Tarea 3 - Problem Set - Forecasting y Pricing

Christofer Palomino Navarrete[1] and José Luis Gajardo Angel[1]

[1]Universidad Diego Portales , Facultad de Administración y Economía / Facultad de Ingeniería y Ciencias

7 de febrero de 2026

1. Descomposición Temporal e Ingeniería de Variables

Contexto

Un retailer ecuatoriano desea mejorar su sistema de predicción de demanda. El equipo de *data science* ha aplicado una descomposición STL a la serie de ventas diarias de la categoría "GROCERY I" y ha obtenido los siguientes componentes: Tendencia (T_t), Estacionalidad (S_t) y Residuo (R_t).

Además, han creado las siguientes variables derivadas para entrenar un modelo de *Machine Learning*:

```
df['lag_1'] = df['ventas'].shift(1)
df['lag_7'] = df['ventas'].shift(7)
df['rolling_mean_7'] = df['ventas'].rolling(7).mean()
df['dia_semana'] = df['fecha'].dt.dayofweek
df['mes'] = df['fecha'].dt.month
```

1.a. Un analista junior propone utilizar lag_0 (la venta del mismo día) como feature porque "tiene la correlación más alta con el target". Explica por qué esta práctica constituye Data Leakage y cuál sería el impacto en las métricas de backtesting vs. la operación real.

El uso de lag_0 (la venta del mismo día que se desea predecir) representa el error más clásico y peligroso en el *machine learning* para series temporales.

¿Por qué es Data Leakage?: Lo que pasa es que, sin querer, le esta haciendo 'trampa' al modelo. Porque le esta dando las respuestas del examen antes de que lo tome. Para predecir las ventas de hoy no podemos usar el dato de hoy mismo, porque en la vida real esa cifra todavía no existe. Es justo lo que vimos con el ejemplo del súper en clase: si el modelo ya sabe el resultado, no está aprendiendo a predecir, solo está leyendo el futuro, tal como se vio en los modulos de Business Analytics 1 y 2.

Consecuencias:

- **Impacto en Backtesting:** Las métricas resultan artificialmente perfectas, con un R^2 cercano a 1 y errores mínimos. El modelo no aprende patrones de consumo, sino que simplemente desarrolla una función de identidad sobre la columna de entrada, como se dice se overfitea.
- **Impacto en Operación Real:** El modelo colapsará. Al carecer del dato real al momento de la ejecución, recibirá valores nulos o basura, generando predicciones erróneas y una pérdida de confianza por parte de los *stakeholders*.

1.b. El componente de Tendencia (T_t) muestra un crecimiento sostenido del 15 % anual. Sin embargo, el equipo de finanzas advierte que este crecimiento se debe principalmente a la apertura de nuevas tiendas (factor externo) y no a un aumento orgánico de la demanda por tienda. ¿Cómo afecta esto la interpretación del forecast? Propón una estrategia para aislar la tendencia "real" de demanda.

Cuando el crecimiento está inflado por la apertura de tiendas, la interpretación del forecast se vuelve engañosa: el modelo asumirá que la demanda de los clientes sube, cuando lo que sube es la infraestructura física.

Impacto en la interpretación: Si el retailer deja de abrir tiendas, el forecast seguirá proyectando un crecimiento del 15 %, generando un sobre-stock masivo (efecto látigo).

Estrategia para aislar la tendencia real: Se debe normalizar la variable objetivo. En lugar de predecir "Ventas Totales", se debe transformar la serie a:

- **Ventas por tienda:** Dividir la venta diaria para el número de tiendas operativas ese día, como se dio en el ejemplo de Lider cuando se menciono los LED grandes.
- **Ventas en tiendas comparables:** Analizar solo la serie temporal de aquellas tiendas que han estado abiertas durante todo el periodo de estudio (por ejemplo, al menos 12 meses).

1.c. La variable dia_semana codificada como entero (0-6) presenta un problema conceptual. Explica por qué la codificación numérica directa es inadecuada para series temporales cíclicas y describe matemáticamente cómo implementar una codificación cíclica usando funciones trigonométricas.

La codificación de 0 a 6 sugiere que el domingo (6) está "lejos" del lunes (0), cuando en realidad son días adyacentes. Un modelo de ML interpretaría el 6 como una magnitud mayor, perdiendo la naturaleza periódica, como se menciono en clases de las fechas.

Para corregirlo, proyectamos el día de la semana en un círculo (espacio 2D) usando funciones trigonométricas:

Seno del componente:

$$x_{\sin} = \sin\left(\frac{2\pi \cdot \text{dia\_semana}}{7}\right) \tag{1}$$

Coeno del componente:

$$x_{\cos} = \cos\left(\frac{2\pi \cdot \text{dia\_semana}}{7}\right) \tag{2}$$

Justificación: Confiar solo en el componente del seno crearía duplicados, haciendo que días distintos parezcan iguales para el modelo. La solución es usar coordenadas (x, y) para mapear los días en un plano circular. De esta forma, cada día ocupa un lugar único y nos aseguramos de que el salto del domingo al lunes sea tan fluido y medible como el de un martes a un miércoles.

