Escuela Técnica Superior de
Ingeniería Informática

TRABAJO FIN DE GRADO

Integración de la inteligencia artificial en el E-Commerce para la mejora de la experiencia de usuario y optimización de ventas.

Realizado por
Antonio Silva Gordillo

Para la obtención del título de
Grado en Ingeniería informática - Tecnologías informáticas

Dirigido por
Dr. Juan Antonio Ortega Ramírez

En el departamento de
Lenguajes y sistemas informáticos

Convocatoria de junio, curso 2024-2025

"Recuerda: la tecnología no es nada. Lo importante es tener fe en la gente, que son básicamente buenos e inteligentes, y si les das herramientas, harán cosas maravillosas."

— Steve Jobs

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi tutor, el Dr. Juan Antonio Ortega Ramírez, por su orientación y apoyo durante todo el proceso de realización de este proyecto. Su guía no solo me ha permitido profundizar en un tema tan fascinante como la inteligencia artificial aplicada al eCommerce, sino que también ha sido fundamental para encontrar el enfoque adecuado para este trabajo.

También quiero agradecer a mi familia, por su incondicional apoyo y comprensión durante los momentos más intensos del estudio de este grado que me han permitido llegar hasta aquí. Su paciencia y motivación constante han sido una fuente de energía que me ha permitido seguir adelante y superar los desafíos que se presentaron en el camino.

A mi pareja, le agradezco su constante ánimo y por ser mi pilar en los momentos más difíciles. Su apoyo emocional ha sido clave para mantener el equilibrio entre el trabajo y la vida personal durante este proceso.

Finalmente, quiero agradecer a todos mis amigos y compañeros de carrera, quienes me han acompañado a lo largo de este viaje académico. Sus palabras de ánimo, colaboración y las conversaciones compartidas han sido invaluables. Este trabajo no habría sido posible sin la red de apoyo que me ha rodeado. Gracias a todos.

Resumen

Este proyecto se desarrolló con el objetivo de diseñar e implementar una plataforma web con productos reales donde mostrar y testear los diferentes algoritmos de inteligencia artificial utilizados actualmente para crear sistemas avanzados de recomendación en plataformas líderes como *Amazon*, *Alibaba* o *Netflix*.

Para su desarrollo se ha utilizado como base *Django*, así como *Python* nativo acompañado de librerías específicas como *Pandas*, *NumPy* y*Hugging-faces* para el análisis y manipulación del catálogo de productos de *Amazon*, o *Scikit-learn* y *Surprise* para el desarrollo de los algoritmos de recomendación, basados principalmente en *KNN*, *SVD* y *Content-Based Filtering*. Se usó *Postgre* como base de datos.

La metodología seguida durante el desarrollo del proyecto fue en *cascada*, lo cual permitió una planificación clara y secuencial.

Durante la implementación del sistema web se afrontaron algunos retos importantes, como trabajar con grandes volúmenes de datos y entrenar algoritmos de inteligencia artificial en un ordenador convencional. Para superarlos, se elaboró un proceso ETL optimizado y eficiente, además del apoyo en funciones externas que permitían el consumo de datasets masivos mediante streaming y virtualización de datos. También se usaron modelos preentrenados para reducir el tiempo y la carga del sistema, logrando un modelo de recomendación funcional.

Finalmente, se obtuvo una plataforma completa y robusta que permite explorar y entender cómo se comportan estos algoritmos en un entorno web con productos reales, descubriendo el potencial de estas técnicas para mejorar las ventas en eCommerce, personalizando y mejorando la experiencia del usuario.

Palabras clave: Inteligencia artificial, eCommerce, sistema de recomendación, Django, Python, asprendizaje automático

Abstract

This project was developed with the aim of designing and implementing a web platform with real products to showcase and test the different artificial intelligence algorithms currently used to build advanced recommendation systems on leading platforms such as *Amazon*, *Alibaba*, or *Netflix*.

For its development, *Django* was used as the main framework, along with native *Python* and specific libraries such as *Pandas*, *NumPy*, and *Datasets* for the analysis and manipulation of *Amazon*'s product catalog, as well as *Scikit-learn* and *Surprise* for the implementation of recommendation algorithms, mainly based on *KNN*, *SVD*, and *Content-Based Filtering*. *PostgreSQL* was used as the database.

The methodology followed during the development of the project was the waterfall model, which allowed for clear and sequential planning. During the implementation of the web system, several significant challenges were faced, such as handling large volumes of data and training artificial intelligence algorithms on a conventional computer. To overcome these obstacles, an optimized and efficient ETL process was designed, along with the use of external functions that enabled the consumption of massive datasets through streaming and data virtualization. Pre-trained models were also employed to reduce system load and development time, resulting in a functional recommendation model.

In the end, a complete and robust platform was achieved, enabling users to explore and understand how these algorithms behave in a web environment with real products, thus demonstrating the potential of these techniques to boost eCommerce sales by personalizing and enhancing the user experience.

Keywords: *Artificial intelligence*, *eCommerce*, *personalized recommendation systems*, *machine learning*, *Django*, *Python*.

Índice general

Índice de figuras

Índice de tablas

Índice de extractos de código

1. Introducción

En este capítulo se presenta la temática en la que se basa el proyecto, así como el contexto, la motivación y los principales objetivos marcados en su desarrollo. Al final se presenta la estructura completa del mismo.

1.1. Contexto

El contexto de este trabajo de fin de grado se enmarca en la inteligencia artificial (IA) y cómo en los últimos años ha adquirido un papel fundamental en la transformación digital de diversas industrias, y el *ecommerce* no es la excepción. Las empresas que operan en este ámbito buscan constantemente mejorar la experiencia de usuario y optimizar sus procesos de venta.

Se ha elegido un sistema recomendador como la herramienta de IA a implementar debido a la importancia que tienen en aumentar métricas clave en el comercio electrónico, como son las tasas de conversión, el valor medio del carrito y la fidelización del usuario, sobre todo cuando se implementa con técnicas de marketing efectivas como el *upselling* y el *cross-selling*.

El desarrollo del sistema y del prototipo de *ecommerce* se realizará utilizando tecnologías modernas como *Django* y *Python* para la construcción del backend, y librerías especializadas en IA como *Scikit-Learn* o *Surprise* para el entrenamiento y despliegue de los modelos de recomendación. El frontend se implementará con *Bootstrap*, buscando ofrecer una interfaz intuitiva y eficiente que facilite la interacción del usuario con la plataforma.

1.2. Motivación

Mi motivación inicial para desarrollar este proyecto viene de mi interés en explorar el mundo de la inteligencia artificial debido a su gran repercusión, que aunque para nada es una "nueva tecnología" (ya que los primeros modelos y algoritmos se empezaron a desarrollar en los años 70), sí que ha tenido un avance sin precedentes en los últimos años. No quería quedarme solo en el plano teórico, sino desarrollar un sistema completo donde aplicar la inteligencia artificial de forma real.

Impulsado por la propuesta de mi tutor Juan Antonio Ortega Ramírez de basarlo en *Python* y los algoritmos de aprendizaje automático, me decanté por los sistemas de recomendación que tanto han revolucionado la optimización de ventas en grandes plataformas en los últimos años, ya que es una aplicación muy potente de esta tecnología.

El resultado es este proyecto, que me ha permitido afianzar y poner en práctica los conocimientos y habilidades técnicas adquiridos durante el grado, generando una herramienta con impacto positivo en la sociedad, que mejora tanto las ventas como la retención y la experiencia de los usuarios de la plataforma donde se implemente.

1.3. Objetivos del proyecto

Este trabajo pretende demostrar cómo la integración de la inteligencia artificial en el *ecommerce* no solo mejora la experiencia de los usuarios, sino que también puede tener un impacto significativo en las métricas de negocio, aumentando la eficiencia operativa y optimizando los ingresos.

El proyecto no se limitará al desarrollo teórico del sistema de IA, sino que también se llevará a cabo la implementación práctica de este en un prototipo real de plataforma de *ecommerce*, desarrollado específicamente para este propósito. Este prototipo servirá como un entorno de prueba donde se podrán evaluar de manera efectiva los beneficios de la inteligencia artificial en la experiencia del usuario y la eficiencia del proceso de ventas, marcado por el potencial de las recomendaciones ajustadas a cada usuario.

Para entrenar el sistema y montar la plataforma se ha usado un dataset de más de 150.000 productos reales extraídos de Amazon, al cual se le ha aplicado un proceso ETL para poder cargarlos en el sistema de forma correcta.

