Informe Técnico: Reto Corona - Sistemas Inteligentes de Recomendación B2C y B2B

Equipo: UniChampions
Integrantes: Andes Chaparro – Juan Bernal
Universidad de los Andes - Facultad de Ingeniería

Tabla de Contenido

1. Introducción

En un entorno de mercado altamente competitivo como el de Corona, la personalización de la oferta y la eficiencia operativa son fundamentales para el éxito sostenido. La capacidad de anticipar las necesidades del cliente y sugerir productos relevantes no solo mejora la experiencia de compra, sino que también impulsa indicadores clave como la tasa de conversión, el valor promedio del pedido (AOV) y la lealtad del cliente. Las soluciones analíticas avanzadas, basadas en el aprovechamiento inteligente de los datos transaccionales y de comportamiento, permiten a Corona diferenciarse, optimizar la gestión de su portafolio y fortalecer su posición como socio estratégico tanto para consumidores finales (B2C) como para clientes empresariales (B2B).

Como equipo UniCampios desarrollamos una solución enfocada en la creación de sistemas de recomendación de productos personalizados para los segmentos B2C y B2B. Se detallan la problemática abordada, los objetivos perseguidos, la metodología analítica empleada, la arquitectura del prototipo funcional desarrollado y las propuestas estratégicas derivadas del análisis, con el fin de demostrar la generación de valor real para Corona y sus clientes.

2. Descripción del Problema y Objetivos

2.1. Problema de Negocio

Corona enfrenta el desafío de mejorar la personalización en la experiencia del cliente y optimizar su oferta de productos en un mercado dinámico. La falta de un sistema automatizado y escalable para identificar y recomendar productos relevantes limita el potencial de ventas cruzadas (cross-selling) y ascendentes (up-selling), afectando la fidelización y el crecimiento de ingresos. Operativamente, la toma de decisiones sobre gestión de inventario y estrategias de portafolio puede beneficiarse de un enfoque más basado en datos de consumo y afinidad entre productos. Esta necesidad se extiende tanto al canal B2C, donde se busca una experiencia de compra más fluida y satisfactoria para el consumidor final, como al canal B2B, donde la recomendación precisa puede optimizar las compras de los clientes empresariales y fortalecer la relación comercial. La ausencia de estas herramientas inteligentes dificulta la diferenciación, la eficiencia operativa y el posicionamiento estratégico de Corona.

2.2. Objetivo Principal

Desarrollar una solución analítica basada en machine learning que permita hacer recomendaciones personalizadas de productos para mejorar la experiencia del cliente (B2C y B2B), incrementar las ventas y optimizar la operación empresarial de Corona.

2.3. Objetivos Específicos

- Analizar los datos transaccionales (B2C y B2B) y de cotizaciones (B2C) para identificar patrones de consumo y afinidad entre productos.
- Un modelo personalizado user-to-item (B2C) para recomendaciones tipo "For You".

- Generar nuevas variables significativas (ingeniería de características) para enriquecer la descripción de los productos y su contexto de compra/cotización.
- Desarrollar e implementar modelos de recomendación híbridos, adaptados a las particularidades de los segmentos B2C y B2B.
- Construir un prototipo funcional (backend y frontend) que demuestre la operatividad de los sistemas de recomendación.
- Definir métricas y proponer estrategias para medir y maximizar el impacto de la solución en los indicadores clave de negocio (KPIs).

3. Metodología General y Fuentes de Datos

3.1. Enfoque Metodológico

Se adoptó un enfoque basado en ciencia de datos y machine learning, siguiendo las siguientes fases:

1. **Entendimiento del Negocio y Datos:** Análisis de los requerimientos del reto y exploración inicial de las bases de datos proporcionadas.
2. **Preparación de Datos:** Limpieza, transformación, ingeniería de características y unificación de datos relevantes para cada segmento (B2C y B2B).
3. **Modelado:** Desarrollo de modelos de recomendación específicos:
    - **B2C:** Un modelo híbrido avanzado combinando similitud de contenido (características de producto de transacciones y cotizaciones) y filtrado colaborativo item-item (co-compra y co-cotización), utilizando una técnica de re-ranking ponderado para la fusión.
    - Un modelo de Filtrado Colaborativo Ítem-Ítem (User-to-Item) para generar recomendaciones personalizadas "For You" basadas en el historial de compras del usuario y la similitud entre productos.
    - **B2B:** Un modelo híbrido pragmático enfocado en co-ocurrencia ponderada por valor, similitud categórica y alineación estratégica, adecuado para las características de las transacciones B2B.
4. **Implementación y Prototipado:** Desarrollo de scripts Python para el pre-cálculo y la lógica de recomendación, y una API (Flask para B2B) para servir las recomendaciones a una interfaz web (React).
5. **Análisis y Propuestas:** Evaluación cualitativa de las recomendaciones y formulación de propuestas estratégicas.