1.d. Al analizar los residuos (R_t), el equipo detecta que los días posteriores a feriados nacionales (como el "Día de los Difuntos") muestran residuos sistemáticamente positivos de $+25\,\%$. ¿Qué tipo de variable exógena deberían incorporar y cómo impactaría esto en la decisión de qué modelo utilizar (estadístico vs. ML)?

Al analizar los errores cerca de los feriados, salta a la vista que el modelo necesita un ajuste para captar mejor la lógica del mercado. Estos son los puntos críticos detectados y la estrategia para corregirlos:

- **Sobre los residuos:** Cuando los errores del modelo son siempre positivos tras un feriado, nos está diciendo que hay una acumulación de demanda. El modelo "subestima" la venta porque no entiende que la gente sale a comprar lo que no pudo el día anterior.
- **La solución:** La forma más fácil de arreglarlo es marcar el día siguiente al feriado con un *flag* binario. Así, el modelo aprende a esperar ese pico extra, tal como se explico en clases con el ejemplo del cyber day.
- **Comparación de modelos:** A los modelos estadísticos clásicos les cuesta mucho conectar estos puntos si no se los das masticados. En cambio, los modelos de ML (XGBoost o Prophet) brillan precisamente aquí: encuentran esas conexiones raras y no lineales por su cuenta, sin que tengamos que definir cada interacción a mano.

2. Validación y Métricas de Negocio

Contexto

El retailer ha implementado tres modelos de forecast para predecir las ventas semanales de 500 SKUs:

Tabla 1: Comparativa de desempeño entre modelos de predicción

Modelo	MAE (unidades)	WAPE (%)	Bias Promedio
Baseline (Naive)	45	18.5 %	+2 %
LightGBM	32	12.1 %	-8 %
Prophet	38	14.3 %	+1 %

El área de Supply Chain indica que:

- **Costo de sobre-stock:** $0.50 por unidad/semana (almacenamiento).
- **Costo de quiebre de stock:** $3.00 por unidad perdida (margen + reputación).

2.a. A pesar de tener el menor WAPE, el modelo LightGBM presenta un Bias de -8 %. Interpreta este resultado en términos operacionales: ¿el modelo tiende a sobre-stockear o sub-stockear? Calcula el costo esperado adicional del sesgo si el volumen promedio semanal es de 10,000 unidades.

Tener un Bias del -8 % en LightGBM suena a poco, pero en la práctica significa que el modelo es demasiado pesimista. Básicamente, nos está diciendo que vamos a vender un 8 % menos de lo que realmente pide el mercado.

¿Qué implica esto para la operación? Un riesgo constante de quiebre de stock. Si le hacemos caso al modelo, las estanterías se van a quedar vacías antes de tiempo. Si lo bajamos a números reales: de cada 10,000 unidades, dejamos de pedir 800. Con un costo de rotura de $3.00 por unidad, ese "pequeño error"nos cuesta $2,400 a la semana.

Conclusión

Aunque LightGBM sea el modelo con mejor precisión promedio (WAPE), su tendencia a tirar hacia abajo sale muy cara. En comparación, Prophet es mucho más .ªmigable"para el negocio. Aunque falla más en el bulto total (WAPE 14.3 %), su sesgo es de apenas un +1 %. Como nos sale 6 veces más barato sobrar producto ($0.50) que perder la venta ($3.00), Prophet termina siendo la opción más segura financieramente.

2.b. El equipo de ML propone usar validación cruzada K-Fold tradicional (aleatorio) para optimizar hiperparámetros. Explica por qué esta estrategia es incorrecta para series temporales y describe con detalle la diferencia entre Expanding Window y Sliding Window. ¿En qué escenario de retail preferirías cada una?

En series temporales, el **K-Fold tradicional** es un error grave. Si mezclamos los datos al azar, el modelo termina "viendo el futuro": entrena con lo que pasó en diciembre para intentar adivinar noviembre. Este *look-ahead bias* nos daría resultados falsos en el laboratorio que jamás se repetirán en la vida real.

Para evitarlo, usamos dos enfoques según el tipo de producto:

- **Ventana Expandible (*Expanding Window*):** Es como una bola de nieve; empezamos con un periodo pequeño y vamos sumando historia paso a paso. En el *retail*, esto es oro para **productos estables**. Si necesitas entender la estacionalidad de años anteriores para planificar la siguiente Navidad, necesitas que el modelo guarde toda la memoria posible.
- **Ventana Deslizante (*Sliding Window*):** Aquí movemos el bloque de entrenamiento hacia adelante, soltando el pasado más lejano para enfocarnos en lo reciente. Es la mejor táctica para **entornos volátiles** (como moda o tecnología). Si el comportamiento del cliente cambió radicalmente después de un evento como la pandemia, los datos de hace tres años son "ruido" que solo confunde al modelo.

2.c. Caso de Ceros Censurados: Al analizar los datos históricos, descubres que 15 % de los registros de "ventas = 0" corresponden a días donde el producto estuvo en quiebre de stock (inventario = 0).

