Necesito crear una aplicación de Streamlit para simular y visualizar predicciones de ventas de noviembre 2025. La app debe ser atractiva visualmente y funcional desde el primer intento. Debes usar seaborn para los gráficos.

Contexto del proyecto:

Tengo un modelo de ML entrenado (HistGradientBoostingRegressor) guardado en `models/modelo_final.joblib` y un dataframe de inferencia preparado en `data/processed/inferencia_df_transformado.csv` con 24 productos para noviembre 2025.

IMPORTANTE - Dataset ya preparado:

El archivo `inferencia_df_transformado.csv` YA tiene todas las transformaciones necesarias aplicadas:

- ✅ Variables temporales (año, mes, dia_mes, dia_semana, semana_año, trimestre, es_fin_semana, es_festivo, es_black_friday, es_cyber_monday, etc.)

*- ✅ Variables de lag (unidades_vendidas_lag_1 a unidades_vendidas_lag_7) **YA calculadas desde octubre***

*- ✅ Media móvil (unidades_vendidas_ma7) **YA calculada desde octubre***

- ✅ Variables de precio (descuento_porcentaje, precio_competencia, ratio_precio)

- ✅ One-hot encoding para nombre, categoria y subcategoria

- ✅ Columnas: Amazon, Decathlon, Deporvillage (precios competencia individuales)

El archivo SOLO contiene datos de noviembre 2025, pero los lags ya fueron inicializados correctamente con datos de octubre antes de filtrar.

Ya están todos los datos necesarios para poder predecir el 1 de noviembre.

CRÍTICO - Predicciones Recursivas:

Aunque los lags iniciales ya están en el archivo, debes actualizar los lags día a día al predecir:

1. El día 1 de noviembre YA TIENE sus lags correctos (calculados desde octubre)

2. Predice el día 1

3. Para el día 2: actualiza lag_1 con la predicción del día 1, desplaza lag_2←lag_1_anterior, lag_3←lag_2_anterior, etc.

4. Actualiza unidades_vendidas_ma7 con las últimas 7 predicciones

5. Predice el día 2 con los lags actualizados

6. Repite para los 30 días de noviembre

Estructura de la app:

*La aplicación debe tener un **sidebar** a la izquierda con controles de simulación y la **zona principal** mostrando el dashboard de resultados.*

En el sidebar necesito:

- Título "Controles de Simulación"

- Selector de producto (dropdown con los 24 productos por nombre)

- Slider de ajuste de descuento (-50% a +50%, pasos de 5%, inicial en 0%)

- Selector de escenario de competencia (3 opciones con radio buttons: "Actual (0%)", "Competencia -5%", "Competencia +5%")

- Botón grande de "Simular Ventas" para ejecutar la predicción

En la zona principal debe aparecer:

*1. **Header**: Título del dashboard indicando que es simulación para noviembre 2025 y el producto seleccionado*

*2. **KPIs destacados** (4 tarjetas en fila):*

  - Unidades totales proyectadas

  - Ingresos proyectados

  - Precio promedio de venta

  - Descuento promedio

*3. **Gráfico de predicción diaria**: Una única línea mostrando las unidades vendidas predichas para cada día de noviembre (días 1-30 en eje X). El gráfico debe marcar visualmente el Black Friday (día 28) con una línea vertical destacada, un punto resaltado en rojo y una anotación clara. NO incluyas múltiples líneas ni leyendas confusas, solo la predicción de ventas.*

*4. **Tabla detallada**: Tabla interactiva con todos los días de noviembre mostrando fecha, día de la semana, precio venta, precio competencia, descuento aplicado, unidades predichas e ingresos proyectados por día. Destaca visualmente la fila del Black Friday (28 de noviembre) con un emoji o color.*

*5. **Comparativa de escenarios**: Sección que compare los 3 escenarios de competencia (sin cambios, -5%, +5%) mostrando en tarjetas las unidades totales e ingresos de cada escenario. Mantén el descuento del usuario y solo varía la competencia.*

Lógica que debe implementar:

Cuando el usuario cambie los controles y pulse "Simular Ventas":

1. Filtrar el dataframe para el producto seleccionado (solo datos de noviembre)

2. Recalcular variables dependientes basándote en los controles:

  - precio_venta según el descuento elegido sobre precio_base

  - precio_competencia ajustando las columnas Amazon, Decathlon y Deporvillage según el escenario

  - descuento_porcentaje y ratio_precio recalculados

*3. **Hacer predicciones RECURSIVAS día por día:**

- Ordenar el dataframe por fecha

- Para el día 1: usar los lags que ya están en el archivo (fueron calculados desde octubre)

- Predecir el día 1

- Para el día 2 y siguientes:

* Actualizar unidades_vendidas_lag_1 = predicción del día anterior

* Desplazar lags: lag_2←valor anterior de lag_1, lag_3←valor anterior de lag_2, etc.

* Actualizar unidades_vendidas_ma7 = promedio de las últimas 7 predicciones

* Predecir ese día con los lags actualizados

4. Actualizar todo el dashboard con los resultados

Aspectos técnicos importantes:

- El dataframe ya tiene TODAS las columnas necesarias, NO hagas transformaciones adicionales

- Los nombres exactos de las columnas de lag son: unidades_vendidas_lag_1, unidades_vendidas_lag_2, ..., unidades_vendidas_lag_7

- La media móvil se llama: unidades_vendidas_ma7

- NO intentes separar octubre/noviembre, solo hay noviembre en el archivo

- Usa las columnas exactas que el modelo espera (están en modelo.feature_names_in_)

- Maneja errores si no se pueden cargar el modelo o datos

- Añade spinners mientras procesa las predicciones recursivas

- Incluye mensajes informativos para guiar al usuario

- Formatea bien los números (2 decimales precios, 0 decimales unidades, formato moneda para euros)

- Usa separadores visuales entre secciones

Diseño visual:

Haz la app moderna y atractiva. Usa colores, emojis en títulos, tarjetas bien diseñadas con st.metric() y gráficos limpios. La paleta puede ser morada/azul (#667eea, #764ba2).

Antes de darme el código final, analízalo cuidadosamente para asegurar que:

- Todas las variables están definidas antes de usarse

- Las columnas del dataframe existen (incluyendo unidades_vendidas_lag_1 a unidades_vendidas_lag_7, unidades_vendidas_ma7)

- NO intentas acceder a datos de octubre (no existen en el archivo)

- Los lags del día 1 de noviembre se usan tal cual están en el archivo

- Los lags de los días 2-30 se actualizan recursivamente con las predicciones previas

- El modelo hace predicciones día por día actualizando lags después de cada predicción

- El gráfico muestra UNA SOLA línea de predicción con el Black Friday marcado

- La app se ejecuta correctamente con `streamlit run app.py`