3.2. Fuentes de Datos Utilizadas

Se utilizaron las tres bases de datos proporcionadas por Corona:

1. **base_1_transaccional.txt (B2C Transacciones):** Información detallada de ventas a consumidores finales (personas naturales), incluyendo fecha, pedido, cliente, producto, categorías, cantidades, precios y asesor. *Utilizada para el modelo B2C*.
2. **base_2_cotizaciones.txt (B2C Cotizaciones):** Información sobre cotizaciones realizadas a consumidores finales, incluyendo estado, fechas, cliente, producto y detalles de la cotización. *Utilizada para el modelo B2C*.
3. **base_3_transaccional_b2b.txt (B2B Transacciones):** Información de ventas facturadas a clientes empresariales (B2B), con identificadores, categorías y valores específicos del segmento B2B. *Utilizada para el modelo B2B*.

4. Sistema de Recomendación B2C

Dada la riqueza de los datos B2C (transacciones y cotizaciones) y la necesidad de una alta personalización, se desarrolló un sistema híbrido sofisticado.

4.1. Procesamiento de Datos e Ingeniería de Características B2C

El objetivo de esta fase, ejecutada principalmente en el notebook ingenieria_de_Caracteres_B2C.ipynb, fue transformar los datos brutos de transacciones (base_1_transaccional.txt) y cotizaciones (base_2_cotizaciones.txt) en un conjunto de características enriquecido y estructurado a nivel de producto, ideal para alimentar los modelos de recomendación.

4.1.1. Limpieza y Preparación Inicial:

- Se cargaron los datos B2C de transacciones y cotizaciones.
- Se realizó una limpieza inicial eliminando registros duplicados y filas con valores nulos en columnas críticas (como fechas o identificadores), asegurando la integridad de los datos base.
- Las columnas de fecha (fecha, fecha_creacion, fecha_modificacion) se convirtieron explícitamente al formato datetime para permitir cálculos basados en tiempo.

4.1.2. Creación de Características Basadas en Tiempo:

- **Para Transacciones:** Se extrajeron componentes temporales como año_venta, mes_venta, dia_semana_venta, dia_mes_venta, y semana_año_venta. Estas variables permiten capturar posibles patrones estacionales o tendencias temporales en las compras.
- **Para Cotizaciones:** De manera similar, se extrajeron año_cot, mes_cot, dia_semana_cot, hora_cot. Adicionalmente, se calculó dias_modificacion_cot (diferencia entre creación y modificación) como un indicador potencial de la complejidad de la cotización o la indecisión del cliente.

4.1.3. Creación de Características Agregadas por Producto (Perfil del Producto):

Se calcularon métricas clave agrupadas por producto para construir un perfil detallado de cada ítem, utilizando ambas fuentes de datos:

- **Desde Transacciones (product_features_trans):**
    - total_unidades_vendidas, valor_total_ventas: Cuantifican la popularidad y el impacto económico general del producto.
    - precio_promedio_venta: Establece el punto de precio típico del producto en ventas reales.
    - n_transacciones_producto, n_clientes_producto: Miden el alcance y la frecuencia con la que el producto participa en transacciones y es adquirido por distintos clientes.
    - frecuencia_venta_prod: Calculada como n_transacciones_producto dividido por los días activo (desde la primera hasta la última venta), indica la regularidad de venta del producto.
    - alineación con portafolio estratégico: Valor promedio de la alineación estratégica reportada para el producto.
    - color: Se tomó la moda (el color más frecuente) como atributo descriptivo principal.
- **Desde Cotizaciones (product_features_cot):**
    - total_unidades_cotizadas, valor_total_cotizado: Reflejan el volumen de interés generado por el producto en la fase de cotización.
    - precio_promedio_cot, valor_promedio_cot: Indican el precio y valor promedio con el que se cotiza el producto.
    - n_cotizaciones_producto, n_clientes_cotizaron: Miden cuántas veces y por cuántos clientes diferentes se ha cotizado el producto.
    - tasa_conversion_cot_prod: Calculada como el número de cotizaciones ganadas dividido por el total de cotizaciones para ese producto, estima la probabilidad de que el interés se convierta en venta.
    - producto_fue_comprado_por_cliente: Un flag (0 o 1, agregado con max) que indica si al menos una de las cotizaciones de este producto para *cualquier* cliente resultó posteriormente en una compra de ese mismo producto por ese cliente (basado en la unión con el historial de transacciones).

4.1.4. Creación de Características de Popularidad Relativa (Contexto del Producto):

Para entender mejor la importancia de un producto dentro de su contexto, se calcularon métricas de popularidad relativa basadas en los agregados por categoría y los agregados globales:

- popularidad_valor_prod_en_cat, popularidad_unidad_prod_en_cat: Proporción del valor/unidades totales de la categoría que representa este producto específico.
- popularidad_valor_prod_global, popularidad_unidad_prod_global: Proporción del valor/unidades totales de *todas* las transacciones que representa este producto.

