MÁSTER EN BIG DATA Y ANALÍTICA AVANZADA DE DATOS

TRABAJO FIN DE MÁSTER

SISTEMA DE RECOMENDACIÓN DE PRODUCTOS BANCARIOS UTILIZANDO TÉCNICAS DE MACHINE LEARNING Y DEEP LEARNING

Autor: Valeria Capurro Behr

Director: Elena Igualada Villodre

Madrid

Junio 2023

SISTEMA DE RECOMENDACIÓN DE PRODUCTOS BANCARIOS UTILIZANDO TÉCNICAS DE MACHINE LEARNING Y DEEP LEARNING

Autor: Capurro Behr, Valeria.
Director: Igualada Villodre, Elena.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN

En este proyecto, se aborda el desafío de construir un modelo recomendador de productos bancarios utilizando técnicas de Machine Learning y Deep Learning. El objetivo principal es proporcionar recomendaciones personalizadas y relevantes a los clientes, teniendo en cuenta su perfil y necesidades individuales. Para ello, se recopilan y analizan datos relevantes como historiales de transacciones y perfiles de clientes. Se exploran diferentes algoritmos y técnicas de recomendación, incluyendo en primer lugar un modelo híbrido que modela la tenencia de productos independientemente a través de modelos XGBoost de predicción binaria, y en segundo lugar un filtrado colaborativo basado en modelos utilizando Keras y redes neuronales: Generalized Matrix Factorization (GMF) y Neural Matrix Factorization (NMF).

Se evalúa el rendimiento del recomendador mediante el MapK@3, MapK@5, MapK@7 y MapK@9. En el proceso de desarrollo del modelo hibrido se tuvo la limitación de falta de datos en muchos productos con tasa de adquisiciones bastante bajas, por lo cual se limitó la recomendación basada en este método a tan solo productos con datos suficientes. Los resultados de los modelos construidos se comparan contra una recomendación base de recomendar los K productos más populares a todos los usuarios. A pesar de que los modelos binarios XGBoost que forman parte del ensamble del modelo híbrido obtienen resultados favorables, el ensamble tan solo es capaz de mejorar ligeramente la recomendación base. Por otro lado, al analizar las predicciones de las interacciones segregadas por productos, los modelos GMF y NMF obtienen buenos resultados de recall pero malos de precisión, es decir, el coste de reducir los falsos negativos, fue obtener una tasa elevada de falsos positivos. Sin embargo, logran mejorar los valores de MapK del modelo base en mayor medida que el modelo hibrido. En conclusión, la recomendacion de productos utilizando técnicas de Machine Learning y Deep Learning puede ser estratégico para las entidades financieras al mejorar la satisfacción del cliente. Se enfatiza la importancia de la recopilación de datos relevantes, la exploración de diferentes algoritmos y la selección de modelos eficientes.

Palabras clave: Recomendador de productos, Machine Learning, Deep Learning, Filtrado Colaborativo, Redes Neuronales

BANKING PRODUCT RECOMMENDATION SYSTEM USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES

Author: Capurro Behr, Valeria.
Supervisor: Igualada Villodre, Elena.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

In this project, we address the challenge of building a recommender system for banking products using Machine Learning and Deep Learning techniques. The main objective is to provide personalized and relevant recommendations to customers, taking into account their individual profiles and needs. To achieve this, relevant data such as transaction histories and customer profiles are collected and analyzed. Different recommendation algorithms and techniques are explored, including a hybrid model that models product holdings for each product independently using XGBoost binary prediction models, and a collaborative filtering models using Keras and neural networks: Generalized Matrix Factorization (GMF) and Neural Matrix Factorization (NMF). The performance of the recommender is evaluated using MapK@3, MapK@5, MapK@7, and MapK@9.

During the development process of the hybrid model, there was a limitation of insufficient data for many products with low acquisition rates. Therefore, the recommendation based on this method was limited to only products with sufficient data. The results of the constructed models are compared against a baseline recommendation: recommending the K most popular products to all users. Although the binary XGBoost models that are part of the ensemble of the hybrid model obtained favorable results, the ensemble or recommendation system can only slightly improve upon the baseline recommendation. On the other hand, when analyzing the predictions of the interactions segregated by products, the GMF and NMF models achieve good recall but poor precision, meaning that they minimized the amount of false negative at the cost of having a high rate of false positives. However, they managed to improve the MapK values of the baseline model to a greater extent than the hybrid model.

In conclusion, the development of a recommender system for banking products using Machine Learning and Deep Learning techniques can be strategic for financial institutions by improving customer satisfaction. The importance of collecting relevant data, exploring different algorithms, and selecting efficient models is emphasized.

Keywords: Recommender System, Machine Learning, Deep Learning, Collaborative Filtering, Banking entities, Neural Networks.

ÍNDICE DE LA MEMORIA

ÍNDICE DE TABLAS

INDICE DE FIGURAS

Capítulo 1. Introducción

1.1 Motivación

Los productos bancarios que ofrecen las distintas instituciones financieras a sus clientes suelen ser servicios o productos estándares como cuentas, tarjetas, préstamos, hipotecas, entre otros. Sin embargo, bajo el contexto de un mercado altamente competitivo caracterizado por el surgimiento de bancos digitales y la oferta creciente de productos y servicios innovadores, los bancos tradicionales que antes limitaban su cartera de productos se enfrentan a la necesidad de desarrollar nuevas estrategias para mantenerse competitivos, captar nuevos clientes y fidelizar a los actuales.

Cada producto que ofrece la entidad bancaria al tener diferentes características, condiciones y beneficios presenta información latente que hace que este se adecúe a las necesidades de los distintos perfiles de los usuarios; y a la vez cada usuario es caracterizado por un conjunto de datos personales, financieros y de comportamiento que lo hacen único y similar a sus adyacentes a la vez. Es por ello por lo que, para que los bancos utilicen sus recursos de una manera más eficiente, puedan identificar qué productos son más adecuados para cuáles usuarios con la finalidad de recomendarlos.

La construcción de un modelo recomendador de productos bancarios se convierte en una iniciativa estratégica que puede impulsar la satisfacción del cliente y fortalecer las relaciones con la entidad bancaria. Además, este proyecto busca demostrar cómo se pueden explotar los datos que las entidades bancarias ya tienen aprovechando los avances en el análisis de datos y las tecnologías Big Data para explorar y comparar distintos mecanismos de recomendación que finalmente resulten en un beneficio económico para la entidad. Al proporcionar recomendaciones personalizadas y relevantes a los clientes, este modelo recomendador ayudará a los bancos a ofrecer una experiencia mejorada, diferenciarse en el mercado y mantenerse competitivos en un entorno de constante evolución.

1.2 Objetivos

El objetivo principal del trabajo es, mediante el estudio y utilización de técnicas de Machine Learning y Deep Learning construir un modelo que en base a información sobre la tenencia de productos y sobre características de usuarios recomiende productos a clientes.

Otros objetivos que se tendrán en cuenta son:

- Recopilar y analizar los datos relevantes, como historiales de transacciones, perfiles de clientes y características de productos bancarios, para entrenar y ajustar el modelo recomendador.
- Comprender el ciclo de vida de un caso de uso de ciencia de datos en una empresa dedicada al sector banca, desde la etapa de planteamiento de la problemática, identificación de datos disponibles, planteamiento de población base y target, ingeniería de variables modelado y medición.
- Explorar diferentes algoritmos y técnicas de recomendación utilizados en el ámbito de la analítica de datos y el aprendizaje automático, y evaluar su aplicabilidad y eficacia en el contexto de productos bancarios.
- Desarrollar y validar un modelo recomendador de productos bancarios que sea capaz de proporcionar recomendaciones personalizadas y relevantes a los clientes, teniendo en cuenta su perfil, necesidades y preferencias individuales.
- Realizar comparaciones y evaluaciones de los distintos mecanismos de recomendación utilizados, considerando métricas como la precisión de K recomendaciones, con el objetivo de identificar el enfoque más efectivo y eficiente.
- Analizar el impacto de las recomendaciones en la satisfacción del cliente y la rentabilidad del banco, y proponer mejoras o acciones para optimizar el modelo.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estudio del estado del arte.** En este capítulo se introducirán las técnicas y las herramientas que existen y se han utilizado durante el estudio, así como las que podrían utilizarse para la construcción de un modelo recomendador de productos.
- **Datos.** En este capitulo se realizará un análisis inicial del conjunto de datos disponible para el proyecto.
- **Diseño y Desarrollo**. En este capítulo se expondrán las decisiones de diseño que se han ido tomando respecto a la utilización de las técnicas y herramientas explicadas previamente, las características extraídas y clasificadores utilizados para el aprendizaje. Asimismo, se explicará con detalle el flujo de trabajo que se ha seguido para desarrollar el estudio.

- **Pruebas y Resultados**. En este apartado se expondrán los resultados obtenidos al aplicar las diferentes técnicas de detección de recomendación de productos y se hará una comparativa entre ellas.
- **Conclusiones, Limitaciones, Recomendaciones y Trabajo Futuro**. Por último, a partir de los resultados obtenidos se desarrollarán unas conclusiones y se expondrán posibles mejoras e ideas para trabajos futuros

Capítulo 2. Estado del Arte

2.1 Introducción

En el siguiente capítulo se presentará una visión general de los avances más recientes y relevantes en el campo de estudio de los sistemas de recomendación. Este capítulo tiene como objetivo contextualizar el proyecto dentro del contexto más amplio de investigación y desarrollo en sistemas de recomendación. En esta sección, se exploran y analizan las investigaciones, métodos y enfoques más destacados que han surgido en el área de los sistemas de recomendación. Se revisan y sintetizan las principales contribuciones de la literatura científica, los avances tecnológicos y las aplicaciones prácticas en este campo.

