Cook My Meal! Plataforma gastronomica inteligente con recomendaciones personalizadas usando SVD y Análisis de Sentimiento

Diego Exequiel Tabares

***Resumen*—El presente trabajo se enmarca en la línea de investigación sobre sistemas de recomendación inteligentes aplicados al ámbito gastronómico, continuando el proyecto teórico previo *"Plataforma inteligente de platos personalizados con recomendación basada en ratings y reviews colaborativos"*. Dicho estudio abordó los fundamentos del filtrado colaborativo mediante *Singular Value Decomposition* (SVD) [1] y el análisis de sentimiento con *VADER* [2], proponiendo un modelo híbrido capaz de incorporar tanto las preferencias individuales de los usuarios como las percepciones colectivas expresadas en sus reseñas.**

Esta investigación presenta una contribución concreta al demostrar que la integración de ambas técnicas puede mejorar de manera significativa la personalización de la experiencia gastronómica digital. El modelo propuesto ofrece un enfoque innovador para transformar los tradicionales catálogos estáticos de comida en sistemas dinámicos de recomendación que aprenden y evolucionan con la comunidad. Su relevancia radica en mostrar cómo la unión entre el aprendizaje automático y el análisis semántico puede generar recomendaciones más precisas, emocionalmente contextualizadas y alineadas con los gustos reales de los usuarios, estableciendo las bases para futuras aplicaciones en plataformas de delivery inteligentes.

I. Introducción

El panorama moderno de delivery digital está saturado de aplicaciones que presentan menús estáticos y uniformes para todos los usuarios [3] [4]. Este proyecto parte de la hipótesis de que la experiencia del usuario puede transformarse aplicando *machine learning* para crear un recorrido culinario dinámico, inteligente y personalizado.

La plataforma *Cook My Meal!* se desarrolló para desafiar el paradigma de menú estático, proponiendo un ecosistema vivo donde los platos disponibles y su ranking se ven directamente influenciados por la inteligencia colectiva y el *feedback* explícito de los usuarios [5] [6] [7] [8] [9].

El objetivo central fue desarrollar una PoC centrada en el backend que validara un motor de recomendación híbrido. Este motor combina dos técnicas poderosas:

- **Filtrado colaborativo:** utiliza SVD para analizar patrones de comportamiento de los usuarios (ratings y compras) y predecir afinidad hacia platos que aún no han probado [1].
- **Análisis de sentimiento:** aplica VADER para cuantificar el tono emocional de las reseñas textuales, aportando información cualitativa que los ratings numéricos por sí solos no capturan [2].

La PoC incluye las siguientes funcionalidades principales:

- **Crear Plato:** permitir que los usuarios registren nuevos platos dentro del sistema.
- **Calificar Plato:** asignar puntuaciones y comentarios a los platos consumidos.
- **Obtener Menú:** visualizar el catálogo de platos disponibles.
- **Ver Recomendaciones:** recibir un menú personalizado en base a predicciones generadas por el motor de recomendación.

Para aislar el núcleo del problema, el proyecto excluyó deliberadamente funcionalidades complejas como autenticación, detalles ampliados de cada plato, pagos y logística de pedidos. Se utilizó un dataset inicial para poblar la base de datos SQLite, mitigando el problema de *cold start* y permitiendo pruebas inmediatas de la lógica de recomendación. Este documento describe la evolución desde el marco teórico inicial hasta un backend funcional, robusto y bien documentado, como así también una aplicación mobile donde se puede apreciar la interacción con el sistema y tener una visión integral del mismo.

II. Caso de Estudio

El caso de estudio simuló un escenario de uso real en un entorno controlado del backend, siendo consumido a través de una aplicación mobile en Android. El sistema se pobló inicialmente con un dataset de usuarios, platos de varias categorías (p. ej., *Vegetariano*, *Carne de Res*, *Pollo*) y un historial de compras y valoraciones. Se definieron cuatro funcionalidades clave: crear un plato, calificar un plato, obtener el menú y recibir recomendaciones personalizadas. El caso de uso principal contempla que un usuario ingrese a la aplicación, visualice su menú recomendado, cree un nuevo plato y posteriormente otros usuarios lo califiquen. De este modo, puede observarse cómo las calificaciones y reseñas impactan en el ranking dinámico de platos, modificando la oferta disponible en tiempo real.