4.1.5. Unificación de Características e Imputación (product_features_unified):

- **Unificación:** Se combinaron las características agregadas por producto de transacciones y cotizaciones en un único DataFrame (product_features_unified) utilizando un outer join sobre el índice producto. Esta estrategia asegura que *todos* los productos presentes en cualquiera de las dos fuentes (ventas o cotizaciones) sean incluidos en el perfil final. El DataFrame resultante contiene n_products filas (7277 productos únicos en total).
- **Imputación:** El outer join genera valores NaN para los productos que existen solo en una fuente (ej., un producto solo vendido tendrá NaN en las métricas de cotización). Estos valores faltantes se imputaron cuidadosamente:
    - Variables numéricas: Se rellenaron con la mediana de la respectiva columna.
    - Variables categóricas (categoria_macro, categoria, subcategoria, color): Se rellenaron con la moda (valor más frecuente) o 'Desconocido' si la moda no estaba definida.
- **Resultado:** El DataFrame product_features_unified final contiene un perfil completo y sin valores faltantes para cada uno de los 7277 productos únicos B2C teniendo en cuenta los datos de las transacciones como el de las cotizaciones y nuevas variables creadas que aportan mayor información al modelo, listo para ser utilizado en el modelo basado en contenido.

4.1.6. Almacenamiento:

Los DataFrames resultantes (transacciones_con_features.csv y cotizaciones_con_features.csv) con todas las características creadas fueron guardados para su uso posterior en el desarrollo de los modelos de recomendación.

4.2. Componentes del Modelo de Recomendación B2C

Se pre-calcularon las siguientes matrices y estructuras de datos:

4.2.1. Modelo Basado en Contenido

Se aplicó MinMaxScaler a las features numéricas unificadas y OneHotEncoder a las categóricas, generando feature_matrix_sparse. Sobre esta, se calculó la content_similarity_matrix usando similitud coseno, capturando la similitud intrínseca entre productos.

4.2.2. Modelo Colaborativo (Co-Compra)

Se construyó la matriz dispersa co_occurrence_matrix contando cuántas veces cada par de productos apareció en el mismo pedido en los datos transaccionales.

4.2.3. Modelo Colaborativo (Co-Cotización)

Se construyó la matriz dispersa co_quotation_matrix contando cuántas veces cada par de productos apareció en la misma cotizacion.

4.3. Estrategia de Hibridación B2C: Re-Ranking Ponderado

Se implementó la función get_recommendations_hybrid_rerank que utiliza la técnica de re-ranking:

1. Obtiene los Top-M candidatos de cada uno de los tres métodos anteriores (get_top_n_candidates_per_method).
2. Calcula un score híbrido para cada candidato sumando las contribuciones ponderadas de su rango inverso en cada lista (weight * (1 / (rank + k))).
3. Incluye una lógica adaptativa: si un producto tiene muy baja evidencia de co-compra (score < 10 en el Top-1 de co-compra), se ajustan los pesos para dar más importancia al contenido (content_weight=0.5, cf_buy_weight=0.3, cf_quote_weight=0.2), priorizando las características del producto sobre patrones de compra débiles. De lo contrario, se utilizan los pesos pre establecidos (content_weight=0.3, cf_buy_weight=0.2, cf_quote_weight=0.2),
4. Ordena los candidatos por este score híbrido y devuelve el Top-N que de forma predetermina devuelve los primeros diez.

4.4. Implementación y Lógica B2C

La lógica completa de preprocesamiento, cálculo de matrices y la función de recomendación híbrida se encuentra detallada en el script algoritmo_de_recomendacion_B2C.py (derivado del notebook). Este script realiza los pre-cálculos necesarios y expone la función get_recommendations_hybrid_rerank para ser utilizada (potencialmente a través de una API similar a la de B2B).

4.5. Modelo de Recomendación Personalizada B2C ('For You' - Ítem-Based CF)

Además del sistema híbrido item-to-item descrito anteriormente, se desarrolló un segundo motor de recomendación B2C enfocado específicamente en la personalización a nivel de usuario, generando sugerencias tipo "For You". Este modelo se basa en la técnica de Filtrado Colaborativo Ítem-Ítem (Item-Based Collaborative Filtering), una metodología clásica y efectiva para la personalización.

- **Objetivo:** Dado un cliente_id específico, recomendar productos que probablemente le interesen basándose en su historial de compras pasado y el comportamiento de compra de usuarios similares (implícitamente a través de la similitud de ítems).

- **Fuente de Datos:** Este modelo utiliza únicamente las interacciones cliente-producto (cliente_id, producto) extraídas del archivo transacciones_con_features.csv. No requiere las características de contenido ni los datos de cotizaciones.
- **Metodología:**
    1. **Preparación de Datos:** Se cargan las interacciones únicas cliente-producto. Se crean mapeos numéricos para usuarios (user_to_idx) y productos (item_to_idx).
    2. **Matriz Usuario-Ítem:** Se construye una matriz dispersa (user_item_matrix_csr) donde las filas representan usuarios y las columnas representan productos. Una celda (u, i) tiene valor 1 si el usuario u compró el producto i, y 0 en caso contrario.
    3. **Cálculo de Similitud Ítem-Ítem:** Se calcula la similitud coseno (item_similarity_matrix) entre todos los pares de ítems, utilizando la transpuesta de la matriz usuario-ítem. La similitud entre el ítem A y el ítem B se basa en cuántos usuarios compraron *ambos*. La diagonal de esta matriz se pone a cero.
    4. **Generación de Recomendaciones (get_item_based_recommendations_allow_repurchase):**
        - Se identifica el historial de ítems comprados por el cliente_id de entrada.
        - Para cada ítem comprado, se buscan los k ítems más similares usando la item_similarity_matrix.
        - Se acumulan los puntajes de similitud para todos los ítems candidatos (los ítems similares encontrados). Un mismo ítem puede recibir puntaje de similitud de varios ítems comprados por el usuario.
        - Los ítems candidatos se ordenan por su puntaje acumulado total.
        - Se devuelven los Top-N ítems con mayor puntaje.
    5. **Permitir Re-compra:** La implementación actual permite que ítems ya comprados por el usuario puedan ser recomendados nuevamente. Esto puede ser útil para productos consumibles o de compra recurrente, aunque podría requerir ajustes (filtrado post-recomendación) si se desean únicamente recomendaciones de descubrimiento.
- **Implementación:** Esta lógica se encapsuló en un servidor Flask dedicado (ejecutado en el puerto 5002, script no proporcionado en el contexto inicial pero inferido del código del servidor B2C ítem-based) que realiza los pre-cálculos (user_item_matrix_csr, item_similarity_matrix) al inicio y expone un endpoint API /recommend/user/<cliente_id> para servir las recomendaciones personalizadas. Sin embargo, como no se pudo añadir correctamente a la aplicación react del