Se aborda aspectos clave, como los algoritmos de recomendación más utilizados, las técnicas de aprendizaje automático y aprendizaje profundo empleadas, así como las métricas de evaluación comunes utilizadas para medir la efectividad de los sistemas de recomendación. Además, se identifican los desafíos y problemas actuales que aún están presentes en este campo, como el problema del "*cold start*", la escalabilidad de los algoritmos y la diversidad de recomendaciones.

2.2 Técnicas

2.2.1 Recomendaciones basadas en popularidad (*popular-based*)

Consiste en recomendar las k opciones más populares. Es la estrategia más simple, pero se suele utilizar como *baseline* para medir el rendimiento del recomendador. La principal limitación de esta estrategia es que todos los usuarios reciben la misma recomendación, es decir estas no son personalizadas. Sin embargo, muchas empresas aún mantienen estas recomendaciones ya que funcionan bien para recomendar contenido o productos a usuarios nuevos, de los cuales todavía no se ha podido recopilar suficiente informacion como para hacer una recomendación personalizada. Ejemplos de esta estrategia son las películas mejor rankeadas de IMBD, o el top 10 de series o películas de Netlfix en tu país.

2.2.2 Recomendaciones basadas en contenidos (content-based)

Como menciona el autor Pathairush Seeda en su artículo "*A Complete Guide To Recommender Systems*" publicado la página web Medium (2021), las recomendaciones basadas en contenidos son una estrategia que consiste en mejorar la recomendación *baseline popular-based* agregando informacion característica de los ítems a recomendar con la finalidad de recomendar los ítems que

le son más relevantes al usuario. Un ejemplo claro de este tipo de recomendaciones es cuando Netflix te recomienda una lista de series o películas porque viste una película o serie en específico. Por lo tanto, esta serie de ítems podrían serle relevante al usuario por su similaridad en aspectos como género, subgénero, país, etc., a un ítem que el usuario ha demostrado gustarle.

Asimismo, Seeda (2021) comenta que la limitación principal de esta estrategia es que la recomendación se limita a lo que el usuario ha interactuado en el pasado. No da oportunidad al usuario a explorar nuevas áreas con las que no ha interactuado antes, sino más bien a explotar lo que el algoritmo ya sabe que le gusta al usuario. Además, no son por completo personalizadas, ya que todos los usuarios que interactúan con un ítem en particular recibirán la misma serie de recomendaciones, ya que no toma en cuenta las características de los usuarios sino de los ítems.

Por otro lado, el autor, Baptiste Rocca en su articulo "*Introduction to Recommender Systems*" publicado en la página web Medium (2021) indica que, a diferencia de los métodos colaborativos que se centran exclusivamente en las interacciones entre usuarios y elementos, los enfoques basados en contenido aprovechan información adicional sobre los usuarios y/o elementos. La premisa de los métodos basados en contenido es desarrollar un modelo que utilice las características disponibles para explicar las interacciones observadas entre usuarios y elementos. Estos métodos basados en contenido enfrentan menos el problema "*cold-start*" en comparación con los enfoques colaborativos, ya que los nuevos usuarios o elementos pueden ser descritos por sus características y, por lo tanto, se pueden realizar recomendaciones relevantes para estas entidades nuevas. Solo los usuarios o elementos nuevos con características previamente no vistas sufrirán esta limitación, pero a medida que el sistema adquiera más antigüedad, las posibilidades de encontrarse con esta situación serán escasas o nulas.

2.2.3 Filtrado colaborativo (*collaborative filtering*)

Seeda (2021) menciona que esta técnica de recomendación utiliza lo que se conoce como la matriz de interacción usuario-ítem para proporcionar recomendaciones personalizadas. En la matriz de interacción (Figura 1) cada celda representa la interacción entre un usuario y un ítem y se puede expresar de manera explícita o implícita. Por lo tanto, la cantidad de filas representa la cantidad de usuarios y la cantidad de columnas la cantidad de ítems.

Figura 1. Ejemplo de Matriz de Interacción

User-Item Interaction Matrix

User / Item	Item 1	Item ...	Item m
User 1	3		1
User ...	2		
User n		5	

El *rating* o *feedback* explicito se refiera a informacion que el usuario directamente ha proporcionado debido a que explícitamente se le ha preguntado.

Un ejemplo propuesto por Seeda en su artículo es,según lo que vemos en la Figura 1, considerando una escala del 1 al 5, al Usuario 1 le gusta el ítem 1 con un *score* de 3. Por otro lado, el *rating* o *feedback* implícito se refiere a informacion que indirectamente se deriva del comportamiento del usuario. Por ejemplo, podrían ser los minutos que un usuario ha visto un contenido, *clicks* que ha dado a una página, *likes* que ha dado a un video, o incluso directamente la tenencia de un producto en un momento determinado del tiempo (0 y 1). Existen muchas variaciones en la manera en que se puede estructurar esta matriz y depende mucho del objetivo de cada negocio y de la información que tiene disponible.

Finalmente, Seeda (2021) comenta que, el principal desafío en el diseño de métodos de filtrado colaborativo es que las matrices de interacción usuario-ítem son altamente dispersas (*sparse matrix*). Esto significa que la mayoría de las interacciones no existen, ya que los usuarios solo han interactuado con una pequeña fracción de las numerosas opciones disponibles.

La idea detrás de los métodos de filtrado colaborativo es que estas calificaciones no especificadas pueden ser estimadas porque las calificaciones observadas suelen estar altamente correlacionadas entre usuarios y elementos.

Dentro de la técnica de filtrado colaborativo podemos identificar 2 formas de realizarse:

a. **Filtrado Colaborativo basado en usuarios (*user-based collaborative filtering*):** Este método se centra en encontrar usuarios similares a través de patrones de comportamiento y preferencias compartidas. El proceso implica comparar las calificaciones o preferencias de un usuario objetivo con las de otros usuarios en la matriz. Se buscan usuarios con un historial similar de calificaciones y preferencias, y luego se utilizan sus patrones de comportamiento para hacer recomendaciones al usuario objetivo. Una gran limitación de esta técnica es el problema "*cold start*". Debido a que se basa en encontrar usuarios similares para realizar recomendaciones, se necesita información sobre las preferencias y calificaciones de un usuario para encontrar usuarios con gustos similares, y cuando un nuevo usuario se une al sistema y aún no ha proporcionado suficientes calificaciones, no hay suficiente información para encontrar usuarios similares, y esto dificulta la generación de recomendaciones precisas.

b. **Filtrado Colaborativo basado en ítems (*item-based collaborative filtering*):** En vez de buscar usuarios similares, este método busca ítems similares a los ítems con los que el usuario ya ha interactuado antes, en función de cómo han sido calificados o utilizados por los usuarios en el pasado. Si un usuario ha calificado positivamente un conjunto de elementos, el sistema de recomendación buscará elementos similares a los que le gustaron y se los recomendará. Este método es similar al filtrado basado en contenido, la única diferencia es la manera de encontrar ítems similares, en vez de utilizar características de estos, utilizamos las interacciones de la matriz. De igual manera, este método también tiene el problema del "*cold start*" cuando se trata de nuevos elementos en el sistema. Dado que se basa en encontrar elementos similares para realizar recomendaciones, se requiere información sobre las calificaciones y preferencias que le han dado los usuarios para identificar elementos similares. Cuando un nuevo elemento se introduce en el sistema y aún no ha sido calificado o no hay suficiente información disponible, se dificulta encontrar elementos similares y generar recomendaciones precisas basadas en él.

c. **Híbridos:** Existen métodos de filtrado colaborativo que, además, se apalancan en las características de los usuarios y los ítems para paliar el problema del '*cold-start'*

Por otro lado, Rocca (2019) aclara que una ventaja clave de los enfoques colaborativos es que no necesitan información específica sobre usuarios o elementos, lo que los hace aplicables en diversas

situaciones. Además, a medida que los usuarios interactúan más con los elementos, las recomendaciones se vuelven más precisas. Con el tiempo, las nuevas interacciones registradas proporcionan información adicional, lo que hace que el sistema sea cada vez más efectivo para un conjunto determinado de usuarios y elementos.

Finalmente, la manera de abordar el problema *cold-start*, indica Rocca (2019), puede realizarse siguiendo distintas estrategias: recomendar elementos al azar a nuevos usuarios o nuevos elementos a usuarios seleccionados al azar (estrategia aleatoria), recomendar elementos populares a nuevos usuarios o nuevos elementos a usuarios más activos (estrategia de máxima expectativa), recomendar un conjunto diverso de elementos a nuevos usuarios o un nuevo elemento a un conjunto variado de usuarios (estrategia exploratoria), o, en última instancia, utilizar un enfoque no colaborativo durante la fase inicial del usuario o del elemento.