Caso de uso principal:

1. El usuario solicita recomendaciones personalizadas vía `GET /reco/user/{user_id}`.
2. El sistema calcula un puntaje híbrido para cada plato y devuelve una lista ordenada de recomendaciones (con score $\geq 3{,}5$).
3. El usuario puede "comprar" o calificar un plato nuevo, proporcionando estrella y comentario textual.

4. La interacción se guarda en la base de datos.
5. Para demostración, un administrador puede disparar el reentrenamiento del modelo vía `POST /admin/retrain-model`.

La siguiente solicitud al sistema reflejará un conjunto actualizado de recomendaciones, mostrando la naturaleza dinámica y adaptable del sistema. Este flujo valida que las interacciones comunitarias —tanto implícitas (compras) como explícitas (valoraciones y reseñas)— impactan de manera directa y significativa en la evolución del menú personalizado.

III. Marco Teorico

El desarrollo del sistema propuesto se apoya en tres pilares conceptuales fundamentales: los sistemas de recomendación, el análisis de sentimiento aplicado a reviews, y la lógica de curado comunitario dinámico.

III-A. Sistemas de Recomendación

Los sistemas de recomendación pueden clasificarse en tres grandes enfoques: el *filtrado colaborativo*, que identifica patrones de comportamiento similares entre usuarios; la *recomendación basada en contenido*, que analiza las características de los ítems ofrecidos (como ingredientes o tipos de cocina); y los *enfoques híbridos*, que combinan ambas técnicas para lograr mayor precisión y cobertura [10] [11]. En este trabajo se adopta un enfoque híbrido porque permite no sólo basarse en las valoraciones cuantitativas de los usuarios, sino también integrar información cualitativa a través del análisis de sentimientos de los comentarios. Se implementó un filtrado colaborativo basado en modelos usando SVD, que descompone la matriz usuario–plato en matrices de menor dimensión representando factores latentes de gusto, capturando conceptos abstractos como "prefiere comida picante" o "disfruta opciones saludables", lo que permite recomendaciones cruzadas potentes y a veces sorpresivas [1] [12] [13].

III-B. Análisis de Sentimiento en Reseñas

El análisis de sentimiento se utiliza para interpretar automáticamente las emociones expresadas en las reseñas textuales de los usuarios. Esta técnica transforma comentarios subjetivos en variables numéricas que permiten enriquecer los datos de entrada del sistema de recomendación. Se utiliza el modelo **VADER**, un analizador de sentimiento ligero optimizado para textos breves y opiniones de usuarios, por su bajo costo computacional y buena performance en contextos de lenguaje informal como reseñas gastronómicas [2]. Calcula un puntaje compuesto de -1 (muy negativo) a $+1$ (muy positivo) para cada reseña, proporcionando una medida cuantificable del contexto emocional detrás de la calificación. El análisis de sentimiento se aplica justo después de la recepción de una reseña, categorizando el comentario como positivo, neutral o negativo, y asignando una puntuación de sentimiento que se integra junto con el rating numérico para ponderar el valor global de la experiencia registrada. Aunque una calificación de 5 estrellas es un buen indicador, carece de matices. Un comentario como "¡Increíble! La mejor pasta que he probado" aporta más información que solo la estrella. Este enfoque permite capturar matices en la satisfacción del usuario que impactan directamente en la visibilidad futura de los platos en el menú [14].

III-C. Curado Comunitario de Menús

En contextos donde los usuarios generan contenido, es necesario aplicar curado automático para evitar la sobrecarga de información y destacar las opciones más relevantes. El sistema propuesto emplea técnicas de ranking dinámico basadas en métricas objetivas de performance (ratings, cantidad de pedidos, sentimiento promedio) para construir un menú que refleje la inteligencia colectiva de la comunidad [15] [16]. Este enfoque asegura que un plato no se recomiende solo por la afinidad del usuario, sino que también se potencie si la comunidad tiene una respuesta positiva fuerte, aumentando la calidad percibida de la recomendación.

