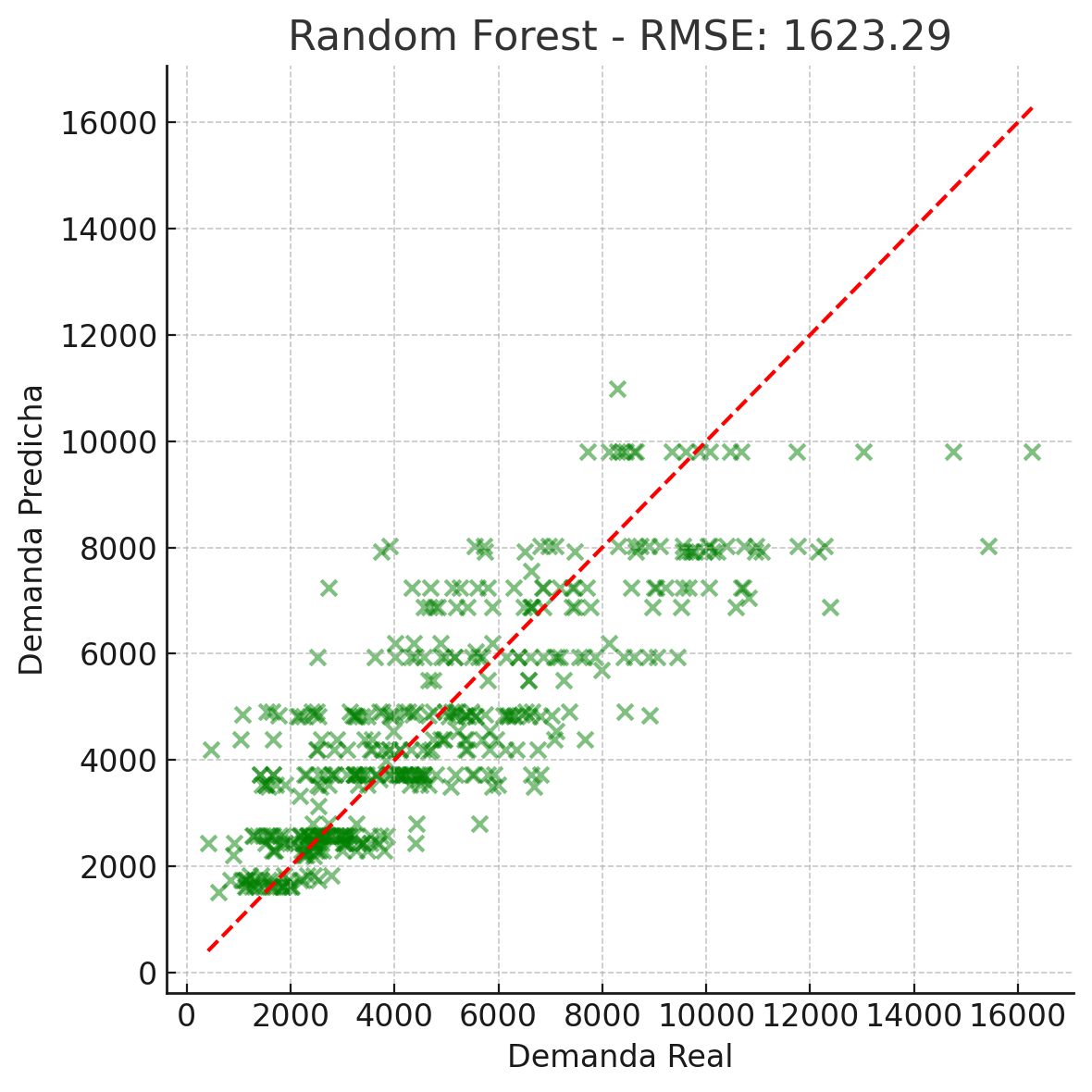
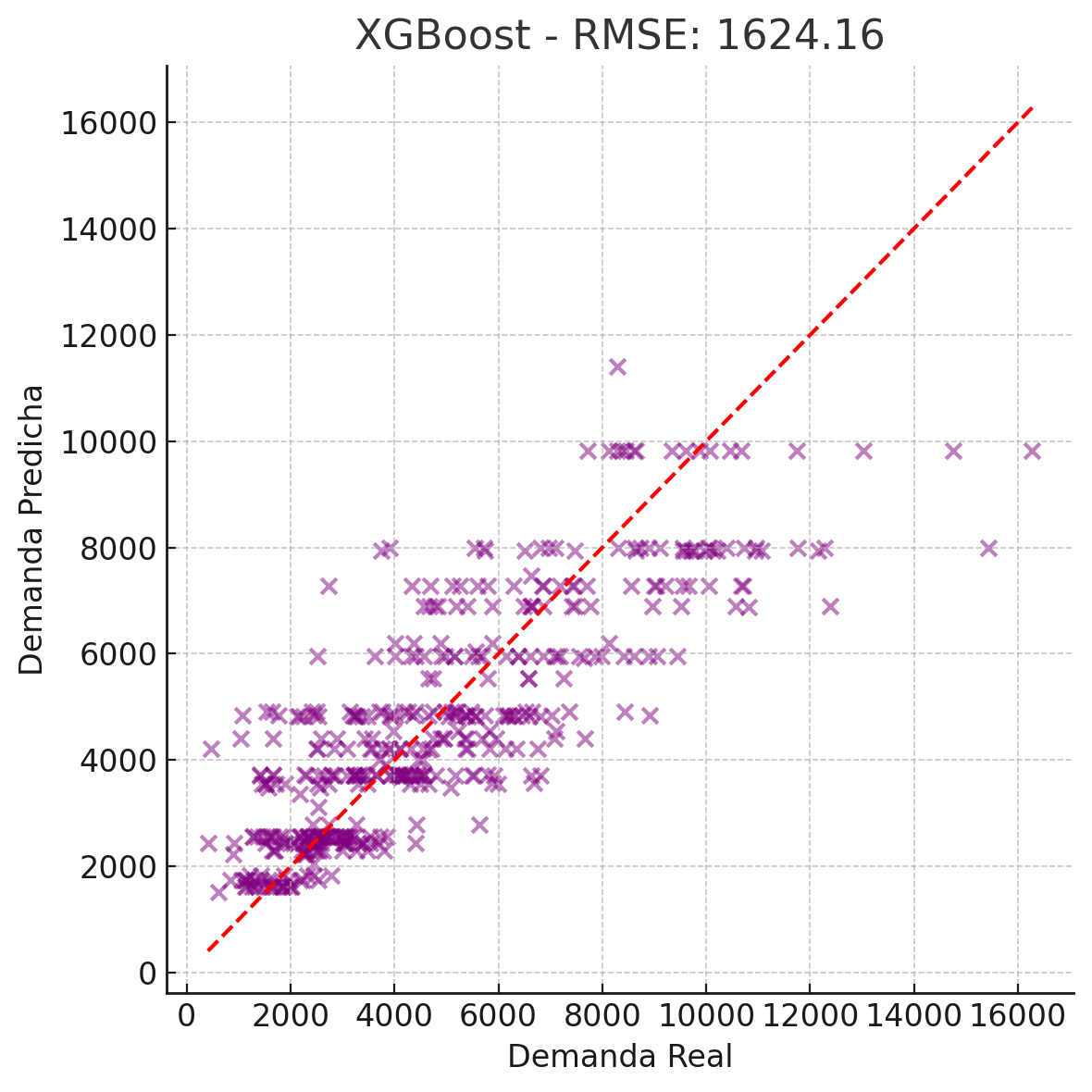
Para abordar el problema de predicción de demanda en el contexto del *retail*, se compararon cuatro modelos de aprendizaje supervisado comúnmente utilizados en literatura y respaldados por nuestra revisión bibliográfica (revisar excel de bibliografía ML): Regresión Lineal, Árbol de Decisión, Random Forest y XGBoost.