proyecto se hizo un jupyter para poder correrlo desde ahí y analizarlo que se llama algoritmo_de_recomendacion_B2C.ipynb

- **Diferencia Clave vs. Modelo Híbrido Item-to-Item:** Mientras el modelo híbrido (Secciones 4.1-4.7) responde a la pregunta "¿Qué productos están relacionados con *este producto*?", este modelo Ítem-Based responde a "¿Qué productos le podrían gustar a *este cliente* basándose en lo que ha comprado antes?". Es intrínsecamente personalizado.

5. Sistema de Recomendación B2B

Para el segmento B2B, se implementó un enfoque híbrido diferente, adaptado a la naturaleza de los datos y las posibles necesidades de los clientes empresariales, utilizando el script server/app.py.

5.1. Procesamiento de Datos y Pre-cálculos B2B

La función preprocess_data en app.py realiza los siguientes cálculos al iniciar el servidor:

- **Limpieza:** Conversión de tipos y limpieza de strings.
- **Mapa Categoría-Subcategoría:** category_subcategory_map para identificar rápidamente productos en la misma categoría/subcategoría.
- **Alineación Estratégica Promedio:** product_strategic_alignment calcula la alineación promedio por producto.
- **Valor Promedio:** product_average_value calcula el valor total promedio por producto.
- **Co-ocurrencia Ponderada por Valor:** Se calcula weighted_co_occurrence_matrix. A diferencia del B2C, aquí el "peso" de la co-ocurrencia entre dos productos se incrementa por el valor_total de la transacción donde aparecen juntos. Esto da más importancia a las co-ocurrencias en transacciones de mayor valor, relevante en B2B.

5.2. Lógica de Recomendación B2B

La función get_recommendations implementa la lógica:

1. **Candidatos Iniciales:** Se obtienen principalmente de la matriz de co-ocurrencia ponderada (weighted_co_occurrence_matrix), ordenados por el peso acumulado.
2. **Relleno (Fallback):** Si no hay suficientes candidatos por co-ocurrencia, se añaden productos de la misma categoría y subcategoría que el producto de entrada (usando category_subcategory_map).
3. **Ponderación y Ranking Final:** Para cada candidato, se calcula un puntaje multidimensional considerando:
    - Similitud de Categoría/Subcategoría (score 2 si coinciden ambas, 1 si coincide categoría).

    - Alineación Estratégica (pre-calculada).
    - Valor Promedio (pre-calculado).
    - Peso de Co-ocurrencia (pre-calculado).
4. **Ordenamiento:** Los candidatos se ordenan priorizando la similitud categórica, luego la alineación, el valor y finalmente el peso de co-ocurrencia.
5. **Salida:** Se devuelven los N productos mejor rankeados.

5.3. Implementación y API B2B

El script app.py utiliza Flask para exponer un endpoint /recommend (método POST) que recibe un product_name y devuelve una lista JSON con los detalles de los productos recomendados, incluyendo información relevante para el contexto B2B (municipio, zona, categorías B2B, valor, alineación). Los datos pre-calculados se cargan en memoria al iniciar el servidor para respuestas rápidas.

El algoritmo de recomendación toma un producto y genera hasta cinco sugerencias de productos B2B relacionados usando datos preprocesados. Verifica si el producto existe; si no, retorna una lista vacía. Extrae productos co-ocurrentes (presentes en las mismas transacciones) de una matriz ponderada por el valor total de transacciones, ordenándolos por peso. Si faltan recomendaciones, incluye productos de la misma categoría y subcategoría. Cada candidato se evalúa con un puntaje que combina: 1) similitud de categoría/subcategoría (2 si coinciden ambas, 1 si solo categoría, 0 si ninguna), 2) alineación estratégica promedio, 3) valor promedio del producto, y 4) peso de co-ocurrencia. Los productos se ordenan priorizando la similitud de categoría, luego los otros criterios, y se retornan los más relevantes, ofreciendo recomendaciones estratégicas y contextuales.