IV. Desarrollo

El núcleo de esta investigación estuvo orientado a responder la siguiente pregunta: ¿puede un motor de recomendación híbrido (SVD + VADER), integrado en una arquitectura ligera, producir listas personalizadas más pertinentes que un catálogo estático y con latencias compatibles con uso móvil? La hipótesis disparadora sostiene que la combinación de preferencias latentes inferidas por SVD con el sentimiento agregado de las reseñas (VADER) mejora la calidad percibida de las sugerencias y favorece el descubrimiento de nuevos platos. Para contrastar esta hipótesis se definió llevar a cabo una Prueba de Concepto (PoC) que materializa el motor y expone endpoints específicos para su evaluación.

El desarrollo de la PoC se estructuró en torno a tres componentes principales: la aplicación móvil, el backend y el motor de recomendación. Además, se definió un flujo de trabajo colaborativo basado en Git para gestionar el versionado del código.

IV-A. App móvil

La aplicación móvil se implementó en *Android*, dado a la experiencia previa que tengo en la plataforma y que ayudó a acelerar el desarrollo de la PoC. La app integra pantallas para visualizar el menú recomendado, crear nuevos platos y calificar opciones existentes. Las interacciones de la app se resuelven mediante llamadas a los endpoints del backend, consumiendo datos en formato JSON [3] [4] [17].

Desde el punto de vista de arquitectura, la app sigue el patrón **Model–View–ViewModel (MVVM)** [18], separando responsabilidades y facilitando testeo, mantenibilidad y escalabilidad. Las *Views* gestionan la UI y el ciclo de vida; los *ViewModels* exponen el estado y orquestan casos de uso; y la capa *Model/Repository* abstrae el acceso a datos remotos.

Para la capa de **networking** se utiliza **Retrofit** [19], por su facilidad de integración con *coroutines*, manejo de serialización JSON y soporte nativo para *interceptors*, lo que simplifica el versionado de la API y el registro de peticiones en esta PoC.

En la pantalla de **Menú** se incorporó un *filtro por categoría* (p. ej., *Vegetariano*, *Carne de Res*, *Pollo*) que permite al usuario acotar rápidamente las opciones y **comprar más rápido** dentro de su preferencia de ese momento. Para simplificar el flujo principal de la PoC y focalizar en el motor de recomendación, **sólo se permite comprar de a un plato por vez**; esta restricción reduce complejidad de estados (carrito, múltiples ítems, combinaciones/promociones) y facilita la evaluación del impacto del algoritmo en la conversión por sesión.

IV-B. Backend

El backend de *Cook My Meal!* funciona como el sistema nervioso central de la plataforma, manejando toda la lógica de negocio, la persistencia de datos y los cálculos complejos requeridos por el motor de recomendación. Fue diseñado para ser modular, escalable y altamente eficiente.

B.1. Stack Tecnológico

El backend está construido sobre un stack moderno de Python, elegido por su rendimiento, facilidad de desarrollo y ecosistema robusto:

- **Python 3.12**: Lenguaje principal, con un amplio conjunto de librerías para ciencia de datos y desarrollo web.
- **FastAPI**: Framework web asincrónico de alto rendimiento para construir APIs [20]. Sus ventajas principales:
  - Soporte asincrónico: Basado en Starlette y Pydantic, maneja conexiones concurrentes de forma eficiente.
  - Documentación automática de API: Genera documentación interactiva conforme a OpenAPI (en `/docs`), útil para pruebas, demos e integración con frontend.
  - Validación de datos: Usa *type hints* de Python para validar automáticamente requests y responses.
- **Uvicorn**: Servidor ASGI para correr la aplicación FastAPI [21].
- **SQLAlchemy**: ORM para mapear objetos Python [22] a la base de datos y permitir interacciones seguras.
- **NumPy**: Librería fundamental para operaciones matriciales en el algoritmo SVD [23].
- **VADER (vaderSentiment)**: Librería ligera para análisis de sentimiento de reseñas.