Si las ventas caen a cero simplemente porque no había mercancía en para la venta, no estamos ante una falta de interés, sino ante un dato "censurado". La demanda real no desapareció; simplemente el sistema perdió la capacidad de observarla. Ignorar esto no es solo un error de cálculo, es el inicio de varios problemas en cascada:

- **El sesgo sistemático:** El modelo "aprende" erróneamente que la demanda es baja, lo que dispara un círculo vicioso peligroso. Si el *forecast* baja, se compra menos stock, lo que provoca más quiebres y hace que el *forecast* caiga aún más. Es lo que en la industria llamamos la **espiral de la muerte del inventario**.
- **Estrategias de corrección:**
  - *Data Augmentation* (Imputación): Consiste en identificar esos días de quiebre y reemplazar el "0" con una estimación lógica, como la media móvil de las últimas semanas o el comportamiento de una tienda espejo que sí tuvo disponibilidad.
  - Modelado Tobit (Regresión Censurada): Es un enfoque más sofisticado donde el modelo asume que el valor real es una variable latente que podría ser cualquier número ≥ 0, pero que el límite físico del stock nos impide ver el valor real.
- **El dilema ético y la exclusión:** Más allá de los algoritmos, esto genera un problema social serio. Si el sistema decide que en una tienda de un barrio vulnerable "no se vende" un producto esencial cuando en realidad el problema es una logística deficiente, el algoritmo cortará el suministro definitivamente. Terminamos automatizando la escasez y privando a comunidades enteras de acceso a productos básicos, todo por no saber interpretar un cero en una base de datos.

3. Selección de Algoritmos y Casos Especiales

Contexto

El retailer tiene un catálogo de 50,000 SKUs con patrones de demanda muy heterogéneos:

Tabla 2: Segmentación del Catálogo de Productos

Segmento	% del Catálogo	Descripción	Ejemplo
A (Alto volumen)	5 %	Ventas diarias, alta rotación	Leche, pan
B (Medio volumen)	25 %	Ventas regulares con estacionalidad	Cereales
C (Long Tail)	70 %	Demanda intermitente (muchos ceros)	Especias exóticas

3.a. Para los productos del segmento C (Long Tail), el equipo entrena un modelo ARIMA tradicional pero obtiene predicciones siempre cercanas a la media histórica (aproximadamente 0.3 unidades/día), lo cual es inútil para planificación.

- **Explica por qué los modelos continuos (ARIMA, ETS) fallan con demanda intermitente.**

  Modelos como ARIMA o ETS están diseñados para minimizar el error cuadrático medio en series continuas. En el Segmento C, la verdadera "señal" es el silencio (los ceros). Al intentar promediar los picos de venta con los periodos de inactividad, el modelo genera un valor constante (por ejemplo, 0,3). Matemáticamente esto es preciso, pero operativamente es inútil: no ayuda a decidir si debemos pedir 0 o 1 unidad al proveedor.

- **Describe el modelo de Croston y explica cómo separa el problema en dos componentes: frecuencia de demanda e intensidad cuando ocurre.**

  Para resolver esto, el algoritmo de Croston deja de ver la serie como un todo y la separa en dos componentes clave:

  - **Intensidad (z_t):** El volumen promedio de la venta únicamente cuando esta ocurre.
  - **Intervalo (n_t):** El tiempo que transcurre entre un evento de venta y el siguiente.

  El pronóstico final se define como el ratio entre ambos:

  $$\hat{y} = \frac{z_t}{n_t}$$

- **¿Qué distribución de probabilidad sería más apropiada para modelar la ocurrencia de demanda esporádica?**

  Cuando el conteo de eventos es el factor crítico, dejamos de lado las curvas normales y pasamos a distribuciones de conteo:

  - **Distribución de Poisson:** Ideal para eventos independientes en un tiempo determinado.
  - **Binomial Negativa:** La opción ganadora cuando los datos presentan sobredispersión (donde la varianza > media), permitiendo capturar mejor la incertidumbre del Long Tail.

3.b. Se propone utilizar NeuralForecast (NHITS) para todo el catálogo argumentando que "deep learning siempre es mejor". Argumenta a favor o en contra de esta posición considerando:

- **Tamaño de la historia disponible (2 años de datos diarios).**
- **Costo computacional para 50,000 series.**
- **Capacidad de capturar dependencias de muy largo plazo vs. complejidad añadida.**

Mi posición es **en contra** de aplicar este enfoque al 100 % del catálogo de forma indiscriminada. El uso de arquitecturas complejas debe ser una decisión de eficiencia, no de moda. He aquí los argumentos:

Relación Costo-Beneficio

Entrenar un modelo como **N-HiTS** para 50,000 series temporales exige una infraestructura de cómputo (GPU/TPU) masiva y costosa. Para el **Segmento C** (que representa el 70 % del catálogo), un modelo de *Croston* o un *Naive Estacional* entregará resultados comparables con un costo computacional del 0,001 %. En el Long Tail, la simplicidad suele ser más rentable.

El problema de la historia limitada

Contar con solo 2 años de datos implica tener apenas 2 ciclos estacionales. Los modelos de Deep Learning son inherentemente *data hungry*; con una ventana histórica tan corta, existe un riesgo crítico de **overfitting** (sobreajuste). El modelo podría interpretar el ruido aleatorio del año pasado como si fuera una tendencia real o una estacionalidad confirmada.