6. Arquitectura de la Solución y Prototipo

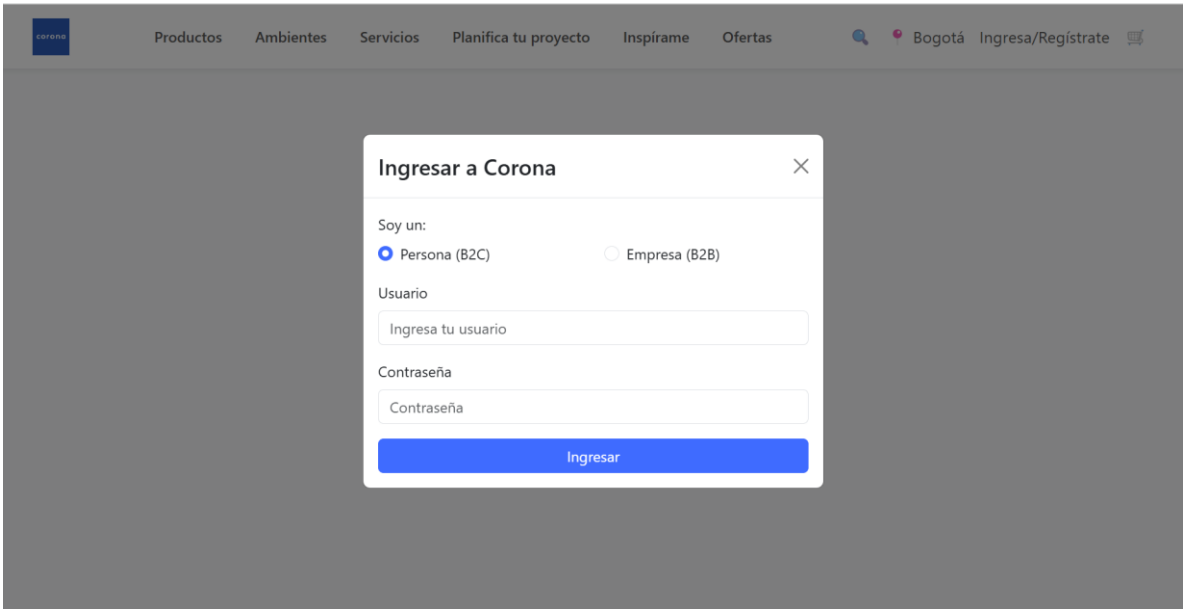
Se desarrolló un prototipo funcional que integra los componentes de backend y frontend.

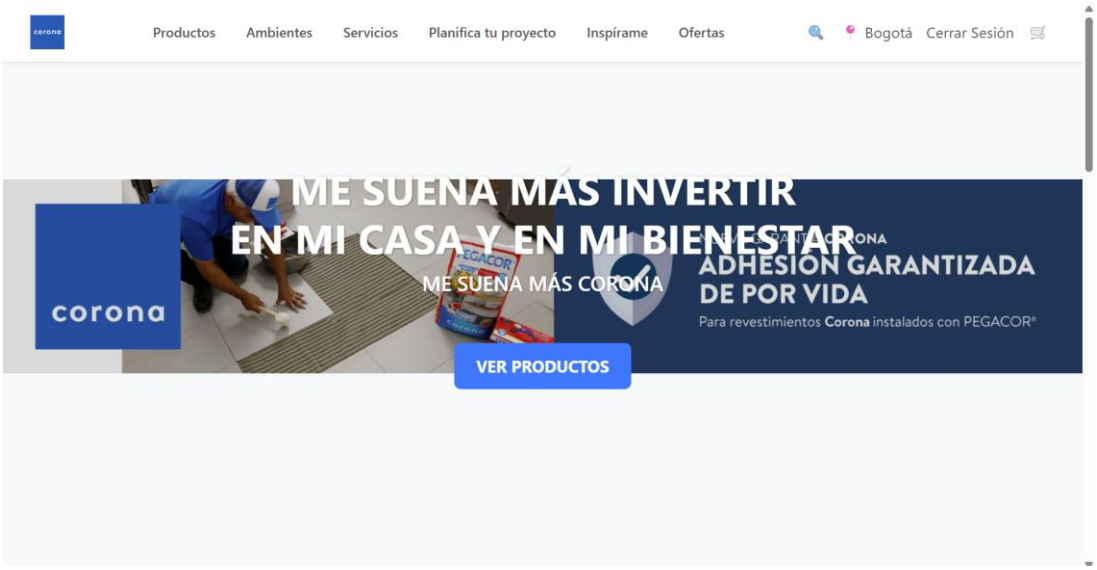
(Sugerencia: Incluir un diagrama de arquitectura aquí)

6.1. Arquitectura General

La solución sigue una arquitectura cliente-servidor:

- **Frontend:** Una aplicación web desarrollada con React y Bootstrap (create-react-app).





- **Backend:**
    - Un servidor API Flask (app.py) para las recomendaciones B2B, que carga y pre-procesa los datos B2B al inicio.
    - Un script Python (algoritmo_de_recomendacion_B2C.py) que contiene la lógica B2C. Para una implementación productiva, este script realizaría los pre-cálculos offline y la función de recomendación se expondría a través de otra API Flask (o se integrarían ambas lógicas en un único backend más robusto).
- **Datos:** Archivos CSV con datos brutos y/o pre-calculados (features, matrices).