B.2. Arquitectura del Proyecto

El backend sigue una estructura modular basada en funcionalidades para asegurar separación de responsabilidades y mantenibilidad:

- `main.py`: Punto de entrada principal, inicializa la app FastAPI, conexión a DB y routers.
- `db.py`: Gestiona la conexión y ciclo de vida de la sesión de la base de datos.
- `models.py`: Contiene los modelos ORM de SQLAlchemy que definen las tablas.
- `svd_recommender.py`: Núcleo del motor de recomendación, con la clase `SVDRecommender`.
- `routers/`: Directorio con módulos separados por grupo lógico de endpoints:
  - `reco.py`: Lógica de recomendaciones.
  - `ratings.py`: Creación y recuperación de valoraciones y reseñas.
  - `purchases.py`: Registro de compras.
  - `dishes.py`: Gestión de platos.
  - `admin.py`: Endpoints administrativos.

B.3. Endpoints Clave de la API

Aunque la API expone operaciones CRUD completas, los siguientes endpoints son críticos:

1. `GET /reco/user/{user_id}` – **Recomendaciones Personalizadas**. Orquesta todo el proceso híbrido:
   - Llama a `get_svd_recommendations` para obtener $\hat{r}$ con factores precomputados.
   - Consulta las valoraciones para calcular el promedio de sentimiento de cada plato.
   - Calcula el score híbrido: score $= \hat{r} + (\alpha \cdot sentiment)$.
   - Filtra resultados con score menor a 3.5 y devuelve el top ordenado.
2. `POST /ratings` – **Envío de Reseñas y Análisis de Sentimiento**. Canal para recoger feedback explícito:
   - Recibe usuario, plato, estrellas (1–5) y reseña opcional.
   - Procesa reseñas textuales con VADER.
   - Persiste rating, texto y sentimiento en la tabla `ratings`.
3. `POST /purchases` – **Registro de Feedback Implícito**. Guarda un registro de compra (`user_id`, `dish_id`, cantidad, fecha), fundamental para usuarios que no dejan reseñas.
4. `POST /admin/retrain-model` – **Reentrenamiento Bajo Demanda**. Endpoint administrativo que ejecuta el script de retraining en segundo plano usando `BackgroundTasks`, devolviendo `202 Accepted` inmediatamente y manteniendo la API disponible.

IV-C. Base de Datos

La capa de persistencia de la plataforma *Cook My Meal!* es un componente crítico que sustenta todo el sistema. Para esta prueba de concepto, se eligió una tecnología de base de datos ligera y serverless para garantizar simplicidad y desarrollo rápido, mientras que el modelo de datos fue diseñado para ser robusto y escalable en un entorno de producción.

A.1. Tecnología de Base de Datos

El proyecto utiliza **SQLite** como motor de base de datos, gestionado a través del Object-Relational Mapper (ORM) **SQLAlchemy**. Esta combinación ofrece varias ventajas clave para un PoC:

- **Arquitectura Serverless:** SQLite funciona sin un proceso de servidor separado, almacenando toda la base de datos en un único archivo (`cookmymeal.db`). Esto elimina la necesidad de configuraciones complejas.
- **SQLAlchemy ORM:** Esta potente librería permite definir las tablas de la base de datos como clases de Python (modelos) e interactuar con la base de datos usando

objetos y métodos de alto nivel. Esto evita el uso directo de SQL, reduce código repetitivo y ayuda a prevenir vulnerabilidades de inyección SQL.
- **Portabilidad:** Al basarse en ORM, la aplicación no está fuertemente acoplada a SQLite. Migrar a una base de datos más robusta como PostgreSQL o MySQL requeriría cambios mínimos en el código.

A.2. Modelo de Datos y Esquemas de Tablas

La base de datos está compuesta por seis tablas principales, cada una diseñada para almacenar un tipo específico de información. Las relaciones entre estas tablas forman la columna vertebral de la lógica de la aplicación.

1. **Tabla users**: Almacena información esencial de cada usuario.

Columna	Descripción
id (PK)	Identificador único del usuario.
name	Nombre visible del usuario.
email	Correo electrónico único del usuario.