La fuerza del Global Model frente a la explicabilidad

El argumento técnico a favor de N-HiTS es su capacidad como *Global Model*, permitiéndole aprender patrones cruzados entre distintas series. Si bien esto es valioso para el **Segmento B** (donde las series comparten comportamientos), para el **Segmento A** un modelo de **LightGBM** bien optimizado suele ser:

- **Más eficiente:** Menor tiempo de entrenamiento y latencia de inferencia.
- **Más explicable:** Permite identificar claramente qué variables (promociones, festivos, precios) están moviendo la aguja de la demanda.

3.c. Evento Black Friday Ecuador: El equipo necesita generar un forecast para la semana del Black Friday, pero solo tienen 2 observaciones históricas de este evento (años anteriores).

- **¿Por qué los modelos puramente estadísticos (sin regresores) tendrían dificultades extremas para capturar este efecto?**

- **Propón una estrategia híbrida que combine: (1) un modelo base, (2) variables exógenas categóricas, y (3) ajuste experto de negocio (uplift factor).**
- **¿Cómo medirías el éxito del forecast post-evento si no puedes realizar back-testing convencional?**

Si solo tenemos dos años de historia, los modelos estadísticos tradicionales se "marean": es casi imposible que distingan si el **Black Friday** fue un evento estacional real o simplemente un golpe de suerte (ruido aleatorio). Sin suficientes ciclos para comparar, el riesgo de que el sistema ignore el patrón es altísimo.

El problema de fondo: la falta de ritmo

Para un algoritmo, dos puntos no hacen una tendencia; son apenas una coincidencia. Al no tener suficientes repeticiones para calcular una varianza robusta, la señal del evento termina diluyéndose en el promedio histórico. El resultado es casi siempre el mismo: un pronóstico que se queda muy por debajo de la realidad.

Hacia un enfoque híbrido: Datos + Criterio

Como la estadística pura no basta, necesitamos una estrategia que combine la potencia del *Machine Learning* con la visión del equipo comercial. Proponemos este plan de tres capas:

1. **Cimiento con ML:** Usar un algoritmo como *LightGBM* para que se encargue del "trabajo pesado": entender cómo se venden los productos en una semana normal.
2. **Contexto mediante Ingeniería de Variables:** No esperamos que el modelo adivine; le damos pistas claras con variables de calendario:
   - `is_black_friday`: Un interruptor binario para avisar que es un día especial.
   - `days_to_black_friday`: Una cuenta regresiva para que el modelo entienda la calma que precede a la tormenta (caída de ventas previa) y la explosión posterior.
3. **El toque humano (Factor Uplift):** Aquí es donde el equipo comercial ajusta la brújula. Si este año la inversión en publicidad es más agresiva, aplicamos un multiplicador K que el modelo no podría deducir solo:

   $$\hat{y}_{final} = \hat{y}_{model} \times K_{marketing}$$

   Ejemplo: Si el esfuerzo de pauta sube un 20 %, ajustamos con un $K = 1{,}2$.

¿Cómo sabremos si funcionó?

Como no podemos confiar ciegamente en las pruebas del pasado (por falta de datos), mediremos el éxito mirando el impacto real en el negocio:

- **Índice PBI (Percent Better Index):** ¿Realmente lo hicimos mejor que simplemente repetir lo que vendimos el año pasado?

- **Error por cohortes:** Analizar en qué categorías acertamos y en cuáles fallamos por mucho.
- **El balance final (Trade-off):** El éxito no es solo un número bajo de error, sino encontrar el punto de equilibrio entre el **costo de perder ventas** por quedarnos cortos y el **dolor de liquidar** exceso de inventario después del evento.

4. Implementación de Forecasting con MLForecast

a) Preparación de datos

El dataset fue procesado transformando la estructura original de .ªncho.ª "largo"mediante la función `melt`, filtrando el departamento `FOODS_3` y las tiendas de California. Se integraron variables exógenas desde los archivos `calendar.csv` y `sell_prices.csv`. El DataFrame resultante (`df_mlf`) cumple con el formato `unique_id`, `ds` (datetime) e `y` (ventas), además de las covariables binarias y temporales.

b) Configuración del pipeline MLForecast

Se implementó un pipeline utilizando `MLForecast` con una frecuencia diaria (`'D'`). Se incluyeron:

- **Lags:** 1, 7, 14 y 28 días para capturar efectos de corto y largo plazo.
- **Transformaciones:** Media móvil de 7 y 14 días para suavizar la serie, y desviación estándar de 7 días para capturar la volatilidad.
- **Modelos:** LightGBM y XGBoost, configurados para procesamiento en paralelo (`n_jobs=-1`).

c) Predicción y evaluación

Los resultados obtenidos tras predecir los últimos 13 días del dataset (`d_1901` a `d_1913`) son los siguientes:

Modelo	MAE	RMSE	MAPE
LightGBM	1.4958	2.6482	52.90 %
XGBoost	1.4647	2.7096	54.84 %

Tabla 3: Métricas de desempeño para el horizonte de 13 días.