6.2. Componente Backend (API y Lógica de Recomendación)

- **B2B:** El servidor Flask (app.py) maneja las solicitudes /recommend, llama a la función get_recommendations y devuelve los detalles de los productos recomendados en formato JSON. Realiza pre-cálculos en memoria.
- API para recomendaciones B2C "For You" (basada en la lógica del servidor Ítem-Based en puerto 5002).
- **B2C:** La lógica (algoritmo_de_recomendacion_B2C.py) realiza pre-cálculos más intensivos (matrices de similitud y co-ocurrencia). La función get_recommendations_hybrid_rerank genera las recomendaciones.

6.3. Componente Frontend (Interfaz de Usuario)

- Desarrollado con React (package.json indica dependencias como react, react-bootstrap, react-slick).
- Permite al usuario (presumiblemente un asesor de ventas o un cliente navegando) buscar o seleccionar un producto.
- Realiza una llamada a la API backend correspondiente (B2B o B2C).
- Muestra los productos recomendados, potencialmente en un carrusel (react-slick) u otro formato visualmente atractivo, junto con detalles relevantes.
- El README.md indica cómo construir (npm build) y ejecutar (npm start) la aplicación frontend.

6.4. Flujo de Recomendación (Usuario a Sistema)

1. Usuario interactúa con el frontend (busca/selecciona un producto).
2. Frontend envía el ID del producto a la API backend adecuada.
3. Backend (Flask API para B2B, o la lógica B2C) recibe la solicitud.
4. Se llama a la función de recomendación correspondiente (get_recommendations o get_recommendations_hybrid_rerank).
5. La función utiliza los datos y matrices pre-calculadas para generar la lista de recomendaciones Top-N.
6. Backend devuelve la lista de productos recomendados (con detalles) al frontend en formato JSON.
7. Frontend muestra las recomendaciones al usuario.

7. Resultados y Hallazgos (Ejemplos)

7.1. Ejemplos de Recomendaciones B2C

Al probar el modelo híbrido B2C con example_product_id = "producto_2", se obtuvieron las siguientes recomendaciones Top-10 (con pesos 0.3 contenido, 0.5 co-compra, 0.2 co-cotización, k=1):

Recomendaciones de Contenido		*Recomendaciones por Co-Compra*		*Recomendaciones por Co-Cotización*	
Producto	**Similarity Score**	**Producto**	**Co-Purchase Count**	**Producto**	**Co-Quotation Count**
producto_1232	0.999869	producto_413	17	producto_3126	13
producto_1120	0.999325	producto_3126	17	producto_632	6
producto_1110	0.999110	producto_1120	16	producto_208	6
producto_3115	0.998920	producto_119	16	producto_642	6
producto_837	0.998911	producto_632	15	producto_865	6

Recomendación modelo hibrido	
Producto	**Hybrid Score**
producto_3126	0.350000
producto_1120	0.234524
producto_1232	0.213462
producto_413	0.166667
producto_632	0.118095

Comparando con los resultados individuales:

- **Contenido:** producto_1232, producto_1120 aparecen alto.
- **Co-Compra:** producto_413, producto_3126, producto_1120, producto_119, producto_632 aparecen alto.
- **Co-Cotización:** producto_3126, producto_632, producto_208 aparecen.

El modelo híbrido logra combinar estas señales, dando prominencia a producto_3126 (fuerte en co-compra y co-cotización), producto_1120 y producto_1232 (fuertes en contenido y co-compra), y producto_413 (muy fuerte en co-compra).

7.2. Ejemplos de Recomendaciones B2C "For you"

Producto	**Recommendation Score**
producto_49	0.786332

Producto	Recommendation Score
producto_119	0.653563
producto_19	0.619442
producto_3	0.482461
producto_28	0.478430
producto_248	0.350241
producto_176	0.344945
producto_27	0.334039
producto_40	0.316651
producto_66	0.285180

Para un cliente específico, como el id 6, el sistema genera recomendaciones personalizadas tipo "For You". La lista resultante, encabezada por productos como producto_49 (score ≈ 0.79), producto_119 (score ≈ 0.65) y producto_19 (score ≈ 0.62), se obtiene mediante un algoritmo de Filtrado Colaborativo Ítem-Ítem. Este método funciona identificando primero los productos que el cliente 6 ha comprado históricamente. Luego, utiliza una matriz de similitud entre ítems (pre-calculada a partir de patrones de co-compra de *todos* los clientes) para encontrar otros productos que son muy similares a los que el cliente 6 ya compró. El "Recommendation Score" final representa la fuerza agregada de estas similitudes, priorizando ítems que guardan una fuerte relación con el historial de compras conocido del cliente 6.

7.3. Ejemplos de Recomendaciones B2B

El modelo B@B prioriza productos co-comprados en transacciones de alto valor y aquellos con alta similitud categórica y alineación estratégica. Se espera que las recomendaciones para un producto B2B incluyan:

- Productos frecuentemente facturados junto con el producto de entrada, ponderados por el valor de esas facturas.
- Otros productos dentro de la misma categoria_b2b y subcategoria_b2b.
- Productos con alta puntuación en alineación con portafolio estratégico b2b.