1.4. Estructura de la memoria

Este documento recoge todo el desarrollo e implementación del proyecto desde el inicio hasta su finalización. El trabajo está compuesto por nueve partes generales que se detallan a continuación:

Parte 1: Introducción
En este primer bloque, se introduce el tema del proyecto y los objetivos principales.

Parte 2: Estudio previo
Se presenta un análisis general de los antecedentes y el marco teórico. Se detallan los fundamentos y conceptos clave en el proyecto, así como los objetivos específicos y la metodología elegida para el correcto desarrollo de este.

Parte 3: Estado del arte
Esta sección expone los enfoques y técnicas más representativas en el diseño de sistemas de recomendación, utilizados con el objetivo de predecir las preferencias de los usuarios a partir de datos.

Se detallan desde técnicas clásicas como la factorización de matrices hasta modelos probabilísticos y redes neuronales. También se analizan los principales

desafíos y problemas asociados a su implementación, así como estrategias probadas para mitigarlos.

Finalmente, se presentan casos de uso en plataformas digitales reconocidas, que muestran la aplicación real y el impacto de estas tecnologías en el comercio electrónico y los servicios de contenido digital *mainstream*.

Parte 4: Tecnologías
En este capítulo y desde una perspectiva más técnica, se describen en detalle las tecnologías utilizadas durante el desarrollo. Se proporciona una visión en detalle de *Python* junto con las principales librerías que hemos usado, también de *PostgreSQL*, *Django* y *Bootstrap*, destacando sus características, cualidades y arquitectura.

Parte 5: Análisis
En este apartado se estudian en profundidad los requisitos tanto funcionales como no funcionales, la lógica de negocio y la arquitectura del sistema, de forma que aseguren un diseño y desarrollo que satisfaga las necesidades del usuario final.

Parte 6: Planificación
Esta sección explica cómo se han organizado las tareas, recursos y tiempos para alcanzar los objetivos definidos y cumplir el cronograma, así como la estimación de los costes del desarrollo de este proyecto.

Parte 7: Diseño
Esta sección aborda el diseño y la arquitectura técnica del sistema, detallando su estructura mediante diagramas y modelos de datos, así como la integración de las tecnologías utilizadas, con el objetivo de asegurar una solución escalable, eficiente y mantenible.

Parte 8: Implementación
Este apartado expone detalladamente el proceso de desarrollo e implementación del sistema, desde el preprocesamiento del conjunto de datos hasta la construcción del *pipeline* de datos y la aplicación web.

Se explican las distintas etapas del desarrollo, las decisiones técnicas tomadas y los principales retos enfrentados, incluyendo fragmentos de código y diagramas que muestran cómo se llevó a cabo el desarrollo de la plataforma.

Parte 9: Cierre
Aquí se exponen las conclusiones extraídas de este proyecto, con especial foco en los logros alcanzados en relación con los objetivos iniciales, así como en los problemas y desafíos que surgieron durante su desarrollo.

2. Estudio previo

2.1. Introducción

En este capítulo se presenta un análisis general de los antecedentes y la fundamentación teórica que sustentan el desarrollo del presente proyecto. Se examina cómo la inteligencia artificial (IA) ha sido aplicada en el ámbito del comercio electrónico (eCommerce), considerando los avances más recientes y las mejores prácticas en la implementación de sistemas de recomendación y estrategias de marketing automatizadas. Además, se definen los objetivos específicos del proyecto, se describe la metodología empleada y se sientan las bases conceptuales necesarias para comprender la relevancia de la solución propuesta.

2.2. Objetivos

El objetivo principal de este proyecto es diseñar e implementar un sistema de inteligencia artificial en un prototipo real de eCommerce, con el propósito de mejorar la experiencia del usuario y optimizar las ventas. Para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

- Investigar y analizar las tecnologías y algoritmos de inteligencia artificial más efectivos para su aplicación en el ámbito del eCommerce.
- Diseñar y desarrollar un sistema de recomendaciones personalizadas adaptado a las preferencias y comportamientos de los usuarios.
- Integrar estrategias de marketing automatizadas que respondan en tiempo real a las tendencias y dinámicas del mercado.
- Implementar y evaluar el sistema en un entorno de prueba basado en un prototipo real de eCommerce.
- Validar la efectividad del sistema mediante pruebas, métricas de rendimiento y análisis de resultados.

2.3. Metodología

Para este proyecto se ha elegido una metodología de desarrollo en cascada. Este es un modelo de desarrollo de software secuencial y estructurado que sigue una serie de fases bien definidas, las cuáles deben de finalizarse completamente antes de pasar a la siguiente. Algunas de las ventajas son las siguientes:

- **Simplicidad y claridad**: Al ser lineal facilita tanto la planificación como el desarrollo del proyecto, sobre todo para equipos pequeños donde es complicado paralelizar tareas.
- **Documentación consistente**: Cada fase genera documentación completa por lo que se mejora la comprensión tanto de la fase concreta como del propyecto en genral tanto en la ejecuciñon de este como en su posterior evaluación.
- **Facilidad de gestión**: Al tener etapas claramente definidas, es más fácil asignar tareas, establecer hitos y controlar tiempos y recursos.
- **Requisitos bien definidos**: Funciona bien cuando los requisitos están bien definidos desde el inicio y es poco probable que cambien a lo largo de su ejecución.

He elegido este modelo porque permite una planificación clara desde el inicio, lo que es ideal cuando se dispone de un tiempo limitado y unos requisitos bien definidos y que no cambiará significativamente a lo largo de la ejecución. Este enfoque facilita tanto su ejecución y seguimiento como su posterior evaluación

Las principales fases metodológicas contempladas son las siguientes:

- **Investigación**: Revisión exhaustiva de la literatura relacionada con la aplicación de IA en eCommerce y análisis comparativo de tecnologías y algoritmos disponibles.
- **Diseño del sistema**: Definición de la arquitectura general del sistema y diseño detallado de los componentes y módulos clave.
- **Desarrollo e implementación**: Codificación e integración del sistema de IA en el prototipo de eCommerce, asegurando su funcionalidad y escalabilidad.
- **Pruebas y validación**: Realización de pruebas unitarias, pruebas de integración y validación del sistema en términos de usabilidad, rendimiento y mejora en indicadores de negocio.
- **Documentación y análisis de resultados**: Recopilación de datos relevantes, análisis de resultados obtenidos y elaboración de la documentación final, incluyendo conclusiones y recomendaciones.

2.4. Fundamentación teórica

La inteligencia artificial se ha consolidado como una herramienta clave para la transformación digital de numerosos sectores, destacando especialmente su impacto en el comercio electrónico. La capacidad de los sistemas basados en IA para procesar grandes volúmenes de datos, identificar patrones complejos y automatizar decisiones ha permitido a las empresas ofrecer experiencias más personalizadas y eficientes a sus clientes.

Uno de los campos donde la IA ha mostrado mayor relevancia es en el desarrollo de sistemas de recomendación. Estos sistemas tienen como finalidad

predecir y sugerir productos o servicios que resulten de interés para los usuarios, tomando en cuenta información diversa, como el historial de comportamiento, preferencias declaradas, o características de los productos. En el contexto del eCommerce, su implementación no solo mejora la experiencia del cliente, sino que también contribuye al aumento de las ventas, la fidelización y la competitividad empresarial.

A lo largo del tiempo, se han desarrollado diversas metodologías para la optimización de estos sistemas. Desde enfoques tradicionales, como el filtrado colaborativo y los sistemas basados en contenido, hasta modelos más sofisticados que integran métodos híbridos y técnicas de aprendizaje profundo, cada uno de ellos ofrece ventajas específicas y plantea desafíos técnicos. El avance en tecnologías de procesamiento de datos y el desarrollo de algoritmos de aprendizaje automático han sido determinantes para perfeccionar estas metodologías y adaptarlas a las exigencias cambiantes de los mercados digitales.

El análisis de casos prácticos de grandes plataformas líderes en el sector permite observar cómo la aplicación estratégica de la inteligencia artificial, en combinación con sistemas de recomendación y técnicas de marketing automatizado, ha sido clave para su éxito. Estas plataformas han logrado integrar soluciones avanzadas capaces de adaptarse en tiempo real a las preferencias y comportamientos de los usuarios, ofreciendo experiencias altamente personalizadas.

Esta fundamentación teórica proporciona el marco conceptual necesario para comprender la importancia y las aplicaciones de la inteligencia artificial en el diseño e implementación de sistemas de recomendación dentro del comercio electrónico. Asimismo, establece las bases para el desarrollo del sistema propuesto en este proyecto, cuyo objetivo principal es aprovechar estas tecnologías emergentes para optimizar la experiencia del usuario y mejorar los procesos de venta.