2.3 Implementación

2.3.1 Recomendaciones basadas en popularidad (*popular-based*):

Se suele buscar el ítem con el mayor rating promedio entre todos los usuarios, el ítem con el conteo de votaciones más alto o el ítem con la mayor cantidad de interacciones. Asimismo, se suelen agregar características como tiempo o localización para hacer las recomendaciones más relevantes hacia los usuarios; por ejemplo, Netflix agrega el tiempo (hoy) y la localización (tu país) para diferenciar estas top 10 ítems basados en popularidad.

2.3.2 Recomendaciones basadas en contenidos (*content-based*):

Se podría tomar por ejemplo en el contexto de recomendación de películas a usuarios de Netflix el género de esta como acción, comedia o drama para enriquecer la recomendación.

2.3.3 Filtrado colaborativo (*collaborative filtering*):

Seeda (2021) indica que, en el filtrado colaborativo, hay dos tipos de métodos comúnmente utilizados: los métodos basados en memoria y los métodos basados en modelos. Los métodos basados en memoria utilizan directamente las calificaciones de los usuarios y su similitud para hacer recomendaciones, mientras que los métodos basados en modelos construyen un modelo estadístico o algoritmo que se ajusta a los datos de calificación para hacer predicciones.

a. Método basado en memoria (memory-based approach):

Este enfoque utiliza la matriz de interacción usuario-ítem directamente para calcular la matriz de similitud. Algunas medidas de similitud que suelen utilizarse son:

- Correlación de Pearson. Medida estadística que evalúa la relación lineal entre dos variables. Se usa para medir la similitud entre dos usuarios o elementos según la dirección de cómo cambian sus valores.
- Distancia euclidiana. Medida geométrica que calcula la distancia entre dos puntos en un espacio euclidiano. Es una medida de proximidad o similitud que se utiliza en el contexto del filtrado basado en contenido.
- Similitud coseno. Mide el ángulo del coseno de los dos vectores en el espacio multidimensional. A diferencia de la correlación de Pearson se basa en el ángulo formado entre los vectores, en lugar de las magnitudes de los vectores, y mide la similitud direccional entre dos vectores.

Según Seeda (2021) hay dos tipos de implementaciones que se pueden abordar dentro de los métodos basados en memoria:

Implementación basada en usuarios. Se encuentra un grupo de usuarios similares utilizando la medida de similitud elegida, se promedia el rating de cada ítem en el grupo de usuarios similares, y se ordenan descendientemente los ítems según el rating promedio, para finalmente recomendar desde esta lista ordenada ítems nuevos a usuarios target.

Implementación basada en ítems. Se encuentra un grupo de ítems similares utilizando la medida de similitud elegida y se seleccionan los top-k ítems más similares para recomendar.

Podemos ver que en este enfoque no hay necesidad de determinar parámetros como en otros modelos de Machine Learning, sino que el motor de recomendación logra acordarse de que le gusta y que no a los usuarios, para extraer similitud de esas interacciones; es decir, en este enfoque no existe la necesidad de inferir (predecir) nada.

Finalmente, Seeda (2021) aclara que la gran limitación que tiene este enfoque es la escalabilidad. Como se basa en memorizar cada interacción usuario ítem, y trabaja con matrices sparse (dispersas) tendrá una baja capacidad de generalización. Por otro lado, tiene el problema de que cada vez que se quiera agregar un nuevo usuario y/o ítem para incluirlo en la recomendacion, es necesario volver a ajustar el modelo.

b. Método basado en modelos (*model-based approach*):

El autor Seeda (2021) en su artículo menciona que, para evitar las limitaciones anteriores, en lugar del cálculo directo con la matriz de interacción usuario-ítem, descompondremos la matriz en matrices de menor dimensión que representen los factores latentes en dichas interacciones. Estos vectores reciben también el nombre de *embeddings* (especialmente en el contexto del Deep-learning).

La idea de la descomposición es que creemos que la matriz observada de interacciones usuario-ítem se construye a partir de la matriz subyacente de factores latentes de usuario e ítems; y que podemos extraer la mejor matriz subyacente de factores latentes que minimice la pérdida entre la matriz reconstruida y la matriz original para luego, utilizar el producto interno de las matrices de factores latentes de usuario e elemento para inferir interacciones no observadas.
En comparación con los métodos basados en memoria, los métodos basados en modelos extraen los datos de la matriz de interacción usuario-item y los utiliza como entrada a un modelo para hacer recomendaciones. Logra hacer predicciones y recomendaciones para usuarios y elementos sin necesidad de realizar cálculos en tiempo real con la matriz de interacción completa. Esto resuelve el problema de escalabilidad del enfoque basado en memoria y, por lo tanto, facilita su implementación en el mundo real (Seeda, 2021).

El autor Seeda (2021) presenta algunas de las técnicas existentes para la descomposición matricial en su artículo:

Truncated SVD (Sklearn): TruncatedSVD es una variante de la descomposición de valores singulares que trunca la matriz a las K principales componentes (n_componentes). Además, aplica reducción de dimensionalidad lineal y funciona bien con matrices 'sparse'.

Funk Matrix Factorization (Surprise). Librería de sistemas de recomendación que implementa varios algoritmos de predicción, 'baselines', métodos de vecinos más próximos, métodos basados en factorización de matrices (SVD, PMF, SVD++, NMF) y muchos otros. Además, incorpora varias medidas de similitud (coseno, MSD, Pearson, etc.) y proporciona herramientas para evaluar, analizar y comparar diferentes algoritmos.

LightFM. Como se ha mencionado anteriormente, los métodos de filtrado colaborativo basados en la factorización matricial presentan el problema de cold-start, en donde se les dificulta hacer

recomendaciones para ítems y usuarios nuevos ya que hay pocos o ningún dato de interacción disponible. Sin embargo, esto puede solucionarse con un enfoque hibrido.

LightFM es una biblioteca de Python que proporciona una implementación rápida y escalable de algoritmos basados en factorización de matrices para sistemas de recomendación híbridos. Es conocido por su velocidad, capacidad de escalabilidad y capacidad para combinar múltiples fuentes de información como calificaciones de usuarios, características de elementos y características de usuario en el proceso de recomendación.

Generalized Matrix Factorization (GMF) (Keras). Método de recomendación basado en factorización de matrices que utiliza redes neuronales para capturar las interacciones no lineales entre usuarios y elementos.

En GMF, se factoriza la matriz de interacción usuario-elemento en dos matrices de factores latentes: una matriz de factores latentes de usuarios y una matriz de factores latentes de elementos. Estas matrices representan características latentes que capturan las preferencias y las características de los usuarios y los elementos respectivamente.

La arquitectura GMF en Keras combina la información de estos dos conjuntos de factores latentes mediante una operación de multiplicación punto a punto, lo cual permite que la red neuronal aprenda las interacciones no lineales entre los factores latentes de los usuarios y los elementos y se utilice para realizar predicciones de ratings/interacciones y generar recomendaciones de nuevos pares usuarios-ítems

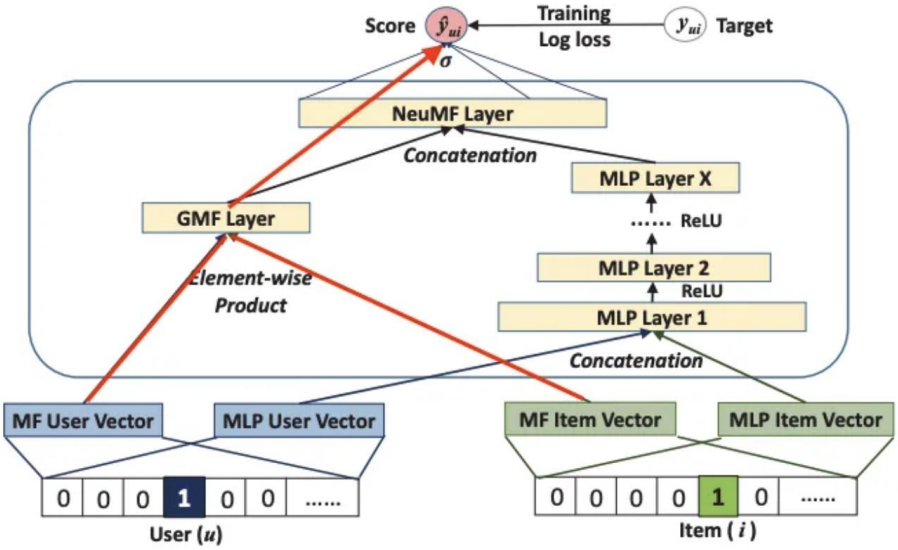
Neural Collaborative Filtering (NFC): Método de recomendación que utiliza redes neuronales para capturar las interacciones no lineales entre usuarios y elementos. A diferencia de los enfoques tradicionales de filtrado colaborativo basados en factorización de matrices, NCF combina tanto la información de factorización de matrices como la información de redes neuronales en una sola arquitectura.

Como indican He et al (2017) en el artículo *Neural Collaborative Filtering* (2017) la arquitectura NCF se compone de dos componentes principales: el componente de factorización de matrices y el componente de redes neuronales. El componente de factorización de matrices utiliza técnicas como la factorización matricial para capturar las relaciones lineales entre los usuarios y los elementos, mientras tanto, el componente de redes neuronales se encarga de aprender las interacciones no

lineales, con la finalidad de realizar predicciones de interacciones o generar recomendaciones personalizadas para nuevos pares usuario-ítem.