2. **Tabla dishes**: Contiene información de los platos.

Columna	Descripción
id (PK)	Identificador único del plato.
name	Nombre del plato.
ingredients	Descripción de los ingredientes.
preparation_steps	Instrucciones de preparación.
category	Categoría culinaria.
created_by (FK)	Usuario que creó el plato.
created_at	Fecha de creación.

3. **Tabla purchases**: Registra eventos de compra, fuente de feedback implícito.

Columna	Descripción
id (PK)	Identificador único de la compra.
user_id (FK)	Usuario que compró.
dish_id (FK)	Plato comprado.
quantity	Cantidad de unidades.
price	Precio del plato.
created_at	Fecha de la compra.

4. **Tabla ratings**: Almacena feedback explícito (estrellas y reseñas).

Columna	Descripción
id (PK)	Identificador único.
user_id (FK)	Usuario que valoró.
dish_id (FK)	Plato valorado.
rating	Valoración numérica (1–5).
review	Reseña escrita.
sentiment	Puntaje VADER (-1 a +1).
created_at	Fecha de la valoración.
uq_user_dish	Un usuario solo puede valorar un plato una vez.

5. **Tablas user_factors y dish_factors**: Almacenan los factores latentes del modelo SVD.
user_factors:

Columna	Descripción
user_id (PK, FK)	ID del usuario.
bias	Sesgo aprendido del usuario.
factors_json	Vector latente (JSON).

dish_factors:

Columna	Descripción
dish_id (PF, FK)	ID del plato.
bias	Sesgo aprendido del plato.
factors_json	Vector latente (JSON).

A.3. Relaciones entre Tablas

Las relaciones aseguran integridad y permiten consultas complejas:

- **Uno a Muchos:** Un usuario puede tener muchas compras y valoraciones; un plato puede estar en muchas compras y recibir muchas valoraciones.
- **Uno a Uno:** Un usuario tiene exactamente un registro en `user_factors`; un plato tiene exactamente un registro en `dish_factors`.

Esta estructura relacional permite unir datos de manera eficiente, por ejemplo, para obtener todas las valoraciones de un plato o construir la matriz usuario–plato necesaria para entrenar SVD combinando información de compras y valoraciones.

IV-D. Algoritmo de Recomendación

El motor de recomendación combina filtrado colaborativo y análisis de sentimiento:

- **Filtrado colaborativo:** se aplicó *Singular Value Decomposition (SVD)* para descomponer la matriz usuario–plato y predecir afinidades en platos no consumidos [1].
- **Análisis de sentimiento:** cada comentario textual se procesó con *VADER*, generando un puntaje entre -1 y 1 que refleja la percepción emocional. Este valor se normaliza y se integra al score final [2].
- **Datos de entrenamiento:** basados en la matriz usuario–plato usando valoraciones explícitas; si un usuario compró pero no calificó, se usa un valor neutral de 3,0 [24].
- **Reproducibilidad:** proceso determinista con semilla fija (`np.random.seed(42)`).
- **Reentrenamiento asíncrono:** endpoint administrativo (`/admin/retrain-model`) que ejecuta el reentrenamiento en segundo plano, sin bloquear la API.

El puntaje de recomendación para un usuario u y un plato i se calcula como:

$$\text{score}_{u,i} = \hat{r}_{u,i} + \alpha \cdot \text{sentimiento}_i$$

donde $\hat{r}_{u,i}$ es el valor predicho por SVD y α un factor de ponderación configurable.

IV-E. Gestión de Código y Flujo de Trabajo

El versionado del código se gestionó mediante **Git**, alojando el repositorio en GitHub. Para ordenar el trabajo se adoptó la convención *GitFlow*, que organiza el desarrollo en ramas principales y de soporte:

- **main:** contiene versiones estables y listas para producción de la PoC.
- **develop:** rama de integración donde se consolidan nuevas funcionalidades antes de ser promovidas a main.
- **feature/*:** ramas temporales para el desarrollo de nuevas funcionalidades (ej. feature/recommendations).