7.3. Observaciones Preliminares

- La señal de **co-compra (B2C)** parece ser densa y fuerte, indicando patrones de compra conjunta bien definidos.
- La señal de **co-cotización (B2C)** es significativamente más dispersa, sugiriendo que los productos cotizados juntos varían más o que hay menos productos por cotización en promedio. No obstante, puede aportar información de interés temprano.
- La **similitud de contenido (B2C)**, enriquecida con features de ambas fuentes, proporciona una base sólida, especialmente para productos con menos historial de interacción (problema de *cold start*).
- El modelo **B2B** se enfoca correctamente en señales relevantes para ese contexto (valor de transacción, categorías B2B, alineación estratégica).

8. Propuestas Estratégicas e Impacto en el Negocio

La implementación de estos sistemas de recomendación tiene el potencial de generar valor significativo para Corona en diversas áreas:

8.1. Creación de Valor para el Consumidor y Cliente B2B

- **Mejora de la Experiencia B2C:** Sugerir productos relevantes y complementarios facilita el proceso de descubrimiento y compra, aumentando la satisfacción y potencialmente reduciendo el tiempo de búsqueda.
- **Optimización de Compra B2B:** Recomendar productos basados en co-compra ponderada por valor y similitud categórica puede ayudar a los clientes B2B a construir pedidos más completos y eficientes, alineados con sus propias necesidades o las de sus clientes finales.

- **Personalización:** Aunque los modelos actuales son item-to-item, sientan las bases para futuras personalizaciones basadas en el historial del cliente.
- El modelo híbrido item-to-item facilita el descubrimiento y la venta cruzada contextual. El modelo "For You" crea una experiencia personalizada que puede aumentar la lealtad y el engagement a largo plazo.

8.2. Apoyo a Asesores de Venta

- El prototipo frontend demuestra cómo la herramienta puede ser utilizada por asesores en puntos de venta físicos.
- El asesor puede usar el item-to-item para responder a un interés específico del cliente, y el "For You" (si el cliente está identificado) para dar sugerencias más generales basadas en el historial del cliente.
- Al buscar un producto de interés para el cliente, el asesor recibe instantáneamente sugerencias de productos relacionados (similares, complementarios, frecuentemente comprados juntos).
- Esto empodera al asesor con información basada en datos para realizar ventas cruzadas/ascendentes efectivas, ofrecer alternativas y dar un servicio más informado y personalizado.

8.3. Optimización Operativa y de Portafolio

- **Gestión de Inventario:** Analizar las co-ocurrencias fuertes puede informar decisiones sobre qué productos mantener en stock juntos o en ubicaciones cercanas.
- **Estrategia de Portafolio:** Identificar productos con alta similitud de contenido pero bajas ventas cruzadas podría señalar oportunidades de marketing o bundling. El análisis de la alineación estratégica en B2B puede guiar el desarrollo o promoción de productos clave.
- **Campañas de Marketing:** Las recomendaciones pueden alimentar campañas de email marketing o publicidad dirigida, sugiriendo productos basados en compras anteriores.

8.4. Medición del Impacto (KPIs Sugeridos)

Se sugiere monitorear los siguientes KPIs para evaluar el éxito de la implementación:

- **Tasa de Conversión (Click-Through Rate en Recomendaciones):** % de veces que un usuario hace clic en un producto recomendado.
- **Tasa de Conversión (Add-to-Cart/Purchase Rate from Recommendations):** % de veces que un producto recomendado es añadido al carrito o comprado.
- **Incremento en Valor Promedio de Pedido (AOV):** Comparar el AOV de pedidos que incluyen productos recomendados vs. aquellos que no.

- **Ingresos por Recomendaciones:** Valor total de los productos vendidos que fueron descubiertos a través de una recomendación.
- **Cobertura del Catálogo:** % del catálogo que aparece en alguna recomendación.
- **Diversidad de Recomendaciones:** Medir qué tan variadas son las recomendaciones para evitar la sobreexposición de los mismos best-sellers.
- **Satisfacción del Asesor/Cliente (Encuestas):** Medir cualitativamente la utilidad percibida.

9. Escalabilidad, Mantenimiento y Futuras Mejoras

9.1. Consideraciones de Escalabilidad

- **Infraestructura:** Para producción, se recomienda desplegar el backend en una plataforma cloud robusta (Azure, AWS, GCP) utilizando servicios como Azure Functions/App Service, AWS Lambda/EC2/ECS, o Google Cloud Run/App Engine. El uso de contenedores (Docker) facilitaría el despliegue.
- **Pre-cálculo:** Los cálculos intensivos (matrices de similitud/co-ocurrencia) deben ejecutarse como procesos batch offline (ej. usando Databricks, Azure ML Pipelines, Airflow). La frecuencia dependerá de la volatilidad de los datos (¿diaria, semanal?).
- **Servicio de Recomendaciones:** La API que sirve las recomendaciones debe ser escalable horizontalmente para manejar picos de demanda. Los datos pre-calculados (matrices, mapeos) deben cargarse eficientemente en memoria o en una base de datos/caché rápida (como Redis).
- **Frontend:** La aplicación React puede servirse estáticamente a través de una CDN para un rendimiento óptimo.