En Figura 2 podemos ver representadas tanto la arquitectura de GMF (línea roja) como la de NCF la cual adiciona el componente MLP. Mientras que GMF se centra en la factorización de matrices y la captura de relaciones lineales, NCF utiliza redes neuronales para aprender representaciones latentes más complejas y capturar relaciones no lineales entre usuarios y elementos. Esto permite que la arquitectura NCF tenga una mayor capacidad de modelado y potencialmente pueda mejorar la precisión de las recomendaciones al capturar relaciones más sofisticadas entre los usuarios y los elementos. Asimismo, los autores mencionan que, tras realizar varias pruebas, los mejores resultados se obtienen si es que a la MLP se le da una estructura de torre, es decir, la cantidad de neuronas en las capas ocultas va disminuyendo a medida que se avanza a través de la red.

Figura 2. Arquitectura Factorización Matricial Neuronal



Los autores He et al (2017) indican que una opción al diseñar este modelo es hacer que el componente GMF y MLP compartan la capa de embedding, fijando el mismo tamaño de embedding para ambos; esta decisión de diseño puede llegar a limitar el rendimiento del modelo, evitando encontrar el ensamble optimo. Mientras que, por otro lado, los autores proponen que cada componente tenga su propio embedding y que aprenda cuál es el tamaño óptimo que este debe tener, de manera que se concatenen en la última capa oculta.

Finalmente, los autores He et al (2017) indican que la inicialización del entrenamiento es un crucial determinante para obtener resultados óptimos. Como NMF es un modelo de ensamble, proponen entrenar previamente y por separado los componentes de GMF y MLP, guardar sus parámetros y finalmente concatenar sus salidas.

2.3.4 Diferencias de sesgo y varianza

El autor Rocca en su artículo (2019) explora la influencia que el nivel de modelado tiene sobre el sesgo y la varianza del sistema recomendador. En los métodos colaborativos basados en memoria, no se asume ningún modelo latente. Los algoritmos trabajan directamente con las interacciones usuario-elemento: por ejemplo, los usuarios se representan mediante sus interacciones con los elementos y se utiliza una búsqueda de vecinos más cercanos en estas representaciones para producir sugerencias. Como no se asume ningún modelo latente, estos métodos teóricamente tienen un sesgo bajo, pero una varianza alta.

Por otro lado, en los métodos colaborativos basados en modelos, se asume algún modelo latente de interacción. El modelo se entrena para reconstruir los valores de las interacciones usuario-elemento a partir de su propia representación de usuarios y elementos. Las nuevas sugerencias se pueden realizar en función de este modelo. Las representaciones latentes de usuarios y elementos extraídas por el modelo tienen un significado matemático que puede ser difícil de interpretar para un ser humano. Al asumir un modelo para las interacciones usuario-elemento, estos métodos teóricamente tienen un sesgo más alto pero una varianza más baja que los métodos que no asumen un modelo latente (Rocca, 2019).

Por último, en los métodos basados en contenido también se asume algún modelo latente de interacción. Sin embargo, en este caso, el modelo se proporciona con contenido que define la representación de usuarios y/o elementos: por ejemplo, los usuarios se representan mediante características específicas y tratamos de modelar para cada elemento el tipo de perfil de usuario que le gusta o no le gusta dicho elemento. Aquí, al igual que en los métodos colaborativos basados en modelos, se asume un modelo de interacciones usuario-elemento. Sin embargo, este modelo está más restringido (debido a que se proporcionan representaciones de usuarios y/o elementos) y, por lo tanto, el método tiende a tener un sesgo más alto pero una varianza más baja (Rocca, 2019).

2.4 Métricas

2.4.1 MapK

El MAPK (Mean Average Precision at K) es una métrica utilizada para evaluar la calidad de un sistema de recomendación o de recuperación de información, especialmente cuando se trata de problemas de clasificación binaria. Proporciona una medida de la precisión promedio de los primeros K elementos recuperados por el sistema.

Como menciona el autor Konstantin Rink en el artículo *"Mean Average Precision at K (MAP@K) Clearly Explained"* publicado en *Towards Data Science*, el cálculo del MapK se puede dividir en los siguientes pasos:

- Para cada consulta o usuario en el conjunto de datos de prueba, se obtiene la lista de elementos recomendados por el sistema en orden de relevancia (por ejemplo, puntuaciones o probabilidades).
- Se determina si cada elemento recomendado es relevante o no según las etiquetas de relevancia conocidas. Por ejemplo, si los elementos tienen etiquetas binarias (0 o 1), se considera relevante aquellos con etiqueta 1.
- Se calcula la precisión acumulada en cada posición K para cada consulta o usuario. La precisión acumulada en la posición K se define como la proporción de elementos relevantes recuperados hasta esa posición K.
- Se calcula la precisión promedio (AP) para cada consulta o usuario dividiendo la suma de las precisiones acumuladas relevantes en cada posición relevante por el número total de elementos relevantes.
- Finalmente, se calcula el MapK promediando los valores de AP de todas las consultas o usuarios.
- Es importante tomar en cuenta que, si el usuario no tiene ninguna interacción relevante, la precisión promedio será de 0 (AP).

El MapK proporciona una medida más completa y robusta de la calidad del sistema de recomendación o recuperación de información en comparación con la precisión o la tasa de éxito. Es especialmente útil cuando el orden de los elementos recuperados es importante, como en los casos en que se presentan al usuario solo los primeros K elementos recomendados.

Capítulo 3. Datos

3.1 Descripción

El dataset contiene 17 meses (desde enero del 2015 hasta mayo del 2016) de datos de clientes del Banco Santander. Este se obtuvo de Kaggle y es importante recalcar que son datos ficticios.

Para cada cliente a cada foto temporal (columna de partición fechadato) se tiene informacion tanto del cliente como de su tenencia de productos. En la Tabla 1, podemos ver las características que se tiene del cliente, a estas las llamaremos variables de clientes; y en la Tabla 2 podemos ver las tenencias de productos que se tiene, a estas las llamaremos variables de tenencia. Podemos ver que estas últimas variables representan productos estándares para una entidad bancaria tradicional.

Tabla 1. Variables de Clientes

Variable	Descripción
fecha_dato	Columna de partición
ncodpers	Identificador de cliente
ind_empleado	Indicador del empleado
pais_residencia	País de residencia
sexo	Género
age	Edad
fecha_alta	La fecha en que el cliente se convirtió en el primer titular de un contrato en el banco
ind_nuevo	Índice de nuevos clientes
antiguedad	Meses de antigüedad
indrel	Indicador de cliente principal
ult_fec_cli_1t	Última fecha como cliente principal
indrel_1mes	Tipo de cliente al comienzo del mes
tiprel_1mes	Tipo de relación con el cliente al principio del mes
indresi	Índice de residencia (S/N)
indext	Índice de extranjeros (S/N)
conyuemp	Índice de cónyuges
canal_entrada	Canal utilizado por el cliente para unirse
indfall	Índice de fallecidos. N/S
tipodom	Tipo de dirección (1 dirección principal)
cod_prov	Código provincia
nomprov	Nombre de provincia
ind_actividad_cliente	Índice de actividad (activo; 0 cliente inactivo)
renta	Ingresos brutos del hogar
segmento	Segmentación

Tabla 2. Variables de tenencia de productos

Variable	Descripción
ind_ahor_fin_ult1	Cuenta de ahorros
ind_aval_fin_ult1	Garantías
ind_cco_fin_ult1	Cuenta corriente
ind_cder_fin_ult1	Cuenta derivada
ind_cno_fin_ult1	Cuenta de nómina
ind_ctju_fin_ult1	Cuenta Junior
ind_ctma_fin_ult1	Cuenta particular maxima
ind_ctop_fin_ult1	Cuenta particular
ind_ctpp_fin_ult1	Cuenta plus particular
ind_deco_fin_ult1	Depósitos a corto plazo
ind_deme_fin_ult1	Depósitos a mediano plazo
ind_dela_fin_ult1	Depósitos a largo plazo
ind_ecue_fin_ult1	Cuenta electrónica
ind_fond_fin_ult1	Fondos
ind_hip_fin_ult1	Hipotecas
ind_plan_fin_ult1	Pensiones
ind_pres_fin_ult1	Préstamos
ind_reca_fin_ult1	Impuestos
ind_tjcr_fin_ult1	Tarjeta de crédito
ind_valo_fin_ult1	Valores
ind_viv_fin_ult1	Cuenta vivienda
ind_nomina_ult1	Nómina
ind_nom_pens_ult1	Pensiones
ind_recibo_ult1	Débito directo

3.2 Análisis Exploratorio

3.2.1 Cantidad de Datos.

Con respecto a la cantidad de datos, tenemos en promedio ochocientos mil datos para cada fechadato, dando un total de 13,647,309 de datos en total, y un total de 70,7201 clientes únicos (tomar en cuenta que los datos de clientes se repiten mes a mes).