3. Estado del arte

3.0.1. Inteligencia artificial aplicada al eCommerce

Definición y evolución de la IA en el comercio electrónico

La inteligencia artificial (IA) ha experimentado un crecimiento acelerado en el comercio electrónico, transformándose desde herramientas básicas de automatización hasta sistemas avanzados de personalización.

Inicialmente, la IA se aplicaba en la gestión de inventarios y en la automatización de tareas repetitivas, lo que mejoraba la eficiencia operativa. Con el desarrollo de algoritmos de aprendizaje profundo y el incremento de datos disponibles, se han implementado sistemas de recomendación, análisis predictivo, chatbots y asistentes virtuales.

Estos avances buscan ofrecer experiencias de usuario personalizadas y optimizar estrategias de marketing, lo que incrementa la competitividad en un mercado digital en constante cambio.

Aplicaciones clave de la IA en el eCommerce

- **Chatbots y asistentes virtuales**: Utilizando técnicas de Procesamiento de Lenguaje Natural (NLP), los chatbots responden de forma automatizada y personalizada a las consultas de los usuarios. A través del aprendizaje continuo de interacciones previas, estos sistemas mejoran la precisión de sus respuestas y predicen necesidades de los clientes. Por ejemplo, Luisa quiere comprar una blusa en HM, pero tiene dudas sobre las tallas disponibles en su país y las políticas de devolución. En lugar de buscar entre las preguntas frecuentes o contactar con el equipo de soporte, seguramente después de un perirodo de espera largo, inicia una conversación con el chatbot en la web. El asistente le responde de forma inmediata y amable, mostrándole una guía de tallas personalizada según su ubicación y explicándole el procedimiento para devolver productos. Gracias a esta interacción, Luisa se siente más segura y finaliza su compra con confianza. El asistente al estar entrenado con los datos reales de la empresa y con problemas similares de otros clientes, es capaz de resolver la incidencia en menos tiempo, sin errores y está disponible las 24 horas del día en todos los idiomas.
- **Análisis de sentimientos**: Permite a las empresas analizar grandes volúmenes de opiniones y comentarios para identificar tendencias y posibles problemas emergentes. Mediante técnicas de NLP, se pueden analizar datos de reseñas y redes sociales, ajustando en tiempo real estrategias de marketing. Por ejemplo, eBay detecta un aumento en comentarios negativos en redes sociales relacionados con la batería de un dispositivo móvil lanzado recientemente.

Utilizando el análisis de sentimientos, identifica rápidamente la causa del malestar y decide retirar temporalmente los anuncios de ese modelo, mientras lanza una campaña informativa para resolver dudas. Esta capacidad de reacción rápida minimiza el impacto negativo en la marca.

- **Sistemas de recomendación**: Emplean algoritmos de aprendizaje profundo y filtrado colaborativo para personalizar la experiencia de compra sugiriendo productos relevantes. En plataformas como Amazon, los sistemas de recomendación analizan tanto el comportamiento de los usuarios como las características de los productos, incrementando la probabilidad de compra y mejorando la fidelización. Por ejemplo, Pedro está buscando unos auriculares en Amazon. Después de visitar un par de modelos, la plataforma le sugiere otros auriculares con mejores valoraciones, compatibles con su móvil y dentro del presupuesto que Pedro suele tener. Además, le muestra accesorios complementarios como fundas para el móvil o estuches para los nuevos auriculares. Pedro encuentra justo lo que buscaba sin necesidad de seguir buscando, y termina comprando más de lo que planeaba inicialmente gracias a las recomendaciones personalizadas. Pedro se va con buenas sensaciones por que esos productos realmente eran necesarios para el pero no había reparado en ello antes de que el sistema se lo recomendara.

  Conbinar estos sistemas con estrategias de ventas online eficaces como los bundles (crear packs de 2 o mas productos con un descuento si se compran en ese momento juntos), el upselling (ofrecer un producto de versión o categoría superior más caro al que esta considerando) o el cross-selling (sugerir productos relacionados o complementarios en la misma frnaja de precios) como en el ejemplo anterior, permite maximizar el valor medio de cada pedido y mejorar la experiencia de compra al anticiparse a las necesidades del cliente.

3.0.2. Sistemas de recomendación en eCommerce

Los sistemas de recomendación son herramientas que ayudan a los usuarios a descubrir productos, servicios o información que puedan ser de su interés. Estos sistemas son ampliamente utilizados en plataformas como Netflix, Amazon, Spotify, entre otros, para personalizar la experiencia del usuario.

Filtrado colaborativo: Definición y concepto teórico

El filtrado colaborativo es una técnica de recomendación que se basa en las interacciones y preferencias de múltiples usuarios para predecir y recomendar elementos que podrían ser de interés para un usuario específico. A diferencia de otros enfoques, como el filtrado basado en contenido, el filtrado colaborativo no requiere información explícita sobre los atributos de los ítems, sino que se apoya en las relaciones entre usuarios e ítems.

El fundamento del filtrado colaborativo consiste en agrupar usuarios por intereses similares, de manera si un grupo de usuarios ha mostrado preferencias

similares en el pasado, es probable que tengan preferencias similares en el futuro. De esta manera, las recomendaciones se generan a partir de las similitudes entre usuarios o entre ítems.

Tipos de filtrado colaborativo

Filtrado colaborativo basado en vecinos

Basado en usuarios (User-Based)

- Se centra en encontrar usuarios similares al usuario objetivo y recomendar ítems que estos usuarios hayan valorado positivamente.
- Utiliza métricas como la similitud del coseno, correlación de Pearson o distancia euclidiana para medir la similitud entre usuarios.
- Identifica los *k* usuarios más similares al usuario objetivo.
- Agrega las preferencias de los vecinos para recomendar ítems no vistos por el usuario objetivo.

Por ejemplo, si el usuario A y el usuario B han calificado positivamente una serie de películas similares, se podría recomendar al usuario A una película que el usuario B ha calificado altamente pero que el usuario A aún no ha visto.

Basado en ítems (Item-Based)

- En lugar de buscar usuarios similares, este método busca ítems similares a aquellos que el usuario ha mostrado interés.
- Se utilizan métricas similares a las del enfoque basado en usuarios para medir la similitud entre ítems, recomendando ítems que son similares a los que el usuario ha valorado positivamente.

Por ejemplo, si un usuario ha disfrutado de una película como "El señor de los anillos", el sistema puede recomendar otras películas de ciencia ficción con temáticas similares como la serie "Juego de tronos".

Filtrado colaborativo basado en modelos

a. **Factores de matriz (Matrix factorization)**

La factorización de matrices se origina en los modelos de factores latentes, los cuales postulan la existencia de variables ocultas que caracterizan tanto a los usuarios como a los productos. Por ejemplo, es posible representar películas mediante dos factores numéricos: una dimensión que oscila entre comedia y seriedad, y otra entre realidad y fantasía.

En la primera dimensión, un valor elevado indica una película más seria, mientras que un valor bajo sugiere una mayor inclinación hacia la comedia. De

manera análoga, en la segunda dimensión, un valor alto refleja una mayor tendencia hacia la fantasía, y un valor bajo denota un apego a la realidad. Estos factores latentes también se aplican a los usuarios; por ejemplo, un usuario con un valor alto en la primera dimensión preferirá películas serias, y uno con un valor alto en la segunda dimensión mostrará predilección por películas de fantasía.