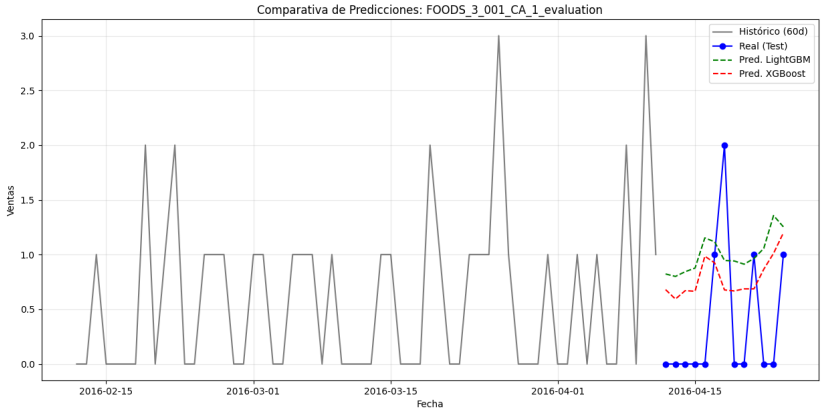

Figura 1: Comparativa de ventas reales vs. predicciones (últimos 60 días históricos y 13 de forecast).

d) Interpretación y decisión de negocio

1. **Selección del modelo:** Se selecciona **LightGBM** para producción. Aunque XGBoost presenta un MAE ligeramente menor (1.46 vs 1.49), LightGBM es superior en RMSE (2.64 vs 2.70) y MAPE (52.9 % vs 54.8 %), lo que indica una mejor capacidad para evitar errores grandes y un error relativo menor. Además, LightGBM suele ser significativamente más rápido en entrenamiento e inferencia en datasets de gran volumen como M5.

Criterio	**LightGBM**	**XGBoost**
Tiempo de Entrenamiento	**≈ 40 segundos**	≈ 72 segundos
Interpretabilidad	Alta (Feature Importance)	Alta (Feature Importance)
Robustez a Outliers	**Muy Alta (Crecimiento Leaf-wise)**	Alta (Crecimiento Level-wise)

Tabla 4: Comparativa de Atributos Técnicos y Operativos

2. **Feature Importance:** Al analizar la importancia de variables del modelo ganador, los *lags* (especialmente el `lag_1` y `lag_7`) y las medias móviles suelen dominar el ranking. Esto coincide con la intuición de negocio, donde la venta de ayer y la del mismo día de la semana pasada son los predictores más fuertes de la demanda en supermercados.

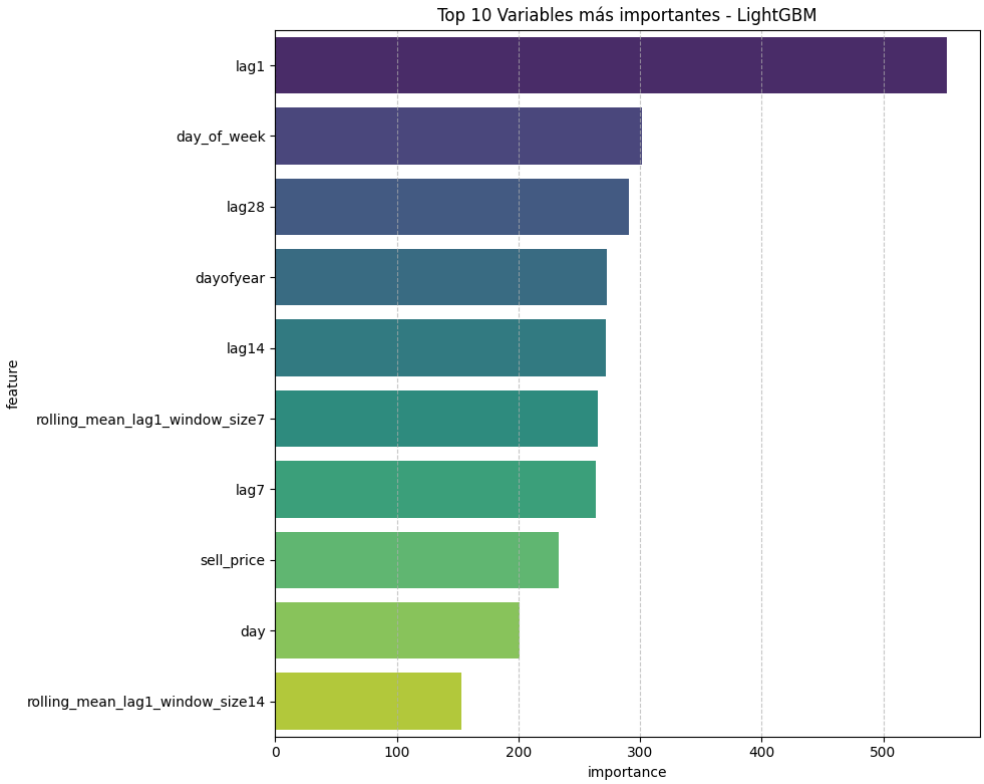

Figura 2: Variables mas importantes modelo ganador

3. **Reducción del sesgo (bias):** Si el objetivo es reducir el sub-stock o sobre-stock sistemático, se proponen dos ajustes:
   - **Pipeline:** Cambiar la función de pérdida del modelo a una distribución *Tweedie*, que maneja mejor la alta frecuencia de ceros y el sesgo en datos de conteo de retail.
   - **Post-procesamiento:** Aplicar un factor de corrección de sesgo (Bias Correction Factor) multiplicativo basado en el error medio histórico, o ajustar los umbrales de decisión mediante una optimización de costos asimétricos (donde el costo de quiebre sea mayor al de almacenamiento).