9.2. Mejoras para la siguiente entrega

Una limitación clave del prototipo actual, y por tanto una prioridad para futuras iteraciones, es la integración completa de todos los motores de recomendación desarrollados en la interfaz de usuario frontend. Si bien el sistema de recomendación B2B está funcionalmente conectado a la aplicación React, demostrando la viabilidad de la arquitectura, la integración de los dos algoritmos B2C (el híbrido item-to-item y el personalizado 'For You' basado en usuario) no pudo completarse debido principalmente a limitaciones de tiempo y a los desafíos inherentes al desarrollo frontend con React dentro del cronograma del reto.

No obstante, es fundamental destacar que ambos algoritmos B2C son funcionalmente robustos y su lógica ha sido validada, estando disponibles para demostración y prueba a través de los notebooks Jupyter (ingenieria_de_Caracteres_B2C.ipynb, algoritmo_de_recomendacion_B2C.ipynb) y los scripts Python asociados (algoritmo_de_recomendacion_B2C.py y el servidor Ítem-Based B2C).

Para una siguiente entrega o fase del proyecto, el objetivo primordial sería completar esta integración frontend, permitiendo visualizar y interactuar con las recomendaciones B2C directamente en la aplicación. Adicionalmente, como una mejora significativa para enriquecer la interacción del usuario y ofrecer un canal alternativo de recomendación, proponemos explorar la incorporación de un chatbot inteligente. Este podría asistir a los usuarios en la navegación, responder preguntas frecuentes sobre productos y, potencialmente, entregar recomendaciones personalizadas de forma conversacional, creando una experiencia de cliente aún más dinámica y asistida.

10. Ejecución del Prototipo

Primero, asegúrese de que los archivos de datos originales (base_1_transaccional.txt, base_2_cotizaciones.txt, base_3_transaccional_b2b.txt) se encuentren en la carpeta server/Datos/. A continuación, es **indispensable** ejecutar el notebook Jupyter ingenieria_de_Caracteres_B2C.ipynb (asegurándose de estar en el directorio correcto para que encuentre los datos) para procesar los datos B2C y generar los archivos transacciones_con_features.csv y cotizaciones_con_features.csv en la misma carpeta server/Datos/.

Una vez preprocesados los datos B2C, abra tres terminales separadas y en cada una navegue al directorio server/ usando cd server. En la primera terminal, inicie el backend B2B ejecutando python app.py. En la segunda, inicie el backend B2C híbrido item-to-item con python algoritmo_de_recomendacion_B2C.py. En la tercera, inicie el backend B2C personalizado user-to-item ejecutando su script correspondiente (por ejemplo, python algoritmo_b2c_user_based.py).

Para el frontend, abra una terminal en el directorio raíz del proyecto. Si es la primera vez, instale las dependencias con npm install. Luego, inicie la aplicación React ejecutando npm start, lo que debería abrir la interfaz en su navegador.

Nota Importante: Actualmente, solo el backend B2B (app.py) está conectado funcionalmente al frontend. Los backends B2C, aunque operativos, requerirían pruebas directas a través de herramientas como Postman o la ejecución de sus lógicas en los notebooks correspondientes para verificar sus recomendaciones. Asegúrese de tener Python, Node.js/npm y todas las bibliotecas Python requeridas instaladas.

11. Conclusiones

Este proyecto abordó el reto propuesto por Corona mediante el desarrollo de dos sistemas de recomendación de productos híbridos y diferenciados para los segmentos B2C y B2B. Se realizó un análisis profundo de los datos, una significativa ingeniería de características y se implementaron modelos que combinan similitud de contenido y filtrado colaborativo. La solución B2C utiliza un enfoque de re-ranking robusto para fusionar señales de ventas y cotizaciones, mientras que la solución B2B prioriza la co-ocurrencia ponderada por valor y la alineación estratégica.

El prototipo funcional, con backend (Flask para B2B) y frontend (React), demuestra la viabilidad técnica y la aplicabilidad práctica de la solución, especialmente como herramienta de apoyo para los asesores de venta. La implementación de estos sistemas tiene el potencial de mejorar la experiencia del cliente, incrementar las ventas a través de recomendaciones relevantes y optimizar la gestión operativa. Las propuestas estratégicas y las consideraciones de escalabilidad sientan las bases para una futura implementación productiva que genere un impacto medible en los KPIs de negocio de Corona. Sin embargo, la falta de tiempo y conocimiento de react nos limito bastante, pero estas limitaciones se buscaran pasar para la siguiente entrega unificando todos los modelos y creando nuevas mejoras y propuestas nuevas.

12. Referencias Bibliográficas

1. Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer. https://link.springer.com/book/10.1007/978-3-319-29659-3
2. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749. https://pages.stern.nyu.edu/~atuzhili/pdf/TKDE-Paper-as-Printed.pdf
3. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370. https://link.springer.com/article/10.1023/A:1021240730564
4. Cormack, G. V., Clarke, C. L., & Buettcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, 758-759. https://dl.acm.org/doi/10.1145/1571941.1572114
5. Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 5-53. https://dl.acm.org/doi/10.1145/963770.963772