Con esta representación, tanto usuarios como películas pueden modelarse mediante vectores. Un usuario que disfruta significativamente de películas serias y de fantasía podría ser representado por el vector:

$$\mathbf{u}_1 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Por otro lado, un usuario que prefiere comedias y es indiferente entre fantasía y realidad podría ser descrito por:

$$\mathbf{u}_2 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$$

Del mismo modo, podemos asignar vectores a dos películas específicas: una película seria y de fantasía como *El Señor de los Anillos*, y una comedia realista como *Mean Girls*. Sus vectores serían:

$$\mathbf{p}_1 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$$

Para modelar las calificaciones que cada usuario otorgaría a cada película, se utiliza la combinación lineal de los vectores de usuarios y películas. Por ejemplo, la calificación que el usuario $\mathbf{u}_1$ daría a la película $\mathbf{p}_1$ se calcula mediante el producto punto:

$$\mathbf{u}_1^T \mathbf{p}_1 = 3 \times 2 + 4 \times 4 = 6 + 16 = 22$$

Si se hace esto para todos los usuarios y películas, se tiene el siguiente sistema lineal:

$$\mathbf{R} = \mathbf{U}\mathbf{P}^T,$$

donde

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ -2 & 0 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ -3 & -3 \end{bmatrix}$$

Y

$$\mathbf{R} = \begin{bmatrix} \mathbf{u}_1^T \mathbf{p}_1 & \mathbf{u}_1^T \mathbf{p}_2 \\ \mathbf{u}_2^T \mathbf{p}_1 & \mathbf{u}_2^T \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} 22 & -21 \\ -4 & 6 \end{bmatrix}$$

Este modelo refleja que el usuario $\mathbf{u}_1$, quien prefiere películas serias y de fantasía, otorga una calificación más alta a *El Señor de los Anillos* que a *Mean Girls*. Contrariamente, el usuario $\mathbf{u}_2$ aficionado a las comedias, muestra la preferencia opuesta.

En la práctica, los valores de las matrices $\mathbf{U}$ y $\mathbf{P}$ son desconocidos; únicamente disponemos de ciertas entradas de la matriz $\mathbf{R}$. El desafío consiste en estimar de manera precisa las matrices $\mathbf{U}$ y $\mathbf{P}$.

Este problema es una instancia de la descomposición en valores singulares (SVD), con la peculiaridad de que no se tienen todas las entradas de la matriz $\mathbf{R}$, sino solo una pequeña proporción de ellas.

El objetivo es, entonces, encontrar matrices $\mathbf{U}$ y $\mathbf{P}$ de rango K que satisfagan la aproximación $\mathbf{R} \approx \mathbf{U}\mathbf{P}^T$. De este modo, la calificación estimada que el usuario i asigna al artículo j se expresa como:

$$\hat{r}_{ij} = \mathbf{u}_i^T \mathbf{p}_j$$

Para evaluar la proximidad entre el producto de matrices y $\mathbf{R}$, se utilizan diferentes funciones de pérdida. Una métrica común es la raíz del error cuadrático medio (RMSE), calculada como:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{A}|} \sum_{(i,j) \in \mathcal{A}} (r_{ij} - \mathbf{u}_i^T \mathbf{p}_j)^2}$$

donde r_{ij} es la calificación real del usuario i al artículo j, $\mathcal{A}$ es el conjunto de pares (i, j) para los cuales se conoce r_{ij}, y $|\mathcal{A}|$ es el número total de calificaciones.

En esencia, el RMSE representa la raíz cuadrada del promedio de los cuadrados de las diferencias entre las calificaciones estimadas y las reales.

Otra métrica frecuentemente utilizada es el error absoluto medio (MAE), definido por:

$$\text{MAE} = \frac{1}{|\mathcal{A}|} \sum_{(i,j) \in \mathcal{A}} |r_{ij} - \mathbf{u}_i^T \mathbf{p}_j|$$

El RMSE suele ser preferible en contextos donde los errores de predicción pequeños no son significativos, ya que penaliza de manera más severa los errores grandes en comparación con el MAE.

Para minimizar el RMSE, el problema se plantea como:

$$\min_{\mathbf{U},\mathbf{P}} \sum_{(i,j)\in\mathcal{A}} \left[(r_{ij} - \mathbf{u}_i^T \mathbf{p}_j)^2 + \lambda \left(\|\mathbf{p}_j\|^2 + \|\mathbf{u}_i\|^2 \right) \right]$$

donde λ es un parámetro de regularización que ayuda a prevenir el sobreajuste al penalizar la complejidad de los vectores de usuarios y productos.

b. **Modelos probabilísticos**

Los modelos probabilísticos en el filtrado colaborativo se basan en la teoría de la probabilidad para modelar las interacciones entre usuarios e ítems. Estos modelos asumen que las interacciones observadas (como calificaciones o clics) son el resultado de procesos estocásticos subyacentes que pueden ser capturados mediante distribuciones de probabilidad. La ventaja principal de este enfoque es su capacidad para manejar la incertidumbre y proporcionar estimaciones probabilísticas de las preferencias de los usuarios, lo que puede mejorar la robustez y la interpretabilidad de las recomendaciones.

I. **Modelos de mezcla y factorización de matrices bayesianas**

Uno de los enfoques más destacados dentro de los modelos probabilísticos es la factorización de matrices bayesianas. Estos modelos extienden las técnicas de factorización de matrices tradicionales (como Singular Value Decomposition, SVD) incorporando priors bayesianos sobre los factores latentes, lo que permite una inferencia más flexible y una mejor generalización en escenarios con datos escasos.

II. **Factorización de matrices bayesianas**

En la factorización de matrices bayesianas, se modela la matriz de interacciones R (donde $R_{u,i}$ representa la interacción del usuario u con el ítem i) como el producto de dos matrices latentes U y V, donde cada entrada de U y V está distribuida de acuerdo con una distribución previa. Formalmente, se puede expresar como:

$$R_{u,i} \sim \mathcal{N}(U_u V_i^T, \sigma^2)$$

donde U_u y V_i son los vectores latentes para el usuario u y el ítem i, respectivamente, y $\mathcal{N}$ denota una distribución normal con media $U_u V_i^T$ y varianza σ^2. Además, se definen priors sobre U y V, típicamente gaussianos:

$$U_u \sim \mathcal{N}(0, \lambda_U^{-1} I)$$
$$V_i \sim \mathcal{N}(0, \lambda_V^{-1} I)$$

Este enfoque permite la incorporación de conocimiento previo y la regularización automática, ayudando a prevenir el sobreajuste y mejorando la capacidad del modelo para generalizar a nuevos datos.

III. **Inferencia y aprendizaje**:

La inferencia en modelos de factorización de matrices bayesianas generalmente se realiza mediante métodos de inferencia variacional o Markov Chain Monte Carlo (MCMC), que buscan aproximar la distribución posterior de los factores latentes U y V dado los datos observados. El objetivo es maximizar la probabilidad posterior de los parámetros latentes, lo que se logra iterativamente ajustando las estimaciones de U y V para minimizar la divergencia de Kullback-Leibler entre la distribución aproximada y la distribución posterior real.

Ventajas:

- **Manejo de incertidumbre**: Proporciona estimaciones probabilísticas que capturan la incertidumbre en las predicciones.
- **Regularización automática**: Los priors bayesianos ayudan a evitar el sobreajuste, especialmente en escenarios con datos escasos.
- **Flexibilidad**: Permite la incorporación de diferentes tipos de datos y priors personalizados según el dominio de aplicación.

Desventajas:

- **Complejidad computacional**: Los métodos de inferencia probabilística pueden ser computacionalmente costosos, especialmente para grandes conjuntos de datos.
- **Escalabilidad**: A medida que aumenta la dimensión de la matriz de interacciones, el costo de la inferencia puede volverse prohibitivo.

c. **Redes neuronales**

Las redes neuronales han emergido como una herramienta poderosa en el filtrado colaborativo, gracias a su capacidad para modelar relaciones no lineales y capturar patrones complejos en los datos de interacción entre usuarios e ítems. Las arquitecturas de redes neuronales profundas permiten aprender representaciones ricas y abstractas de los usuarios e ítems, mejorando significativamente la calidad de las recomendaciones.

I. **Autoencoders para recomendaciones**: Los autoencoders son una clase de redes neuronales utilizadas para aprender representaciones compactas de los datos mediante un proceso de codificación y decodificación. En el contexto de sistemas de recomendación, los autoencoders se utilizan para aprender representaciones latentes de los usuarios e ítems a partir de la matriz de interacciones R.

II. **Arquitectura de autoencoders**: Un autoencoder típico consta de dos partes principales:

- **Encoder**: Mapea la entrada (por ejemplo, las interacciones de un usuario) a un espacio latente de menor dimensión.

- **Decoder**: Reconstruye la salida a partir de la representación latente, intentando replicar la entrada original.

En sistemas de recomendación, el encoder puede tomar las interacciones de un usuario con diferentes ítems y mapearlas a una representación latente que capture las preferencias subyacentes del usuario. El decoder, a su vez, intenta predecir las interacciones faltantes o futuras del usuario a partir de esta representación latente.

Ventajas de los autoencoders:

- **Captura de relaciones complejas**: Pueden modelar interacciones no lineales y relaciones complejas entre usuarios e ítems.
- **Reducción de dimensionalidad**: Aprenden representaciones latentes compactas que facilitan el procesamiento y la recomendación.
- **Flexibilidad**: Pueden integrarse con otros tipos de datos y características adicionales para mejorar las recomendaciones.