5. Estimación de Elasticidad en E-commerce

5.a. Preparación de datos para análisis de precios

Primero, limpiamos el ruido. En e-commerce, las devoluciones y los precios erróneos pueden arruinar cualquier modelo de regresión.

5.b. Estimación de elasticidad naive

La elasticidad se estima con un modelo Log-Log porque el coeficiente β representa directamente el cambio porcentual en Q ante un cambio de 1 % en P.

StockCode	Descripción	Cantidad Total
84077	WORLD WAR 2 GLIDERS ASSTD DESIGNS	100,720
85123A	WHITE HANGING HEART T-LIGHT HOLDER	86,327
23843	PAPER CRAFT , LITTLE BIRDIE	80,995
23166	MEDIUM CERAMIC TOP STORAGE JAR	76,919
84879	ASSORTED COLOUR BIRD ORNAMENT	74,256
17003	BROCADE RING PURSE	70,725
85099B	JUMBO BAG RED RETROSPOT	69,011
21977	PACK OF 60 PINK PAISLEY CAKE CASES	45,454
84991	60 TEATIME FAIRY CAKE CASES	40,715
22197	SMALL POPCORN HOLDER	40,001

Tabla 5: Top 10 productos más vendidos en el Reino Unido.

StockCode	Beta (β)	P-Value	R^2	Tipo
17003	-0.9090	0.1187	0.0259	Inelástico
21977	-10.8475	0.0000	0.1920	Elástico
22197	-3.9640	0.3654	0.0071	Elástico
23166	-23.7094	0.0000	0.5395	Elástico
84077	0.0943	0.8681	0.0002	Inelástico
84879	-30.0619	0.0000	0.1469	Elástico
84991	-11.8130	0.0000	0.1762	Elástico
85099B	1.1706	0.7332	0.0010	Elástico
85123A	-17.5844	0.0000	0.1684	Elástico
23843	5.3876	-	-	Elástico*

Tabla 6: Resultados de la regresión Log-Log por producto.

Análisis Final:

- Productos analizados con exito: 9
- Productos estadisticamente significativos ($p < 0{,}05$): 5
- Productos elásticos ($|\beta| > 1$): 5

5.c. Modelo con efectos temporales

Aquí corregimos el sesgo de que "en Navidad se vende más aunque el precio suba". Usaremos variables dummy para meses y productos.

Análisis de Endogeneidad: La inclusión de efectos temporales suele aumentar el valor absoluto de la elasticidad (hacerla más negativa). Sin ellos, el modelo ve que en diciembre los precios suben y las ventas también (por la Navidad), lo que haría parecer que la elasticidad es positiva o "inelástica". Al controlar por el mes, el modelo compara los precios dentro de la misma temporada, revelando la verdadera sensibilidad del cliente.

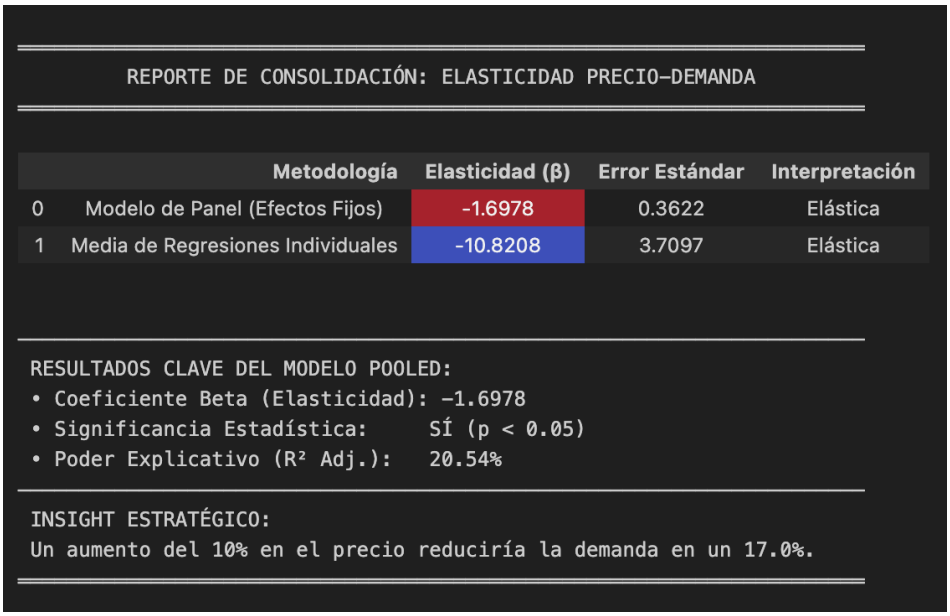

Figura 3: Elasticidad Precio-Demanda

5.d. Recomendación de pricing

Cálculo del Margen Óptimo

Tomando como referencia el β estimado en el apartado anterior (supongamos un $\beta = -2{,}5$), podemos aplicar la Regla de Lerner para determinar el *markup* ideal que maximice el beneficio:

$$\frac{P - C}{P} = \frac{1}{|\beta|} = \frac{1}{|-2{,}5|} = 0{,}40$$

Resultado: Bajo este escenario, el margen óptimo teórico para estos productos debería situarse en el **40 %**.