Este flujo permitió trabajar de manera incremental, integrando cambios de backend y frontend de forma sincronizada, reduciendo conflictos y asegurando trazabilidad en cada etapa. Las ramas se acompañaron de mensajes de commit claros y descriptivos, siguiendo convenciones de estilo que facilitan la comprensión del historial de cambios [25].

V. Conclusión

Este trabajo se desarrolla como una continuación directa del proyecto de investigación teórico titulado *Plataforma inteligente de platos personalizados con recomendación basada en ratings y reviews colaborativos*, donde se exploraron los fundamentos conceptuales y matemáticos detrás de un sistema híbrido de recomendación. A partir de aquel marco teórico, que detallaba los componentes del filtrado colaborativo (SVD) y del análisis de sentimiento (VADER), surgió la necesidad de trasladar esas ideas al plano práctico para verificar su factibilidad y comportamiento en un entorno real.

Con ese propósito, el presente trabajo propuso implementar una **Prueba de Concepto (PoC)** que materializara los conceptos previamente desarrollados. El objetivo fue demostrar que era posible construir una arquitectura liviana y eficiente —basada en *FastAPI*, *SQLite* y una aplicación móvil Android— capaz de integrar un motor híbrido de recomendaciones en tiempo real. En otras palabras, pasar de la teoría a la práctica, transformando los modelos analizados en un sistema funcional que ofreciera recomendaciones personalizadas a partir de ratings y reseñas textuales.

A través de la PoC, se logró validar la hipótesis planteada en la etapa de investigación: que la combinación de SVD y VADER mejora la pertinencia y el contexto emocional de las recomendaciones gastronómicas. El sistema resultante permitió integrar la capa analítica con la experiencia de usuario de manera fluida, demostrando que un motor híbrido puede operar con baja latencia y producir resultados coherentes y reproducibles. En síntesis, este trabajo marcó la transición de un proyecto teórico a un prototipo funcional, confirmando la viabilidad técnica del enfoque propuesto y estableciendo las bases para su futura expansión hacia un sistema completo de delivery inteligente y curado por la comunidad.

VI. Líneas Futuras de Investigación

Entre las principales extensiones de este trabajo se destacan:

- Incluir atributos nutricionales en el motor de recomendación.
- Implementar *clustering* de usuarios para segmentación por hábitos de consumo.
- Optimizar exploración de nuevos platos con algoritmos *multi-armed bandits*.
- Desplegar la solución en un entorno real, integrando autenticación, logística y pagos.
- Implementar técnicas de Business Intelligence (BI) como Canasta de Mercado, para recomendar productos que van de la mano [26].
- Ajustar tanto la aplicación móvil como el backend para permitir compras de más de un ítem o plato a la vez. En un entorno real, los usuarios realizan pedidos agrupados en una **orden de compra** o **pedido**, más que por un solo plato. Esto implicaría rediseñar la lógica del sistema para gestionar historiales de órdenes completas y, en consecuencia, habilitar la calificación de cada plato dentro del detalle del pedido.
- Agregar una pantalla adicional en la aplicación para **visualizar el detalle del pedido**, mostrando todos los platos incluidos, de manera que desde allí el usuario pueda puntuar individualmente cada ítem, extendiendo la lógica actual de ratings.
- Incorporar un esquema de **entrenamiento programado del modelo** utilizando un *cron job* u orquestador de tareas. Esto permitiría reentrenar el motor SVD de manera periódica (por ejemplo, cada cierto intervalo de horas o al final de cada día), actualizando los factores latentes con las interacciones más recientes sin intervención manual. De esta forma se asegura que las recomendaciones se mantengan frescas y relevantes, evitando ejecutar procesos de entrenamiento pesados en línea y reduciendo la latencia percibida por el usuario en la plataforma.
- Evolucionar el backend hacia un enfoque de **API como microservicio**, desacoplada del cliente móvil. Esto permitiría que la plataforma de recomendación pueda integrarse con cualquier otra aplicación o servicio de delivery existente, ofreciendo recomendaciones personalizadas a través de una integración estandarizada. De esta forma, Cook My Meal! podría posicionarse como un motor independiente que enriquece el ecosistema de delivery ya instalado.