III. **Redes neuronales convolucionales (CNNs) para recomendaciones**: Las CNNs, conocidas por su éxito en el procesamiento de datos espaciales como imágenes, también se han adaptado para sistemas de recomendación. En este contexto, las CNNs pueden capturar patrones locales y relaciones espaciales en las interacciones usuario-ítem, mejorando la precisión de las recomendaciones.

Aplicación de CNNs en recomendaciones:

- **Captura de patrones espaciales**: Las CNNs pueden identificar patrones locales en la matriz de interacciones que podrían pasar desapercibidos para otros modelos.
- **Integración con datos de contenido**: Las CNNs pueden combinar datos de interacción con datos de contenido (como imágenes de productos) para generar recomendaciones más ricas y contextuales.

IV. **Ejemplos de redes neuronales en recomendaciones**:

- **DeepCoNN**: Una arquitectura que utiliza dos redes neuronales convolucionales para modelar las interacciones de usuarios e ítems por separado antes de combinarlas para predecir calificaciones.
- **Neural Collaborative Filtering (NCF)**: Combina técnicas de filtrado colaborativo con redes neuronales profundas para aprender interacciones no lineales entre usuarios e ítems.

V. **Ventajas de las redes neuronales**:

- **Capacidad de aprender representaciones complejas**: Pueden capturar interacciones no lineales y relaciones de alta dimensión entre usuarios e ítems.
- **Escalabilidad y flexibilidad**: Las arquitecturas pueden adaptarse y escalarse según la complejidad de los datos y las necesidades del

sistema.

- **Integración de múltiples fuentes de información**: Permiten la incorporación de datos adicionales (como contenido multimedia) para enriquecer las recomendaciones.

VI. **Desventajas**:

- **Requerimientos computacionales**: Las redes neuronales profundas requieren recursos computacionales significativos para el entrenamiento y la inferencia.
- **Complejidad de la implementación**: La configuración y optimización de arquitecturas de redes neuronales puede ser compleja y requiere experiencia técnica avanzada.
- **Interpretabilidad**: Los modelos basados en redes neuronales pueden ser menos interpretables en comparación con enfoques más simples como la factorización de matrices.

d. **Filtrado colaborativo basado en sistemas híbridos**

Combina múltiples enfoques de recomendación para aprovechar las ventajas de cada uno y mitigar sus limitaciones. Por ejemplo, combinar filtrado colaborativo con filtrado basado en contenido o integrar diferentes técnicas dentro del filtrado colaborativo.

Métricas de similitud

Para evaluar la similitud entre usuarios o ítems, se emplean diversas métricas:

Similitud del coseno Mide el coseno del ángulo entre dos vectores de calificación.

$$\text{Sim}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$$

Correlación de Pearson Evalúa la correlación lineal entre las calificaciones de dos usuarios.

$$\text{Sim}(A, B) = \frac{\sum (A_i - \overline{A})(B_i - \overline{B})}{\sqrt{\sum (A_i - \overline{A})^2} \sqrt{\sum (B_i - \overline{B})^2}}$$

Distancia euclidiana Calcula la distancia directa entre dos vectores de calificación.

$$\text{Dist}(A, B) = \sqrt{\sum (A_i - B_i)^2}$$

Desafíos del filtrado colaborativo

A pesar de su efectividad, el filtrado colaborativo enfrenta varios desafíos técnicos:

I. **Escasez de datos (Sparsity)**
En sistemas con una gran cantidad de ítems y usuarios, la matriz de interacción suele ser muy dispersa, lo que dificulta la identificación de patrones de similitud.

Soluciones:

- Utilizar técnicas de factorización de matrices para inferir valores faltantes.
- Implementar métodos de imputación de datos.

II. **Arranque en frío (Cold Start)**
Dificulta la recomendación para nuevos usuarios o ítems que no tienen suficientes interacciones registradas.

Soluciones:

- Incorporar información adicional mediante filtrado basado en contenido.
- Emplear sistemas híbridos que combinen múltiples fuentes de información.

III. **Escalabilidad**
El aumento en la cantidad de usuarios e ítems puede impactar negativamente el rendimiento de los algoritmos basados en vecinos.

Soluciones:

- Optimizar cálculos mediante estructuras de datos eficientes.
- Utilizar algoritmos de aproximación y técnicas de reducción de dimensionalidad.

IV. **Sincronización de preferencias**
Las preferencias de los usuarios pueden cambiar con el tiempo, lo que requiere que los sistemas actualicen continuamente sus modelos.

Soluciones:

- Implementar modelos dinámicos que consideren la evolución temporal de las preferencias.
- Utilizar ventanas de tiempo para ponderar más las interacciones recientes.

3.1. Casos de uso en grandes plataformas.

El estado actual de la inteligencia artificial en el eCommerce muestra una evolución continua hacia sistemas más avanzados y personalizados. Las aplicaciones de IA en este ámbito, como los chatbots, el análisis de sentimientos y

los sistemas de recomendación, han optimizado significativamente la experiencia del usuario y los procesos de venta. Sin embargo, aún existen desafíos, como la escalabilidad, la escasez de datos y los sesgos éticos.

Las grandes plataformas, como Amazon y Netflix, han demostrado el éxito de la integración de enfoques híbridos, ofreciendo importantes lecciones sobre adaptabilidad e innovación para futuras investigaciones en este campo.

3.1.1. Amazon

Amazon utiliza una combinación avanzada de algoritmos de recomendación y análisis de datos para personalizar la experiencia de compra de sus usuarios. Entre sus tecnologías y características principales se destacan:

- **Sistemas de recomendación**: Mediante el uso de IA, Amazon sugiere productos basándose en el historial de compras y navegación del usuario. Las recomendaciones abarcan categorías como "productos relacionados", "los clientes que compraron esto también compraron" y "recomendaciones personalizadas".
- **Análisis predictivo**: Amazon predice la demanda de productos utilizando análisis predictivo para optimizar su inventario, lo que permite una gestión eficiente de stock y tiempos de entrega.
- **Chatbots y asistentes virtuales**: Alexa, el asistente de voz de Amazon, permite a los usuarios realizar compras mediante comandos de voz, mejorando la accesibilidad y facilidad de compra.
- **Optimización de precios**: Amazon ajusta sus precios en tiempo real a través de algoritmos de IA que maximizan la competitividad y las ventas, adaptándose a las fluctuaciones del mercado.

Gracias a estas tecnologías, Amazon ofrece una experiencia personalizada, permitiendo a los usuarios encontrar productos de interés con mayor rapidez, lo cual incrementa la probabilidad de compra y la satisfacción del cliente. Las estrategias de recomendación y optimización de precios han demostrado ser efectivas en el aumento de las ventas y en la retención de clientes.

3.1.2. Netflix

Netflix implementa sistemas de recomendación avanzados que identifican y sugieren contenido relevante para cada usuario. Este sistema se basa en:

- **Algoritmos de recomendación**: Netflix utiliza algoritmos de aprendizaje automático para sugerir películas y series, tomando en cuenta el historial de visualización y preferencias previas de cada usuario.
- **Perfilado de usuarios**: La plataforma construye perfiles detallados que permiten entender mejor los gustos de los usuarios y adaptar las recomendaciones en

consecuencia.

- **Pruebas A/B**: Netflix realiza constantes pruebas A/B para evaluar y optimizar tanto la interfaz de usuario como las recomendaciones, asegurando que la experiencia del usuario sea intuitiva y atractiva.

Las recomendaciones personalizadas de Netflix ayudan a retener a los usuarios en la plataforma, promoviendo el consumo de contenido y aumentando el tiempo de visualización. Este sistema mejora notablemente la satisfacción del usuario al proporcionar contenido relevante y atractivo.

3.1.3. Spotify

Spotify utiliza inteligencia artificial para generar recomendaciones musicales personalizadas, facilitando el descubrimiento de nueva música:

- **Algoritmos de recomendación**: Las playlists personalizadas como "Discover Weekly" y "Daily Mix" son producto de algoritmos que analizan el historial de escucha del usuario y adaptan las recomendaciones en función de sus preferencias (Spotify's Recommendation System).
- **Análisis de datos**: Spotify analiza grandes cantidades de datos de escucha para identificar patrones y tendencias musicales, mejorando la relevancia de sus recomendaciones .
- **Filtrado colaborativo**: Utilizando técnicas de filtrado colaborativo, Spotify recomienda canciones que otros usuarios con gustos similares han escuchado, fomentando la exploración musica.