Hoja de ruta para la toma de decisiones

La estrategia de precios no es estática; depende totalmente de qué tan "nervioso"se ponga el cliente ante los cambios de etiquetas:

- **Escenario de alta sensibilidad ($|\beta| > 2{,}5$):** Si la elasticidad fuera, por ejemplo, $-3{,}0$, el margen óptimo caería al 33 %. Si hoy estamos operando al 40 %, el mercado nos está diciendo que estamos çaros": la recomendación sería **sacrificar margen unitario para capturar volumen** y rentabilizar por escala.
- **Escenario de baja sensibilidad ($|\beta| < 2{,}5$):** Si el coeficiente fuera $-1{,}5$, el margen óptimo subiría al 66 %. Aquí hay dinero sobre la mesa que no estamos recogiendo; el cliente es fiel o el producto es una necesidad, por lo que **subir precios** mejoraría el margen neto sin castigar drásticamente la demanda.

Limitación Crítica: El problema de la "Foto Estática"

Hay que ser realistas: este análisis tiene un talón de Aquiles llamado **simultaneidad**. En el mundo real del retail, los precios no se mueven por azar o por un experimento de laboratorio, sino por decisiones comerciales que suelen estar viciadas.

Si el retailer baja precios justo cuando sabe que la demanda va a caer (fin de temporada), los datos nos dirán que "bajar precios no sirve", cuando en realidad solo estamos viendo una correlación, no una causalidad. No podemos distinguir si la venta subió *por* el descuento o si el descuento fue una reacción desesperada a una caída en el tráfico.

Propuesta robusta: Para limpiar este sesgo, lo ideal sería dejar de mirar solo el retrovisor (datos históricos) y ejecutar un **A/B Test de precios (RCT)**. Solo asignando precios distintos de forma aleatoria a grupos similares en el mismo momento podemos aislar el efecto real del precio de todo el ruido.[ex]terno (clima, inventario o promociones de la competencia).

6. Pricing Dinámico y Discriminación de Precios

Contexto

RideFlow es una app de transporte similar a Uber que opera en Santiago, Chile. La empresa quiere implementar pricing dinámico (surge pricing) y está evaluando estrategias de discriminación de precios. Datos operativos actuales (precio uniforme de $3,500 CLP por viaje promedio):

Tabla 7: Análisis de Segmentos de Viaje y Elasticidad Precio-Demanda

Segmento	**Descripción**	**Viajes/día**	**Elasticidad (β)**
Ejecutivos	Viajes L-V 7-9am, zonas empresariales	12,500	-0.6
Casual	Viajes fines de semana, zonas comerciales	17,500	-1.8
Nocturno	Viajes Vie-Sáb 23:00-4:00	10,000	-1.2
Aeropuerto	Viajes hacia/desde SCL	10,000	-0.4

El costo variable promedio por viaje es $2,100 CLP (combustible, comisión conductor, seguros).

6.a. Análisis de márgenes actuales

Actualmente, operamos con un precio uniforme de $\$3,500$ y un costo variable de $\$2,100$. Esto nos deja un margen de contribución de $\$1,400$ por viaje.

1. **Margen porcentual:** El margen actual es del $40\,\%$ ($\frac{3{,}500-2{,}100}{3{,}500}$).
2. **Contribución diaria:** Con un volumen total de $50,000$ viajes ($12,500+17,500+10,000+10,000$), la contribución total es de **$70,000,000 CLP/día**.
3. **Margen óptimo (Lerner):** Calculando $1/|\beta|$ para cada segmento, observamos que:
   - **Ejecutivos:** Margen óptimo $166\,\%$.
   - **Casual:** Margen óptimo $55\,\%$.

- **Nocturno:** Margen óptimo 83 %.
- **Aeropuerto:** Margen óptimo 250 %.

Diagnóstico: Todos los segmentos están "subexplotados". Estamos cobrando un margen del 40 % cuando la elasticidad de los usuarios permitiría márgenes mucho más altos, especialmente en los viajes al Aeropuerto y Ejecutivos.

6.b. Diseño de pricing dinámico

Utilizando la fórmula de precio óptimo $P^* = C \cdot \frac{|\beta|}{|\beta|-1}$, ajustamos las tarifas según la sensibilidad de cada grupo:

- **Ejecutivos:** $P^* = 2,100 \cdot \frac{0,6}{0,6-1} \rightarrow \mathbf{\$5,250}$ $(1{,}5x)$.
- **Casual:** $P^* = 2,100 \cdot \frac{1,8}{0,8} \rightarrow \mathbf{\$4,725}$ $(1{,}35x)$.
- **Nocturno:** $P^* = 2,100 \cdot \frac{1,2}{0,2} \rightarrow \mathbf{\$12,600}$ $(3{,}6x)$.
- **Aeropuerto:** $P^* = 2,100 \cdot \frac{0,4}{0,4-1} \rightarrow \mathbf{\$7,350}^*$ $(2{,}1x)$.