VII. Experiencia y Recomendaciones

La realización de este trabajo representó una experiencia de aprendizaje integral, tanto técnica como metodológica. Abordar una problemática real desde un enfoque de investigación aplicada me permitió recorrer todas las etapas del ciclo de desarrollo de un sistema inteligente: desde la definición de la hipótesis y la delimitación del alcance, hasta la construcción, prueba y validación de una PoC funcional.

Durante el desarrollo de *Cook My Meal!*, adquirí una comprensión más profunda sobre cómo combinar técnicas de *machine learning* con ingeniería de software moderna. La implementación del motor híbrido de recomendación (SVD + VADER) me permitió internalizar los desafíos que implica llevar un modelo teórico a la práctica, tales como el manejo de datos incompletos, la reproducibilidad de resultados y la necesidad de diseñar un pipeline eficiente y escalable. Asimismo, trabajar con FastAPI y SQLAlchemy reforzó la importancia de adoptar tecnologías que equilibren rendimiento con simplicidad, especialmente en fases tempranas de validación.

En términos personales, esta experiencia fortaleció mis habilidades para planificar y ejecutar proyectos técnicos complejos

de manera estructurada, aplicando pensamiento crítico en la toma de decisiones tecnológicas y metodológicas. También me permitió afianzar competencias en documentación técnica, redacción científica y comunicación de resultados de investigación.

Consejos para futuros proyectos similares:

- **Comenzar simple, pero medible:** una PoC debe ser lo suficientemente acotada para completarse, pero también lo bastante rica para generar resultados observables y validables.
- **Priorizar la reproducibilidad:** fijar semillas aleatorias y registrar versiones de código y datasets desde el principio evita inconsistencias en fases posteriores.
- **Diseñar con modularidad:** separar el motor de recomendación del resto del sistema facilita su evolución hacia entornos de producción o integración con terceros.
- **Combinar teoría con práctica:** cada decisión técnica (por ejemplo, elección de SVD o VADER) debe sustentarse en fundamentos teóricos, pero probarse empíricamente en un entorno real.
- **Documentar continuamente:** mantener trazabilidad del código, supuestos y resultados simplifica tanto la comunicación con pares como la extensión futura del proyecto.
- **Versionar el código de manera incremental:** utilizar sistemas de control de versiones como *Git* y plataformas como *GitHub* permite desarrollar de forma evolutiva e incremental, manteniendo un registro claro de los cambios. Esto facilita la experimentación controlada, ya que es posible crear ramas para probar nuevas funcionalidades, revertir fácilmente modificaciones no deseadas y analizar el impacto de cada iteración antes de integrarla a la versión estable del proyecto.

En conjunto, estas prácticas no solo mejoran la calidad técnica del desarrollo, sino que también fortalecen el enfoque científico del proyecto. Aplicar una metodología de trabajo ordenada, medible y reproducible garantiza que los resultados obtenidos puedan ser verificados, extendidos y reutilizados por otros investigadores o equipos de desarrollo. De esta forma, cada iteración del sistema se convierte en un aporte incremental de conocimiento, consolidando la unión entre la práctica profesional y la investigación aplicada.

En síntesis, la ejecución de este trabajo no solo validó la hipótesis planteada, sino que también consolidó un conjunto de aprendizajes transferibles a otros contextos tecnológicos. La experiencia demuestra que el desarrollo de una PoC bien fundamentada puede ser una herramienta poderosa para transformar ideas de investigación en soluciones concretas y funcionales.

VIII. CV

Diego Exequiel Tabares - 0126981

REFERENCIAS

[1] J. K. B. Sarwar, G. Karypis and J. Riedl, "Application of dimensionality reduction in recommender system – a case study," *Proceedings of the ACM WebKDD Workshop*, pp. 82–90, 2000.

[2] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*, pp. 216–225, 2014.

[3] A. A. Alalwan, "Mobile food ordering apps: An empirical study of the factors affecting customer e-satisfaction and continued intention to reuse," *International Journal of Information Management*, vol. 50, pp. 28–44, 2020.