Las recomendaciones personalizadas en Spotify mejoran la experiencia de los usuarios, quienes descubren música nueva acorde a sus gustos, lo cual incrementa su fidelidad a la plataforma. Este sistema también aumenta el tiempo de permanencia de los usuarios en la plataforma.

3.1.4. Alibaba

Alibaba emplea inteligencia artificial para optimizar la experiencia de compra y potenciar las ventas mediante:

- **Recomendaciones personalizadas**: Alibaba utiliza IA para recomendar productos a sus usuarios, basándose en el comportamiento de compra y navegación, lo que mejora la relevancia de las sugerencias.
- **Asistentes virtuales**: Tmall Genie, un asistente virtual de Alibaba, ayuda a los usuarios en sus compras, proporcionando una experiencia interactiva y eficiente.
- **Análisis de sentimiento**: Alibaba analiza reseñas y opiniones para comprender mejor la percepción de los usuarios sobre los productos, lo cual permite ajustes en la oferta y la mejora de servicios.

Las recomendaciones personalizadas de Alibaba han demostrado incrementar las ventas de la plataforma y mejorar la satisfacción del cliente. El uso de asistentes virtuales y análisis de sentimiento optimiza la calidad del servicio, ofreciendo una experiencia de compra más satisfactoria y enfocada en las necesidades de los clientes.

4. Tecnologías

La selección de las herramientas y las tecnologías se ha realizado tomando en cuenta criterios de rendimiento, facilidad de adopción y soporte activo por parte de la comunidad. A continuación se listan las principales:

4.1. Lenguaje de programación y framework

- **Python 3.x**: Se eligió Python por su extenso ecosistema de librerías científicas (`NumPy`, `Pandas`, etc.) y de aprendizaje automático (`scikit-learn`, `Surprise`, `LightFM`, `PyTorch`, entre otras), facilitando la experimentación con diversos algoritmos y métodos.
- **Django**: Se adoptó Django como framework de alto nivel para la parte web, por su clara separación en aplicaciones (*apps*), su potente ORM y la rapidez de desarrollo que proporciona. La integración con herramientas como `Django REST Framework` o vistas basadas en plantillas (`HTML`) resulta sencilla, según las necesidades.

4.2. Librerías de inteligencia artificial y ciencia de datos

- **Surprise (scikit-surprise)**: Proporciona implementaciones optimizadas de algoritmos de filtrado colaborativo (SVD, SVD++, KNN, etc.), con métricas embebidas de evaluación (RMSE, MAE) y utilidades para validación cruzada o *train/test split*.
- **Pandas, NumPy y SciPy**: Permiten manipular y analizar datos de reseñas, productos y usuarios, así como realizar operaciones matriciales (fundamentales en algoritmos de factorización). `Pandas` facilita la carga de archivos CSV, su preprocesamiento y la comunicación con Django o scripts específicos.
- **Matplotlib / Seaborn**: Permiten construir gráficos de métricas (por ejemplo, comparar el RMSE de distintos algoritmos), o mostrar histogramas de valoraciones para un análisis descriptivo.

4.3. Bases de datos y persistencia de información

- **Base de datos relacional (Postgre)**: PostgreSQL ofrece robustez y flexibilidad. Django facilita el acceso a través de su ORM.

– **Pickle para modelos**: Tras entrenar un algoritmo, se serializa (pickle) el objeto entrenado para reutilizarlo sin recalcular todo, lo cual reduce tiempos de arranque y facilita la integración con la interfaz web.

4.3.1. Interfaz Web / Dashboard

– **HTML + Bootstrap**: La plantilla base se construye en HTML y, para dar un estilo atractivo y responsive, se usa Bootstrap. Ello agiliza el desarrollo de formularios y paneles, permitiendo mostrar fácilmente los resultados de los algoritmos.

5. Planificación

En este capítulo se expone el **plan de desarrollo del proyecto**, centrado en definir las tareas a ejecutar, los hitos principales, los roles y recursos humanos involucrados, así como la distribución temporal (cronograma) y la estimación de costes. Además, se incluyen los planes de gestión complementarios (riesgos, calidad y comunicaciones), alineados con las buenas prácticas de la ingeniería del software.

5.1. Análisis temporal y costes de desarrollo

5.1.1. Actividades, tareas e hitos

La planificación se estructura en **cinco grandes fases**, cada una de ellas descompuesta en tareas específicas y con hitos relevantes identificados mediante entregables:

1. **Inicio del proyecto (Jefe de Proyecto)**
   - 1.a) **Conceptualización:** definición del caso de negocio, objetivos, alcance, actores y riesgos iniciales.
   - 1.b) **Acta de constitución:** documento formal que aprueba y da inicio al proyecto.
2. **Elaboración de planes (Analista y Jefe de Proyecto)**
   - 2.a) **EDT y diccionario:** desglose jerárquico del trabajo (*Estructura de Desglose de Trabajo*).
   - 2.b) **Cronograma:** secuencia temporal de las tareas, identificando el camino crítico.
   - 2.c) **Presupuesto:** estimación de costes según horas por rol y adquisiciones.
   - 2.d) **Plan de riesgos:** análisis y estrategias de mitigación de riesgos.
   - 2.e) **Plan de calidad:** métricas e indicadores de calidad del producto.
   - 2.f) **Plan de comunicaciones:** canales, frecuencia y actores involucrados.
   - 2.g) **Seguimiento y control:** entrega del primer *hito*: versión alfa de los planes revisada.
3. **Desarrollo de interfaz (Diseñador, Desarrollador y Tester)**
   - 3.a) **Diseño:** arquitectura visual, experiencia de usuario y navegación.
   - 3.b) **Desarrollo:** implementación de la interfaz con revisiones iterativas.

3.c) **Seguimiento:** entrega del *hito* alfa de interfaz.

3.d) **Pruebas:** validación funcional, generando la versión beta.

4. **Codificación (Diseñador, Analista, Desarrollador, Tester)**

4.a) **Diseño técnico:** definición de la arquitectura lógica del sistema.

4.b) **Algoritmo IA:** integración colaborativa de modelos como SVD, KNN, SVD++, etc.

4.c) **Implementación complementaria:** conexión entre lógica e interfaz, finalización de módulos.

4.d) **Seguimiento:** versión alfa de la lógica de negocio.

4.e) **Pruebas:** validación por el Tester, generando la versión beta.

5. **Cierre del proyecto (Jefe de Proyecto)**

5.a) **Memoria del TFG:** documentación final del desarrollo.

5.b) **Presentación:** elaboración de diapositivas para la defensa.

5.c) **Defensa del TFG:** *hito final* que marca la entrega oficial.

5.1.2. Estructura de Desglose de Trabajo (EDT)

La Figura 5.1 (pendiente de integrar) muestra la estructura jerárquica del proyecto, detallando fases, subfases y tareas. Cada entrada de la EDT incluye un diccionario con descripción, fechas y responsable. Un ejemplo correspondiente a la tarea de conceptualización del algoritmo es:

- **ID:** 4.1.2.1
- **Nombre:** Conceptualización del algoritmo IA
- **Descripción:** Definir la lógica de factorización matricial y los métodos KNN para el sistema de recomendación.
- **Responsable:** Analista
- **Fechas:** 01/03/2025 – 15/03/2025

Figura 5.1: Estructura de Desglose de Trabajo (EDT) del proyecto.

5.1.3. Cronograma

El cronograma refleja la secuencia y duración de cada tarea, así como sus responsables. (Pendiente de integrar)

5.1.4. Estimación de costes

La estimación de costes se ha basado en tarifas horarias promedio por rol.

Distribución por fases:

- Iniciación y planificación: $\sim$10 %
- Diseño y desarrollo de interfaz: $\sim$15 %
- Implementación IA y backend: $\sim$40 %
- Pruebas e integración: $\sim$20 %
- Documentación y cierre: $\sim$15 %

5.2. Planes de Gestión

5.2.1. Plan de riesgos

Riesgos identificados:

1. Retrasos por carga académica o disponibilidad del equipo.
2. Dificultades técnicas en la integración IA o frontend.
3. Modificaciones de alcance solicitadas por el tutor.

Estrategias de mitigación:

- Reajustar el cronograma con tareas en paralelo.
- Asesoramiento técnico externo (tutores o comunidad).
- Priorización funcional mediante backlog.