Tabla 3. Cantidad de datos por partición

Partición	Cantidad de datos
1/28/2015	625457
2/28/2015	627394
3/28/2015	629209
4/28/2015	630367
5/28/2015	631957
6/28/2015	632110
7/28/2015	829817
8/28/2015	843201
9/28/2015	865440
10/28/2015	892251
11/28/2015	906109
12/28/2015	912021
1/28/2016	916269
2/28/2016	920904
3/28/2016	925076
4/28/2016	928274
5/28/2016	931453

3.2.2 Evolución de la tenencia y adquisición de productos.

En el Anexo 1 podemos ver la evolución de la tenencia de cada producto a lo largo del tiempo. Podemos ver que hay ciertos productos que, claramente siguen una tendencia positiva como las cuentas corrientes (ind_cco_fin_ult1), las cuentas derivadas (ind_cder_fin_ult1), las cuentas nóminas (ind_cno_fin_ult1), las cuentas electrónicas (ind_ecue_fin_ult1), los impuestos o recaudación (ind_reca_fin_ult1), los valores (ind_valo_fin_ult1), las nóminas (ind_nomina_ult1), las pensiones (ind_nom_pens_ult1) y el débito directo (ind_recibo_ult1). Asimismo, podemos ver hay otros productos que claramente siguen una tendencia negativa como la cuenta de ahorros (ind_aval_fin_ult1), las garantías (ind_aval_fin_ult1), la cuenta particular (ind_ctop_fin_ult1), la cuenta plus particular (ind_ctpp_fin_ult1), depósitos a corto, mediano y largo plazo (ind_deco_fin_ult1, ind_deme_fin_ult1 y ind_dela_fin_ult1), hipotecas (ind_hip_fin_ult1), préstamos (ind_pres_fin_ult1) y la cuenta vivienda (ind_viv_fin_ult1). El resto de los productos no siguen una tendencia clara.

Por otro lado, en el Anexo 2, podemos ver cómo evoluciona la adquisición de los productos a lo largo del tiempo, es decir la cantidad de clientes que adquieren el producto en ese mes (la tenencia pasa de 0 en el mes anterior a 1 en el mes actual). Lo más relevante que podemos notar es que hay ciertos productos que se adquieren en magnitudes mucho menores a otros, como la cuenta de ahorros (ind_aval_fin_ult1), las garantías (ind_aval_fin_ult1) y las cuentas derivadas (ind_cder_fin_ult1) principalmente.

Finalmente, en el Anexo 3 podemos ver la proporción de adquisición de los productos, es decir para cada producto en cada mes vemos cuantas adquisiciones ha habido en comparación a la cantidad de clientes. Este indicador luego será de gran importancia ya que nos dirá que tan desbalanceadas están nuestras clases en la construcción del modelo híbrido; más adelante entraremos en detalle, pero la adquisición del producto en un cierto mes viene a ser la target, y como podemos ver para la gran mayoría de productos esta se encuentra en desbalance.

3.2.3 Análisis de Nulos

En la siguiente tabla podemos ver en orden decreciente las variables con % de nulos diferentes de cero. A priori iremos descartando las variables conyuemp (indicador de cónyuge) ult_fec_cli_1t (última fecha como cliente principal) para y renta para evitar problemas.

Tabla 4. Porcentaje de datos nulos

Variable	% De nulos
conyuemp	99.99%
ult_fec_cli_1t	99.82%
renta	20.48%
segmento	1.39%
canal_entrada	1.36%
indrel_1mes	1.10%
tiprel_1mes	1.10%
cod_prov	0.69%
nomprov	0.69%

Capítulo 4. Diseño y Desarrollo

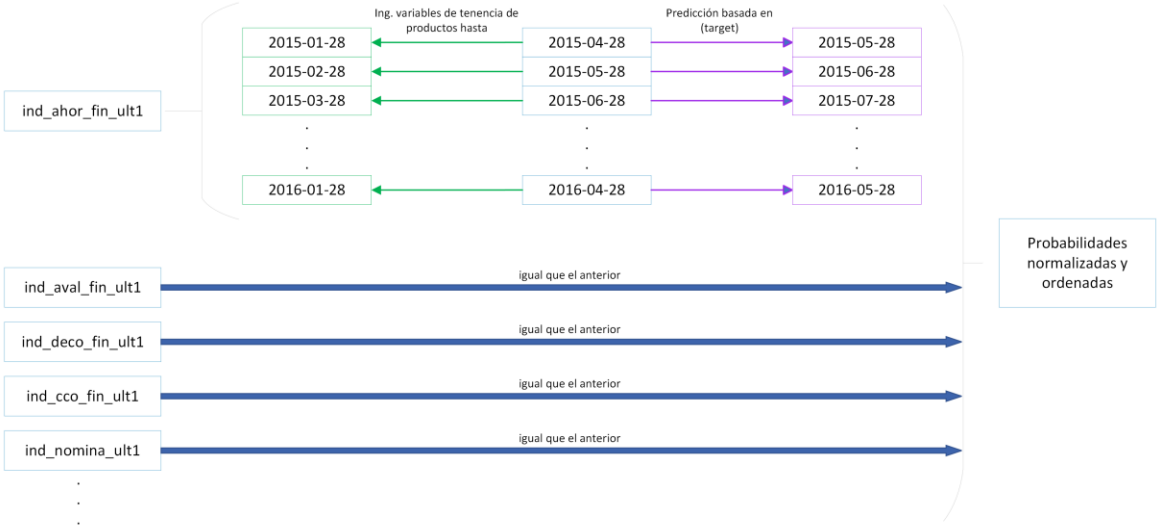
4.1 Introducción

En el siguiente apartado se detallarán cuáles fueron las soluciones planteadas para construir el recomendador y como fueron diseñadas estas. En primer lugar, se decidió construir una estructura de modelo desde cero, al cual llamaremos "Modelo Híbrido", en donde se modela la tenencia de cada producto para cada mes independientemente con un modelo de predicción binaria; en segundo lugar, se construyó el sistema recomendador utilizando las técnicas de filtrado colaborativo basado en modelos detalladas en el Estado del Arte, estas, a diferencia del modelo híbrido se basan en la factorización matricial principalmente y en las redes neuronales. A continuación, se detallará más a profundidad cada solución.

4.2 Modelo Híbrido

Como ya se ha mencionado, este modelo busca predecir independientemente para cada producto la tenencia de este en un determinado mes. Para ello, será necesario armar un modelo predictivo por producto, cada uno con su propia población base, target y variables predictoras; para luego unificar las salidas de dichos modelos en un vector de probabilidades, normalizarlas, ordenarlas de mayor a menor y recomendar los top K productos de dicho vector. A continuación, en la Figura 3 podemos ver el esquema básico del modelo híbrido propuesto, el cual se comentará más a detalle en las siguientes secciones.

Figura 3. Estructura Modelo Híbrido



4.2.1 Construcción Target

La target se estableció como la tenencia del producto al mes siguiente, con la condición de que el cliente al mes actual no tenga el producto. Por lo tanto, para ser más claros la target viene a ser la adquisición como tal del producto. Como podemos ver en la siguiente Figura 4, se pierde el último mes de datos disponible (mayo 2016) por no tener datos para la target, por lo tanto, estaría correcto decir que los datos de mayo 2016 solo nos sirven para la extracción de la target.

Figura 4. Fechadato target para cada foto temporal

fechadato	target
2015-01	2015-02
2015-02	2015-03
2015-03	2015-04
2015-04	2015-05
2015-05	2015-06
2015-06	2015-07
2015-07	2015-08
2015-08	2015-09
2015-09	2015-10
2015-10	2015-11
2015-11	2015-12
2015-12	2016-01
2016-01	2016-02
2016-02	2016-03
2016-03	2016-04
2016-04	2016-05
2016-05	

4.2.2 Población base

Se establece la población base como a clientes del banco que a la fecha (fechadato) no tienen el producto correspondiente y que no son considerados como nuevos. También es relevante mencionar que nos quedamos únicamente con clientes que tanto a la fecha como a la fecha target (1 mes posterior) se encuentran en la base de datos, es decir son clientes que tienen una variable target disponible.

4.2.3 Limitación de datos de tenencia de productos para predicción

Como se presentó en el apartado anterior existe una gran limitación de la target para la gran mayoría de productos, por lo tanto, se decidió limitar la cantidad de productos a incluir en la construcción de este modelo híbrido siguiendo la siguiente regla: aquellos productos que después de realizar el *downsampling* hasta un 50% por encima de la clase minoritaria, tengan menos de 1000 datos para entrenamiento serán

descartados. En esta etapa se descartaron los siguientes doce productos por falta de datos: ind_ahor_fin_ult1, ind_aval_fin_ult1, ind_cco_fin_ult1, ind_cder_fin_ult1, ind_ctju_fin_ult1, ind_deco_fin_ult1, ind_deme_fin_ult1, ind_hip_fin_ult1, ind_fond_fin_ult1, ind_plan_fin_ult1, ind_pres_fin_ult1 y ind_viv_fin_ult1.

4.2.4 Ingeniería de Variables

Una vez establecida la target y definida la población base, se decidió construir un conjunto de variables en base a las características de los usuarias obtenidas en el dataset, que logren capturar características de la evolución de los clientes en el banco a lo largo del tiempo. A este proceso le llamaremos ingeniería de variables. Para la construcción de variables se tomó como máximo 3 meses de histórico, lo cual limita nuestros datos para entrenamiento, validación y test de 16 a 13 meses (los 3 primeros fechadato enero febrero y marzo 2015 se pierden). En la Figura 5 podemos ver en morado el fechadato y en naranja las fechas que cada fechadato toma en cuenta para la ingeniería de variables.