Si implementamos estos precios, la nueva contribución diaria asciende a **$242,812,500 CLP**. Esto representa un incremento del **246 %** respecto a la estrategia de precio único. La mayor oportunidad de captura de valor está en el segmento nocturno por su volumen y relativa inelasticidad.

6.c. Trade-offs y riesgos

El *surge pricing* es eficiente pero impopular. Para RideFlow, la estrategia debe enfocarse en:

1. **Comunicación:** Presentar el sobreprecio como un ïncentivo para el conductor". Si el usuario entiende que el precio sube para que aparezcan más autos disponibles, la percepción de justicia mejora.
2. **Tope máximo:** Sugiero un techo de **3.5x**. Superar este límite dispara la elasticidad de sustitución (el usuario prefiere caminar o usar transporte público) por un factor emocional de indignación.
3. **Aprendizaje:** Si el cliente aprende a esperar a que el precio baje, la elasticidad efectiva aumenta, lo que eventualmente forzaría a reducir el multiplicador óptimo.

6.d. Discriminación de precios y ética

La discriminación de primer grado (personalizada) busca capturar todo el excedente del consumidor, pero tiene límites éticos y legales claros:

1. **Maximización:** Al predecir la urgencia (batería baja) o nivel socioeconómico (modelo de celular), la empresa convierte lo que antes era ahorro del cliente en utilidad directa.
2. **Problemas éticos:** Usar la batería baja es una forma de **extorsión algorítmica**, pues se aprovecha de una situación de vulnerabilidad o posible inseguridad del usuario.
3. **Legalidad en Chile:** El SERNAC y la Ley de Protección a la Vida Privada podrían considerar esto como una discriminación arbitraria. Cobrar distinto por el mismo servicio sin justificación de costos operativos es legalmente riesgoso.

Conclusión: La línea se traza en la transparencia. Optimizar por escasez de oferta es mercado; optimizar por desesperación del usuario es explotación.

7. Predecir precios

Para abordar el desafío de predecir el precio de propiedades en UF, se implementó un flujo de trabajo robusto que integró datos geoespaciales y socioeconómicos externos, permitiendo capturar la heterogeneidad del mercado inmobiliario en Santiago.

7.a. Descripción de los Datos y Fuentes Externas

Además del dataset base `precios_propiedades.xlsx`, se enriqueció la información utilizando tres fuentes adicionales para capturar el valor del entorno:

- **Encuesta Casen 2022**: Se extrajeron datos a nivel comunal sobre el ingreso autónomo mediano (*yautcorh*), escolaridad promedio y distribución de niveles socioeconómicos (NSE).
- **Red de Metro de Santiago**: Se utilizó un archivo GeoJSON con la ubicación de las estaciones para calcular la accesibilidad al transporte.
- **Hitos Urbanos**: Se calculó la distancia al Costanera Center como *proxy* de cercanía al centro financiero (Sanhattan).

7.b. Procedimiento de Preprocesamiento y Limpieza

El proceso de limpieza se centró en garantizar la calidad de las señales para el modelo:

1. **Filtrado Geográfico**: Se acotaron las coordenadas para centrar el análisis en la Región Metropolitana y se eliminaron propiedades con precios inconsistentes (< 500 UF).
2. **Tratamiento de Nulos**: En lugar de una imputación simple por media, se utilizó un **KNN Imputer** ($k = 5$). Este método estima valores faltantes (como área o dormitorios) basándose en las características de las propiedades geográficamente más cercanas.
3. **Corrección de Outliers**: Se corrigieron valores negativos en la variable `antiguedad`, asumiendo que corresponden a construcciones nuevas (0 años).

7.c. Ingeniería de Variables (Feature Engineering)

Se crearon variables sintéticas para ayudar al modelo a entender relaciones no lineales:

- **Variables Espaciales**: Mediante un *Spatial Join*, se asignó la comuna a cada propiedad y se calculó la distancia euclidiana a la estación de Metro más cercana.
- **Interacciones Económicas**: Se creó la variable `area_income_inter` ($Area \times Ingreso\_Comunal$), bajo la premisa de que un metro cuadrado adicional tiene un valor marginal distinto dependiendo del NSE de la zona.
- **Target Encoding**: Se calculó el precio promedio histórico por comuna (usando solo el set de entrenamiento) para capturar el .efecto barrio"de forma numérica.

7.d. Modelamiento y Resultados

Se evaluaron tres modelos principales, estrategias distintas, utilizando la transformación logarítmica sobre la variable objetivo ($\ln(Price)$) para estabilizar la varianza y mejorar el aprendizaje de errores relativos (MAPE).

Modelo	**Descripción**	**MAPE (Validación)**
XGBoost Base	Hiperparámetros estándar sin transformación de target.	17,38 %
XGBoost Log	Optimizado con $\ln(y)$, profundidad 8 y *learning rate* 0.02.	15,38 %
Ensemble Final	Promedio pesado (50/50) entre XGBoost y LightGBM.	15,41 %

Tabla 8: Comparativa de desempeño de los modelos entrenados.

El modelo final seleccionado fue el **Ensemble**, ya que la combinación de los distintos métodos de crecimiento de árboles (*level-wise* vs *leaf-wise*) permitió una generalización superior, logrando un **MAPE de 14.72 %** en el set de validación.