5.2.2. Plan de calidad

Indicadores principales:

1. Precisión del sistema de recomendación: RMSE < 1.2 o Precision > 0.3.
2. Evaluación subjetiva de la interfaz con 5–10 usuarios.
3. Validación de funcionalidades mediante checklist.

Acciones de mejora:

- Revisiones periódicas con el tutor.
- Retrospectivas al cierre de cada fase.
- Refactorización según incidencias detectadas.

5.2.3. Plan de comunicaciones

- Reuniones semanales con el tutor (1 hora).
- Reuniones quincenales con todo el equipo.
- Comunicación vía correo, Slack y GitHub (gestión de tareas e *issues*).
- Informes de seguimiento en cada hito.

5.2.4. Plan de recursos humanos y adquisiciones

Recursos humanos:

- **Jefe de Proyecto**: coordinación global, control de plazos y costes.
- **Analista**: diseño de algoritmos, planificación y validación técnica.
- **Diseñador**: desarrollo UI/UX y arquitectura visual.
- **Desarrollador**: implementación de backend y lógica de negocio.
- **Tester**: validación funcional y de experiencia de usuario.

Adquisiciones:

- Software: IDEs, licencias si aplican, librerías open source.
- Infraestructura: servidor de pruebas o VPS, coste estimado de 300 €/año.

5.3. Conclusiones

Este capítulo ha descrito en detalle la **planificación integral del proyecto**, definiendo tareas, cronograma, hitos, costes y planes de gestión. Esta estructura proporciona una visión clara del desarrollo, facilitando el seguimiento, el control y la toma de decisiones. Con ello, se sientan las bases para una ejecución eficiente y orientada al cumplimiento de los objetivos.

Se adjunta el diagrama de Gantt y el informe de recursos generado con Microsoft Project. Se destacan el camino crítico, dependencias y costes asignados por rol, considerando una disponibilidad estimada de 3 horas diarias de trabajo.

(pendiente de integrar)

6. Diseño de la solución

6.1. Introducción

En este capítulo se describe la arquitectura completa del *playground* de sistemas de recomendación, destacando los componentes clave y las razones que han guiado cada decisión de diseño. El principal objetivo es lograr un entorno de experimentación robusto y flexible que permita:

- Cargar y preprocesar datos masivos de interacción (*ratings*, reseñas o logs de uso).
- Entrenar distintos algoritmos de recomendación (SVD, KNN, SVD++, etc.).
- Comparar resultados con métricas de evaluación estándar, así como con indicadores más avanzados.
- Gestionar y visualizar las recomendaciones en una interfaz que permita la iteración con parámetros y experimentos.

El diseño se ha planteado de forma modular y escalable, para que las partes involucradas (capa de datos, capa de algoritmos y capa de presentación) puedan evolucionar y adaptarse a nuevos requisitos.

6.2. Arquitectura General

La solución propuesta presenta una arquitectura dividida en tres capas principales:

1. **Capa de Datos (Data Layer)**: Responsable de la carga, transformación y acceso a los datos requeridos por los algoritmos del sistema de recomendación.
2. **Capa de Lógica (Algorithm/Business Layer)**: Aloja las implementaciones de los diferentes modelos de recomendación, así como la lógica para su entrenamiento, evaluación y comparación.
3. **Capa de Presentación (Presentation Layer)**: Ofrece una interfaz de usuario que orquesta la interacción entre las capas anteriores y muestra resultados de forma comprensible.

Este patrón de *separación de capas* facilita la mantenibilidad y la extensibilidad: la capa de datos puede cambiar su fuente de información (por ejemplo, de CSV a una base de datos relacional o *data warehouse*) sin impactar directamente en la implementación de los algoritmos.

6.3. Diseño Detallado de los Módulos Principales

A continuación se describen en mayor profundidad los módulos que conforman cada capa, resaltando su responsabilidad dentro del sistema y cómo se comunican con el resto.

6.3.1. Módulo de Datos

Ingesta y Preprocesamiento La ingesta de datos se realiza a partir de un conjunto de ficheros CSV (*reviews_subset.csv*, *products_clean.csv*, etc.) resultantes de un proceso previo de filtrado y muestreo (ETL). Este módulo:

- *Lee* los CSV aplicando validaciones mínimas (comprobar columnas obligatorias: *user_id*, *asin*, *rating*, etc.).
- *Preprocesa* dichos datos para eliminar valores fuera de rango o duplicados.
- *Filtra* usuarios o productos con poca participación cuando sea necesario.
- *Exporta* un conjunto final de interacciones apto para su uso en los algoritmos.

Almacenamiento y Acceso Dependiendo de la envergadura del proyecto, se opta por:

- *Almacenamiento en memoria*: Adecuado para datasets moderados.
- *Almacenamiento en un SGBD (SQLite, PostgreSQL, etc.)*: Facilita manejo de datos de mayor volumen y persistencia más organizada.

En un prototipo o MVP, puede bastar con pandas y ficheros CSV, siempre y cuando exista un método estandarizado de acceder a las interacciones que sirva al módulo de algoritmos.

6.3.2. Módulo de Algoritmos y Evaluación

Dentro de la Capa Lógica, se distinguen dos submódulos importantes: *Módulo de Algoritmos de Recomendación* y *Módulo de Evaluación Métrica*.

Módulo de Algoritmos

- **Implementaciones de Filtrado Colaborativo**:
  - *KNN (Item-based/User-based)*: Basado en similitud (coseno, Pearson) entre ítems o usuarios.
  - *SVD y SVD++*: Factorización matricial que descompone la matriz de interacciones en factores latentes.

- **Interfaces de entrenamiento y predicción**: Cada algoritmo expone métodos para *fit* (entrenar con el conjunto de datos) y *predict* (estimar el rating o la preferencia de un usuario hacia un producto).
- **Capa de Pickle/Serialización**: Para acelerar la puesta en marcha y evitar recalcular todo en cada ejecución, se define un proceso de guardado y carga de modelos ya entrenados.

Módulo de Evaluación Este submódulo calcula métricas de rendimiento:

- *RMSE, MAE*: Orientadas a predecir con precisión las valoraciones (ratings).
- *Precision@k, Recall@k*: Focalizadas en la calidad de los top-N ítems recomendados.
- *Otras métricas*: Cobertura, novedad, diversidad o serendipia, dependiendo del alcance que se desee en la investigación.

Además, es posible almacenar los resultados de sucesivos experimentos, de manera que la plataforma facilite la comparación histórica de algoritmos y configuraciones.

6.3.3. Módulo de Presentación y *Dashboard*

La capa de presentación es la cara visible de la solución. Integra:

- **Panel de Control (Dashboard)**: Un formulario o conjunto de componentes donde el usuario elige el algoritmo (SVD, KNN, SVD++), parámetros (número de factores, k para KNN) y un subconjunto de datos o usuarios con los que experimentar.
- **Visualización de Recomendaciones y Métricas**: Cuando el usuario lanza un experimento, el sistema muestra las recomendaciones generadas (p.ej., top-5 productos) y las métricas asociadas (RMSE, *Precision@5*, etc.).
- **Comparación de Resultados**: Para ver de forma clara, por ejemplo, las diferencias de precisión entre un método y otro o la influencia de un determinado hiperparámetro.

El diseño de la interfaz prioriza la usabilidad y la claridad de la información, pues su fin principal es apoyar el proceso de experimentación y análisis, más que el de un *e-commerce* real.

1. *El* dashboard recibe la petición de un usuario, indicando el algoritmo y parámetros deseados.
2. *El módulo de algoritmos* (Capa Lógica) solicita los datos al *Dataset*, ya sea en memoria o consultando una base (Capa de Datos).
3. Se produce el entrenamiento o la carga de un modelo *pickle*.
4. Se generan las recomendaciones y se calculan métricas.
5. El *dashboard* muestra los resultados y gráficas comparativas.

Esta separación de responsabilidades posibilita que cada componente evolucione de manera independiente, añadiendo, por ejemplo, nuevos algoritmos, sin alterar la capa de datos o la interfaz.

6.4. Patrones y Razonamientos de Diseño

El desarrollo de la solución está inspirado en varios principios de ingeniería de software:

- **Patrón MVP (Model-View-Presenter)** o variante MVC: Aislar la lógica de negocio (algoritmos, entrenamiento) de la lógica de presentación (gráficos, *dashboard*).
- **capas de abstracción**: Las capas superiores se basan en interfaces, no en implementaciones concretas. Podríamos sustituir *SVD* por otra clase sin tocar la interfaz del *dashboard*.
- **Modularidad y extensibilidad**: Los algoritmos de recomendación se ubican en paquetes independientes, con una interfaz común (ej. `train()`, `predict()`, `evaluate()`) que facilita incorporar nuevos métodos.