Figura 5. Historial de fechas para la construcción de variables para cada fechadato

fechadato e historial para ingeniería de variables												
2015-01	2015-01	2015-01	2015-01	2015-01	2015-01	2015-01	2015-01	2015-01	2015-01	2015-01	2015-01	2015-01
2015-02	2015-02	2015-02	2015-02	2015-02	2015-02	2015-02	2015-02	2015-02	2015-02	2015-02	2015-02	2015-02
2015-03	2015-03	2015-03	2015-03	2015-03	2015-03	2015-03	2015-03	2015-03	2015-03	2015-03	2015-03	2015-03
2015-04	2015-04	2015-04	2015-04	2015-04	2015-04	2015-04	2015-04	2015-04	2015-04	2015-04	2015-04	2015-04
2015-05	2015-05	2015-05	2015-05	2015-05	2015-05	2015-05	2015-05	2015-05	2015-05	2015-05	2015-05	2015-05
2015-06	2015-06	2015-06	2015-06	2015-06	2015-06	2015-06	2015-06	2015-06	2015-06	2015-06	2015-06	2015-06
2015-07	2015-07	2015-07	2015-07	2015-07	2015-07	2015-07	2015-07	2015-07	2015-07	2015-07	2015-07	2015-07
2015-08	2015-08	2015-08	2015-08	2015-08	2015-08	2015-08	2015-08	2015-08	2015-08	2015-08	2015-08	2015-08
2015-09	2015-09	2015-09	2015-09	2015-09	2015-09	2015-09	2015-09	2015-09	2015-09	2015-09	2015-09	2015-09
2015-10	2015-10	2015-10	2015-10	2015-10	2015-10	2015-10	2015-10	2015-10	2015-10	2015-10	2015-10	2015-10
2015-11	2015-11	2015-11	2015-11	2015-11	2015-11	2015-11	2015-11	2015-11	2015-11	2015-11	2015-11	2015-11
2015-12	2015-12	2015-12	2015-12	2015-12	2015-12	2015-12	2015-12	2015-12	2015-12	2015-12	2015-12	2015-12
2016-01	2016-01	2016-01	2016-01	2016-01	2016-01	2016-01	2016-01	2016-01	2016-01	2016-01	2016-01	2016-01
2016-02	2016-02	2016-02	2016-02	2016-02	2016-02	2016-02	2016-02	2016-02	2016-02	2016-02	2016-02	2016-02
2016-03	2016-03	2016-03	2016-03	2016-03	2016-03	2016-03	2016-03	2016-03	2016-03	2016-03	2016-03	2016-03
2016-04	2016-04	2016-04	2016-04	2016-04	2016-04	2016-04	2016-04	2016-04	2016-04	2016-04	2016-04	2016-04

Se construyeron las siguientes variables:

- Diferenciando para cada producto:
    - Tenencia del producto en el mes 1 (un mes posterior a la fecha)
    - Tenencia del producto en el mes 2 (dos meses posteriores a la fecha)
    - Tenencia del producto en el mes 3 (tres meses posteriores a la fecha)
    - Cantidad de veces que ha adquirido o se ha dado de baja del producto
    - Cantidad de meses consecutivos que la tenencia del producto ha sido cero
- No diferenciando para cada producto

- Cantidad de productos adquiridos el mes pasado
- Concatenación del indicador de actividad de cliente en el mes actual y en el mes pasado
- Concatenación del tipo de relación del cliente en el mes actual y en el mes pasado

4.2.5 Manejo de desbalance de clases

Para manejar el desbalance de clases, se hizo *downsampling* a un rango de 10% - 50% de datos por encima de la clase minoritaria dependiendo del producto, en la medida que el número de cantidad de datos resultantes para entrenamiento sea coherente y suficiente para obtener resultados.

4.2.6 Distribución de datos para train validación y test

Para la gran mayoría de productos se tomó el último mes de datos (abril 2016) para test el penúltimo mes de datos para validación (marzo 2016) y el resto para entrenamiento. Para casos específicos en donde la distribución de la target no era homogénea a lo largo de los meses de datos disponibles, se retrocedían las fechas para obtener los conjuntos de test y validación y se reducía la cantidad de datos para entrenamiento.

4.2.7 Modelado

Principalmente se probaron y compararon un Random Forest y un XGBoost con parámetros optimizados utilizando Optuna, una biblioteca de optimización de hiperparámetros de código abierto que utiliza técnicas de búsqueda automática para encontrar la configuración óptima de hiperparámetros.

Modelo del producto ind_cno_fin_ult1 (Cuenta nómina):

a. XGBoost

Parámetros:

Tabla 5. Parámetros finales XGBoost - ind_cno_fin_ult1

Parámetro	Valor
learning_rate	0.12411
max_depth	6
n_estimators	114
gamma	1.3739
min_child_weight	0.0746
subsample	0.6222
colsample_bytree	0.7814

Resultados:

Tabla 6. Métricas XGBoost - ind_cno_fin_ult1

	Train	Test
Recall	0.9239	0.8812
Accuracy	0.8853	0.8850

Figura 6. Curva ROC train y test - XGBoost - ind_cno_fin_ult1

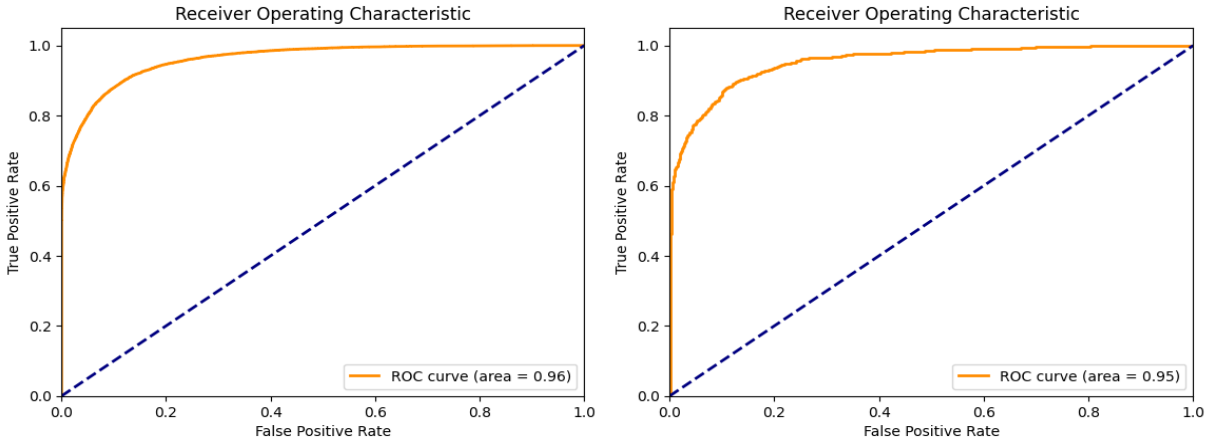
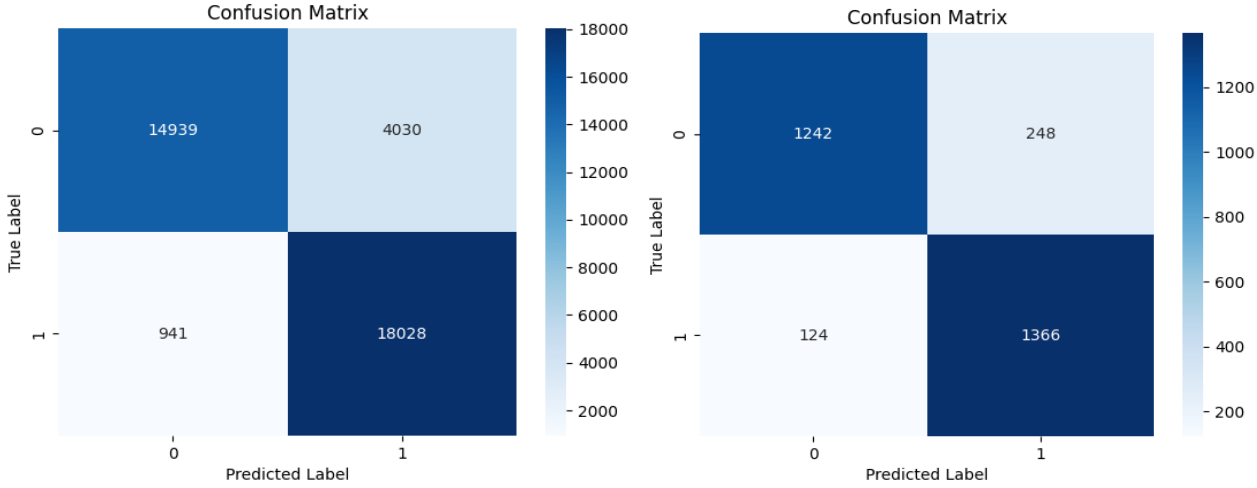


Figura 7. Matrices de confusión train y test - XGBoost - ind_cno_fin_ult1



b. Random Forest

Parámetros:

Tabla 7. Parámetros finales XGBoost - ind_cno_fin_ult1

Parámetro	Valor
criterion	entropy
max_features	30
n_estimators	991
min_samples_split	7
min_samples_leaf	2

Resultados:

Tabla 8. Métricas Random Forest - ind_cno_fin_ult1

	Train	Test
Recall	0.9575	0.8906
Accuracy	0.9181	0.8778

Figura 8. Curva ROC train y test - Random Forest- ind_cno_fin_ult1

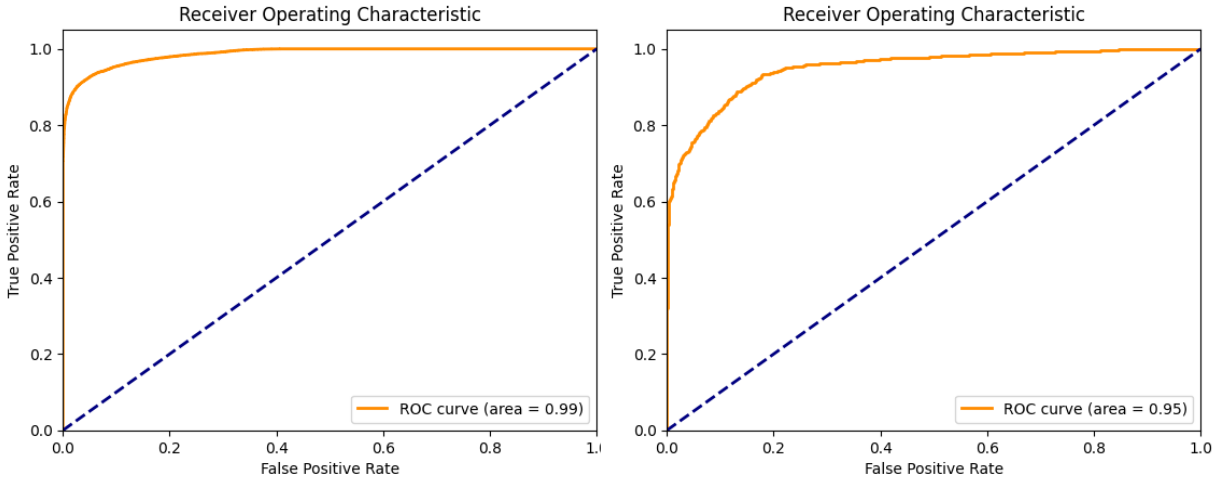
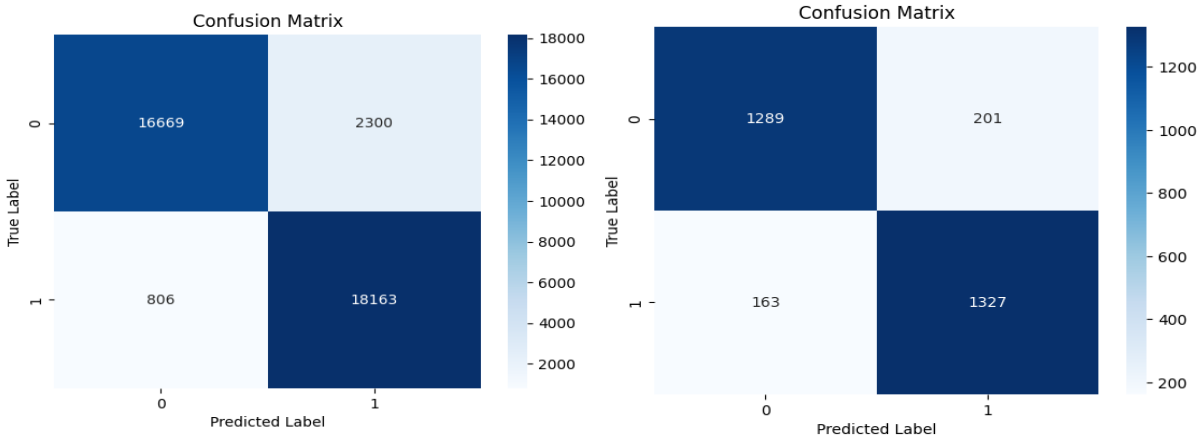


Figura 9. Matrices de confusión train y test – Random Forest - ind_cno_fin_ult1



En primer lugar, cabe recalcar que estos resultados fueron obtenidos son un con *threshold* de 40%. Esto, ya que se quiere minimizar la tasa de falsos negativos, incluso si esto implica aumentar los falsos positivos. Esto ya que es más preferible decir que un cliente tendrá un producto en el futuro y equivocarnos que decir

que este no, cuando en realidad sí lo tendrá. Esto representa una oportunidad de negocio perdida, además que el simple hecho de predecir un falso positivo no implica un gasto extra de recursos, sino solo una recomendación.

Por otro lado, podemos ver que los resultados de ambos modelos son bastante similares, sin embargo, las métricas obtenidas en el train y test del Random Forest se ven ligeramente espaciadas, indicando que este modelo presenta indicios de sobreajuste. A pesar de que el sobreajuste visto en el Random Forest no es tan grande, los resultados del XGBoost son bastante cercanos y aceptables, suficientes como para elegirlo ganador.

4.3 Filtrado Colaborativo

4.3.1 Factorización Matricial Generalizada (GMF)

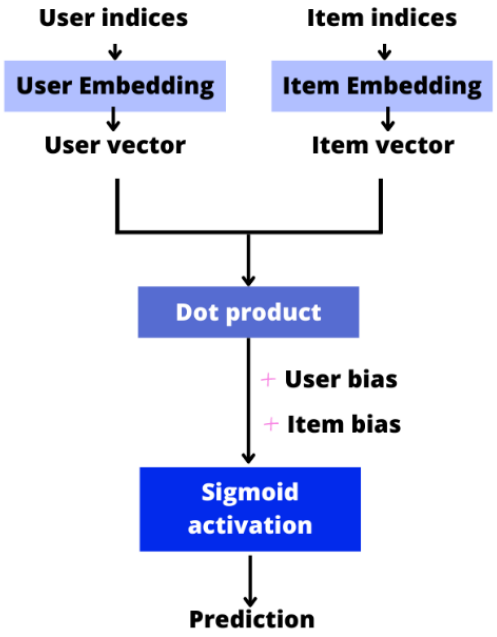
Tomando como base la estructura de modelo propuesta por Siddhartha Banerjee en su trabajo Collaborative Filtering for Movie Recommendations (2020), se implementa un modelo que utiliza embeddings para usuarios y productos. A manera de resumen, este modelo descompone la matriz de interacciones usuario ítem en vectores latentes usuario e item respectivamente, para luego por medio de producto punto de 2 vectores sea capaz de predecir y recomendar elementos relevantes.

El modelo se encuentra compuesto por dos capas de embedding, una para representar a los usuarios y otra a los productos, seguidas de dos capas de *embedding* para representar los sesgos (user *bias e item bias*) correspondientes. Los inputs del modelo consisten en pares de índices que representan usuarios y productos, es decir cada interacción diferente a cero de usuarios con productos. Finalmente, se calcula el producto tensorial de los *embeddings* de usuario y producto, se suman los sesgos y se aplica una función de activación sigmoide para generar una predicciones.

Un *embedding* es una representación de baja dimensión de un objeto en un espacio vectorial, y su tamaño es un hiperparámetro clave en la estructura de este modelo; determina la dimensionalidad de los *embeddings* y tiene un impacto en el rendimiento y la capacidad del modelo para capturar patrones y características relevantes. Incrementar este parámetro puede hacer que el modelo capte más detalles en las representaciones usuario ítem aumentando el rendimiento del modelo, sin embargo, esto conlleva un coste computacional mucho más elevado. Por otro lado, disminuir este hiperparámetro, si bien es menos costos computacionalmente, puede hacer que el modelo tenga representaciones menos complejas y más compactas, limitando la capacidad del modelo de captar características y patrones importantes.

Si bien Banerjee (2020) en su trabajo utiliza un tamaño 50 de *embedding*, aquí este se redujo a 20 debido a la falta de recursos computacionales.

Figura 10. Arquitectura GMF



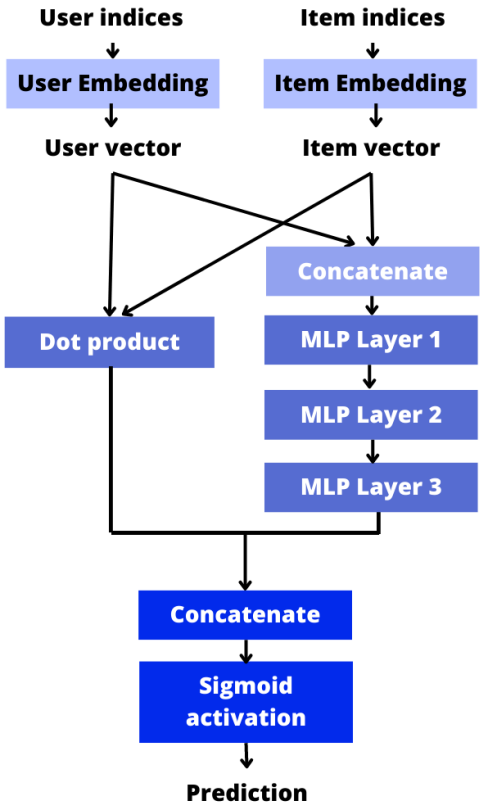
4.3.2 Factorización Matricial Neuronal (NMF)

Se tomó como guía el artículo *Neural Collaborative Filtering* (NCF) de He et al., (2017) presentado en el capítulo 2 Estado del Arte, en donde proponen la arquitectura del modelo *Neural Matrix Factorization* (NMF). Los autores agregan al modelo de GMF el componente de un perceptrón multicapa (MLP), y mencionan que, de esta manera logran combinar la linealidad de una simple factorización matricial con la no linealidad de un perceptrón multicapa para modelar estructuras latentes usuario producto.