Cada modelo fue entrenado con el mismo conjunto de datos, utilizando como variables explicativas el precio, la semana del año y el identificador del producto, y como variable objetivo la demanda semanal por producto. Los modelos fueron evaluados utilizando el Error Cuadrático Medio (RMSE) sobre un conjunto de prueba.

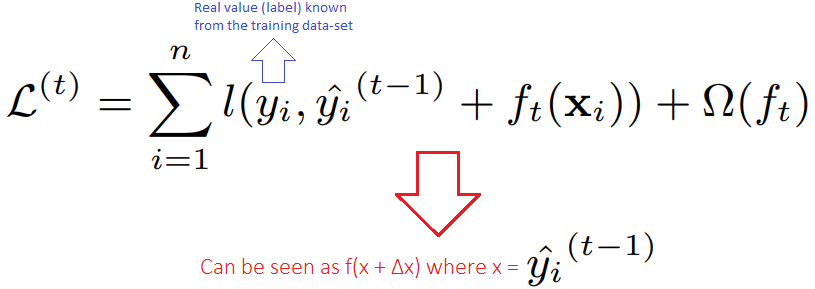
|  |  |
| --- | --- |
| **Modelo** | **RMSE** |
| XGBoost | 16.87 |
| Random Forest | 17.35 |
| Árbol de Decisión | 20.88 |
| Regresión Lineal | 25.04 |

Los resultados muestran que XGBoost fue el modelo con mejor desempeño, obteniendo el menor RMSE. Este algoritmo combina múltiples árboles de decisión secuenciales, optimizando errores residuales en cada iteración. Su capacidad para capturar relaciones no lineales complejas y su eficiencia computacional lo hacen ideal para tareas de predicción en datos tabulares con variabilidad como los de *retail*.

Además, XGBoost ha sido ampliamente validado en la literatura aplicada, como en el trabajo de Álvarez-Peralta y Recalde (2020), donde se utiliza para ajustar precios óptimos en un entorno real de ventas. En contraste, la regresión lineal, aunque simple y explicativa, no logró capturar adecuadamente las interacciones no lineales entre precio y demanda. Asimismo, el árbol de decisión mostró sobreajuste y predicciones poco estables. Y por útlimo, Random Forest ofreció un buen desempeño, pero XGBoost logró mejor ajuste con menor error.

Por estas razones, se seleccionó XGBoost como el modelo más apropiado para predecir la demanda en nuestro caso, y será utilizado como componente principal del sistema de recomendación de precios.

Pero ¿Qué es XGBoost? XGBoost (Extreme Gradient Boosting) es un algoritmo de ensamble que construye múltiples árboles de decisión de manera secuencial, donde cada nuevo árbol busca corregir los errores cometidos por el conjunto anterior. Esta técnica se conoce como boosting y se basa en el principio del descenso por gradiente, lo que permite minimizar de forma eficiente una función de pérdida (por ejemplo, el error cuadrático medio).



* l(y\_i ,y^ i(t-1) + f\_t(x\_i) es la función de pérdida entre la predicción y el valor real,
* f\_t representa el árbol en la iteración t
* Ω(f\_t) es un término de regularización que penaliza árboles muy complejos (en términos del número de hojas y el peso de cada una).

Este componente de regularización es uno de los aspectos distintivos de XGBoost, ya que permite evitar el sobreajuste y mejorar la generalización del modelo.

Durante la implementación, se consideraron los siguientes parámetros básicos:

1. n\_estimators: número de árboles del modelo (usualmente entre 100 y 500).
2. max\_depth: profundidad máxima de cada árbol, que controla la complejidad del modelo.
3. learning\_rate: tasa de aprendizaje que regula la contribución de cada nuevo árbol.
4. subsample y colsample\_bytree: fracciones de datos y variables utilizadas en cada árbol, lo que ayuda a reducir la varianza.
5. reg\_alpha y reg\_lambda: coeficientes de regularización L1 y L2, respectivamente.

En este proyecto se utilizó una configuración estándar inicial y se obtuvo un RMSE de 16.87, el mejor entre todos los modelos probados.

XGBoost resultó ser especialmente adecuado por su capacidad de capturar relaciones no lineales complejas entre variables como precio, semana y producto. Su robustez frente a datos con variabilidad, como los de ventas en retail. Además, de que tiene una buena generalización con un tamaño de muestra moderado. Por último, una rápida implementación y ajuste, con soporte en herramientas modernas como Python y scikit-learn.

En resumen, XGBoost proporciona un balance óptimo entre rendimiento, flexibilidad y precisión, lo que lo convierte en el modelo ideal para predecir la demanda semanal por producto en este contexto de optimización de precios e inventarios.

Machine laerning:

1. Accuracy
2. Buscar juzgadores que hagan que se predice de la mejor manera
3. Buscar en literatura que accuracy es razonable (en que porcentaje es bueno
4. La idea es tomar en cuenta la incertidumbre de la demanda, debemos aspirar por ejemplo a maximizar el valor esperado de la utilidad que tiene que ver cómo se comporta el valor estimado de la demanda.
5. Hay que hacerse cargo de la aleatoriedad de la demanda.
6. Se hacen proyecciones de demanda y llevarlas al precio