La integración con librerías de aprendizaje automático (Scikit-Surprise, LightFM, etc.) se realiza mediante un *wrapper* que unifica la comunicación, evitando dependencias excesivas que dificulten sustituciones futuras.

6.5. Conclusiones

En este capítulo se ha presentado un diseño modular para el *playground* de sistemas de recomendación, donde cada capa (datos, algoritmos, presentación) se ocupa de una responsabilidad clara. Se ha enfatizado la importancia de:

1. **Separación de capas**: Garantiza mantenibilidad y escalabilidad.
2. **Flexibilidad de algoritmos**: Permite introducir nuevos métodos o librerías sin remodelar toda la solución.
3. **Facilidad de experimentación**: A través de un *dashboard* interactivo que muestre recomendaciones y métricas.

Este diseño sienta las bases para la posterior *implementación* y validación empírica de los algoritmos de recomendación. Gracias a la arquitectura planteada, el sistema puede crecer en complejidad —introduciendo más datos y técnicas— sin comprometer los principios de orden, modularidad y legibilidad de la solución. Además, ofrece un marco adecuado para medir el rendimiento de cada estrategia de recomendación, integrando las métricas más comunes y dando apoyo a la investigación y el desarrollo continuo en el área de la inteligencia artificial aplicada al *e-commerce*.

7. Implementación

7.1. Introducción

En este capítulo se describe la construcción práctica de la plataforma de sistemas de recomendación, detallando los pasos, librerías y estructura de archivos usados para pasar del diseño conceptual a un prototipo funcional. El objetivo es exponer cómo se han plasmado en código los requerimientos y las decisiones arquitectónicas del capítulo anterior, cubriendo tanto la parte de *backend* (lógica de negocio y algoritmos) como la de *frontend* (presentación e interacción con el usuario).

Este capítulo también se adentra en la descripción de cómo se gestionan las interacciones con la base de datos o los ficheros CSV, el modo en que se ejecutan los algoritmos de recomendación y el procedimiento para medir y mostrar las métricas de rendimiento. Dado que el proyecto pretende servir como entorno de experimentación, se hará énfasis en la modularidad del código y en los mecanismos para añadir o modificar algoritmos sin alterar la arquitectura base.

7.2. Implementación de la Solución

La estructura de implementación se ha basado en los requerimientos y la arquitectura diseñada en capítulos previos. Se distinguen varios componentes clave:

7.2.1. Estructura de carpetas (ejemplo)

```
tfg_ecommerce_rs/
 manage.py
 ecommerce_rs/            # Configuraciones globales de Django
    settings.py
    urls.py
    wsgi.py
 accounts/                # Gestión de usuarios
 products/                # Gestión de productos y categorías
 recommendations/         # Lógica de recomendación
    algorithms/          # Implementaciones (KNN, SVD, etc.)
    management/          # Commands para ETL, entrenamiento...
    models.py            # Modelos (Review, etc.)
    templates/           # Plantillas para dashboard
       recommendations/
```

```
        dashboard.html
    views.py            # Vistas o endpoints
    urls.py
    ...
data/                    # Archivos CSV de reseñas, productos, etc.
```

7.2.2. Capa de datos e interacciones

Para el MVP, la carga de datos se realiza de la siguiente forma:

1. *Preprocesando los CSV* mediante scripts de ETL, que filtran y muestrean las reseñas, generando un archivo `.csv`.
2. *Importando* dicho *subset* en modelos de Djangopara mostrar al usuario los productos y utilizándolos también directamente en formato `pandas` junto con `Surprise`.

Además, se implementa en `recommendations/models.py` un modelo `Review` que enlaza `User` con `Product`, guardando la puntuación (`rating`) y una fecha.

7.2.3. Entrenamiento de Algoritmos de Recomendación

En la carpeta `recommendations/algorithms/` se ubican las distintas clases o métodos responsables de entrenar y predecir. Sin embargo, para grandes volúmenes de datos, a menudo se crea un *management command* (`train_models.py`) que:

– **Lee** el CSV final (*reviews_subset.csv*).

– **Crea** un objeto *Dataset* de Surprise.

– **Divide** en train/test.

– **Entrena** SVD, KNN, SVD++ (u otros métodos).

– **Evalúa** con métricas (RMSE, MAE, etc.).

– **Guarda** cada modelo con `pickle`.

Una vez generados `model_svd.pkl`, `model_knn.pkl`, etc., no se necesita recalcular todo en cada petición web.

7.2.4. Módulo de Evaluación de Métricas

En la misma carpeta o en un módulo aparte, se incluye la lógica para:

– Calcular **RMSE, MAE** para ratings.

– Calcular **Precision k, Recall k** para recomendaciones top-N.

– Registrar resultados en logs o archivos `CSV` para comparativas sucesivas.

Surprise ya contiene métodos para RMSE y MAE. Para *Precision k* se puede implementar un método que compare los items predichos en el top-k del usuario contra sus items relevantes en el test set.

7.2.5. Capa de Presentación y *Dashboard*

Las vistas definidas en `recommendations/views.py` permiten al usuario:

1. **Seleccionar** un `user_id` y un algoritmo concreto (SVD, KNN, SVD++, etc.).
2. **Cargar** el modelo pickeado desde disco.
3. **Generar** y mostrar el top-N de productos recomendados.
4. **Visualizar** métricas de rendimiento o gráficas comparativas en `dashboard.html`.

Este *dashboard* se apoya en plantillas Django, con formularios para recoger los parámetros y secciones para mostrar los resultados y gráficas.

Interfaz de Usuario

- **Formulario de selección**: Contiene desplegables (*select*) con la lista de usuarios y algoritmos disponibles, así como campos extra (número de recomendaciones a mostrar, por ejemplo).
- **Zona de resultados**: Un panel donde se enumeran los items recomendados y su puntuación estimada.
- **Zona de métricas**: Muestra tablas o gráficos con valores de RMSE o precisión, permitiendo comparar varios modelos.

7.2.6. Integración y Pruebas de Conjunto

Para asegurar la cohesión de todo el sistema, se realizan pruebas integrales (mockeo de datos, scripts de validación, etc.) donde se cubre:

- **Prueba de carga de datos**: Verificar si, con un CSV mayor a 100k filas, el sistema reacciona bien y filtra sin bloqueos.
- **Prueba de entrenamiento y evaluación**: Confirmar que tras ejecutar `train_models.py` se generen archivos `.pkl` y se impriman métricas adecuadas.
- **Prueba de recomendaciones en web**: Desde el *dashboard*, seleccionar un usuario y un algoritmo, ver el top-N y las puntuaciones.
- **Prueba de escalabilidad sencilla**: Incrementar gradualmente el número de interacciones y observar si los tiempos de respuesta continúan siendo razonables.

7.3. Conclusiones

En este capítulo se ha descrito la **implementación** práctica del prototipo de la aplicación web de recomendación, desde la *capa de datos* (con scripts de filtrado y carga), pasando por la *capa de algoritmos* (entrenamiento con Surprise y almacenamiento de modelos), hasta la *capa de presentación* (donde un *dashboard* facilita la configuración y consulta de recomendaciones).

El uso de **Django** como *framework* web agiliza la creación de vistas y la manipulación de datos, mientras que librerías como **Surprise** brindan una forma rápida y optimizada de implementar distintos algoritmos de filtrado colaborativo y compararlos con métricas estándar. Con esta base, el *MVP* es capaz de:

- Procesar y filtrar un gran volumen de reseñas de usuarios para su uso en algoritmos de recomendación.
- Entrenar varios modelos (SVD, KNN, SVD++), medir su rendimiento y guardar los resultados.
- Exponer un entorno visual que facilita la selección de usuarios, algoritmos y parámetros, junto a la visualización de las recomendaciones y sus métricas de fiabilidad.

Así, el sistema resultante cumple los requisitos de servir como banco de pruebas para la investigación y desarrollo de nuevos modelos de recomendación, ofreciendo la modularidad suficiente para incorporar librerías o técnicas más avanzadas (por ejemplo, *deep learning* o sistemas híbridos) con escasas modificaciones en su arquitectura. Sentando las bases para la validación que se presentará en los capítulos siguientes, a fin de contrastar el desempeño real de los algoritmos en el conjunto de datos escogido.

8. Cierre

8.1. Introducción

En este capítulo explicaremos...

8.2. Conclusiones

En este capítulo concluimos que...

Bibliografía