[4] S.-H. Wu, "Minds think alike: How do food delivery mobile applications innovate consumer service," *Journal of Tourism and Services*, vol. 13, no. 25, pp. 137–155, 2022.

[5] R. Y. Toledo, A. A. Alzahrani, and L. Martinez, "A food recommender system considering nutritional information and user preferences," *IEEE Access*, vol. 7, pp. 96 695–96 711, 2019.

[6] S. Hamdollahi Oskouei and M. Hashemzadeh, "Foodrecnet: a comprehensively personalized food recommender system using deep neural networks," *Knowledge and Information Systems*, vol. 65, no. 9, pp. 3753–3775, 2023.

[7] M. J. González Padrón *et al.*, "Sistema inteligente de recomendación de restaurantes," 2022.

[8] B. F. Rioseco Elgueta, "Sistema de recomendación de recetas saludables, basado en la preferencia y necesidad nutricional del usuario," 2020.

[9] I. G. González, "Trabajo de fin de grado desarrollo de una aplicación web full-stack para la planificación de menús."

[10] H. I. Chang Miranda, L. A. Diaz Viejo, and F. D. Ruiz Moncayo, "Sistema de predicción y recomendación personalizada basada en ranking de ítems homogéneos usando filtrado colaborativo," 2006.

[11] F. Méndez López *et al.*, "Sistemas de recomendación en el contexto gastronómico: elaboración y enriquecimiento de un dataset de recetas de cocina," Master's thesis, 2018.

[12] S. I. P. Guano and W. R. R. Lomas, "Análisis de las predicciones en un filtro colaborativo basado en el algoritmo als para una empresa de comida rápida en la ciudad de guayaquil," *Editorial Tecnocientífica Americana*, vol. 600, pp. 1–113, 2023.

[13] D. L. González Arroyave *et al.*, "Estandarización de datos y priorización de algoritmos para un sistema de recomendación," Ph.D. dissertation, Universidad EAFIT, 2019.

[14] D. Ribeiro, T. Barbosa, J. Ribeiro, F. Sousa, E. F. Vieira, M. Silva, and A. Silva, "Souschef system for personalized meal recommendations: A validation study," *Applied Sciences*, vol. 12, no. 2, p. 702, 2022.

[15] M. S. Becerra Sacramento and S. D. L. A. Davila Fernandez, "Plataforma digital que evita la pérdida y excedentes de alimentos en restaurantes de lima metropolitana," 2023.

[16] T. Islam, A. R. Joyita, M. G. R. Alam, M. M. Hassan, M. R. Hassan, and R. Gravina, "Human-behavior-based personalized meal recommendation and menu planning social system," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 4, pp. 2099–2110, 2022.

[17] A. Daza Culqui, H. M. Escalante Guerra, V. I. Méndez Cuadros, and A. A. Pacheco Vidal, "Fit4you–propuesta de implementación de una aplicación móvil de comida saludable personalizada en el distrito de ate," 2021.

[18] G. Developers, "Guide to app architecture: Model-view-viewmodel (mvvm)," https://developer.android.com/topic/architecture, 2019.

[19] S. Inc., "Retrofit: type-safe http client for android and java," https://square.github.io/retrofit/, 2013.

[20] S. Ramírez, "Fastapi: high performance, easy to learn, fast to code, ready for production," 2018.

[21] U. Developers, "Uvicorn: lightning-fast asgi server implementation, using uvloop and httptools," https://www.uvicorn.org/, 2018.

[22] M. Bayer, *SQLAlchemy: Database Access Using Python*. O'Reilly Media, 2020.

[23] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[24] W. McKinney, "Data structures for statistical computing in python," *Proceedings of the 9th Python in Science Conference*, pp. 51–56, 2010.

[25] V. Cosentino, J. L. C. Izquierdo, and J. Cabot, "A systematic mapping study of software development with github," *Ieee access*, vol. 5, pp. 7173–7192, 2017.

[26] J. J. M. Uribe, "Big data e inteligencia de negocios aplicado en el sector gastronómico," *Infometric@-Serie Ingeniería, Básicas y Agrícolas*, vol. 7, no. 1, 2024.