En la Figura 11 podemos ver que la arquitectura del modelo NMF consiste en la combinación de dos enfoques: GMF y MLP. El modelo toma como entrada dos índices: uno para el usuario y otro para el ítem. El GMF realiza una factorización matricial mediante la creación de vectores de usuario e ítem, y calcula una predicción combinando estos vectores junto con sesgos específicos para cada usuario y elemento. Por otro lado, el MLP concatena los vectores de usuario y elemento y los pasa a través de varias capas densas para aprender representaciones no lineales. Finalmente, las salidas del GMF y MLP se concatenan y se

pasan a través de una capa de salida densa con una función de activación sigmoide para obtener la predicción final de relevancia del elemento para el usuario.

Figura 11. Arquitectura NMF



Como se mencionó en el capítulo 2 Estado del Arte, los autores indican que los mejores resultados se obtienen al ensamblar el modelo a partir de los modelos GMF y MLP previamente pre entrenados. No obstante, en esta etapa de la investigación se optó por explorar inicialmente la construcción desde cero, es decir, entrenar el modelo en su totalidad, tanto el componente GMF como el MLP, compartiendo parámetros, para posteriormente, realizar la prueba de pre-entrenar los modelos GMF y MLP de manera independiente con parámetros ya definidos, para luego combinar las salidas de estos.

En el modelo NMF pre entrenado, se optó por utilizar el optimizador SGD, el mismo que utilizan los autores He et al., (2017) en el artículo presentado anteriormente.

Tanto como para el modelo de Factorizacion Matricial Generalizada como para el de Factorizacion Matricial Neuronal se dividió el conjunto de datos para:

- Train. Datos de clientes a fecha 2015-06-28. Este conjunto se dividió 80% para entrenamiento y 20% para validación.
- Test. Para evaluar el recomendador se toman los datos de los usuarios al mes siguiente de entrenamiento, es decir a 2015-07-28, y con la condición de que estos usuarios hayan participado durante el entrenamiento.

A continuación, en la Tabla 9 podemos ver con cuantas interacciones (tenencia del producto igual a 1) se cuenta para cada producto en cada conjunto de datos. En total se tuvo 759,443 interacciones para train y 190,050 interacciones para validación, representando en este conjunto de datos un total de 514,946 usuarios únicos a los cuales se les hará luego la recomendación

Tabla 9. Cantidad de interacciones usuario producto

Producto	Train	Validación
ind_ahor_fin_ult1	63	11
ind_aval_fin_ult1	19	1
ind_cco_fin_ult1	321126	80131
ind_cder_fin_ult1	207	59
ind_cno_fin_ult1	41641	10398
ind_ctju_fin_ult1	5228	1297
ind_ctma_fin_ult1	4331	1018
ind_ctop_fin_ult1	70871	17905
ind_ctpp_fin_ult1	24401	6016
ind_deco_fin_ult1	488	113
ind_deme_fin_ult1	980	237
ind_dela_fin_ult1	24012	6106
ind_ecue_fin_ult1	44245	11326
ind_fond_fin_ult1	10255	2584
ind_hip_fin_ult1	3356	802
ind_plan_fin_ult1	5219	1248
ind_pres_fin_ult1	1337	343
ind_reca_fin_ult1	29097	7258
ind_tjcr_fin_ult1	25293	6295
ind_valo_fin_ult1	13405	3290
ind_viv_fin_ult1	2250	529
ind_nomina_ult1	30397	7605
ind_nom_pens_ult1	33293	8388
ind_recibo_ult1	67929	17090

Capítulo 5. Pruebas y Resultados

5.1 Modelo Híbrido

5.1.1 Evaluación de las predicciones de cada producto

Recall y Accuracy. A continuación, se muestran los resultados en el conjunto de train y test para cada producto modelado.

Tabla 10. Recall de los modelos finales de cada producto

Producto	Train	Test
ind_cno_fin_ult1	0.9514	0.9718
ctmo_fin_ult1	0.98105	1.0000
ind_ctop_fin_ult1	0.97688	0.9889
ind_ctpp_fin_ult1	0.9837	1.0000
ind_dela_fin_ult1	0.9139	0.9862
ind_ecue_fin_ult1	0.9382	0.9518
ind_nom_pens_ult1	0.1901	0.2104
ind_nomina_ult1	0.9266	0.9527
reca_fin_ult1	0.8666	0.9865
ind_recibo_ult1	0.9189	0.9361
ind_tjcr_fin_ult1	0.9310	0.9440
ind_valo_fin_ult1	0.9352	0.8672

Tabla 11. Accuracy de los modelos finales de cada producto

Producto	Train	Test
ind_cno_fin_ult1	0.7799	0.7569
ctmo_fin_ult1	0.8416	0.8005
ind_ctop_fin_ult1	0.8465	0.8209
ind_ctpp_fin_ult1	0.8771	0.8457
ind_dela_fin_ult1	0.6939	0.6241
ind_ecue_fin_ult1	0.7854	0.7704
ind_nom_pens_ult1	0.8738	0.8336
ind_nomina_ult1	0.8725	0.8386
reca_fin_ult1	0.7813	0.7813
ind_recibo_ult1	0.8598	0.7892
ind_tjcr_fin_ult1	0.8850	0.7840
ind_valo_fin_ult1	0.8228	0.7646

En primer lugar, analizando la Tabla 10 podemos ver que la gran mayoría de productos obtienen un recall dentro de un mismo rango de valores, es decir estos no varían mucho entre ellos; no obstante, sí vemos que, el modelo del producto ind_nom_pens_ult1 presenta un recall bastante atípico, y por debajo del rango del resto de productos. Asimismo, podemos ver que hay productos como ctmo_fin_ult1, ind_ctpp_fin_ult1 que obtienen un 100% de recall, lo cual es bastante sospechoso y podría ser un indicador de sensibilidad ante

el *threshold* (que se bajó hasta un 35%). Asimismo, vemos que hay varios productos que obtienen un recall en test más alto que en train, lo cual también se puede explicar por poner el *threshold* bastante bajo, debido a que se tenía en foco disminuir la tasa de falsos negativos.

Por otro lado, en la Tabla 11 podemos ver que el valor del accuracy para todos los productos es más bajo que el recall. Sin embargo, acá si obtenemos resultados más altos en train que en test en la mayoría de los productos, no como en el recall. Un alto recall y un bajo accuracy podría indicar que el modelo está siendo efectivo para identificar los casos positivos, pero puede estar cometiendo errores al predecir los casos negativos (clase mayoritaria). Esto es común en problemas de desbalanceo de clases, donde el enfoque principal del modelo está en la detección de los casos positivos importantes, incluso a expensas de una clasificación menos precisa en general.

Finalmente, podemos decir que los productos ind_dela_fin_ult1, ind_recibo_ult1, ind_tjcr_fin_ult1, ind_valo_fin_ult1 son los que más sufren de sobreajuste.

Lift. Asimismo, se calculó el Lift para cada producto. Este es una métrica que se utiliza para evaluar el rendimiento del modelo en términos de su capacidad para distinguir clases positivas de clases negativas y para comparar este rendimiento con un escenario de referencia. Se le conoce como lift (levantamiento) porque compara los resultados obtenidos en cada tier con un resultado base o aleatorio. Se calcula de la siguiente manera:

- Se ordenan las probabilidades en orden descendiente
- Se dividen los datos en n *tiers* (deciles, por ejemplo)
- Se calcula la tasa de respuesta esperada (proporción de respuestas positivas) en cada tier.
- Se calcula el lift dividiendo la tasa de respuesta esperada del grupo sobre la tasa de respuesta observada en todo el conjunto de datos.

Por lo tanto:

- lift >1: El modelo tiene un rendimiento mejor que el escenario de referencia; es decir, el modelo tiene una capacidad superior para distinguir la clase positiva de la clase negativa en comparación con una clasificación aleatoria.
- lift = 1: El rendimiento del modelo es similar al escenario de referencia; es decir, el modelo no proporciona una mejora significativa en la capacidad de distinguir entre la clase positiva y la clase negativa en comparación con una clasificación aleatoria

- lift < 1: El rendimiento del modelo es inferior al escenario de referencia; es decir, el modelo tiene dificultades para distinguir correctamente la clase positiva y la clase negativa, y su rendimiento es peor que una clasificación aleatoria

Como es de esperarse, debido a que se ordenan descendientemente por probabilidades los datos, los segmentos inferiores tendrán menos instancias positivas que los segmentos superiores; además, dado que tenemos modelos con targets bastante desbalanceadas, el lift en los *tiers* inferiores no son tan significativos como los superiores, y la mayoría de las clases positivas se van a concentrar en la parte superior y muy comúnmente los *tiers* inferiores no contendrán clases positivas.

Dicho esto, en las figuras 12, 13 y 14 podemos ver tanto el lift de cada *tier* (en color azul) como el acumulado por tiers (en color rojo), de los productos ind_cno_fin_ult1, ind_tjcr_fin_ult1 y ind_nom_pens_ult1 el resto de las gráficas se encuentran en el Anexo 4. Hemos elegido estos tres productos, debido a que buscamos comparar los modelos que pueden ser considerados como mejores (ind_cno_fin_ult1 y ind_tjcr_fin_ult1) contra el modelo que obtuvo peores resultados (ind_nom_pens_ult1).

Como mencionamos anteriormente, los top tiers deberían obtener los mejores resultados, en este caso un lift superior a 1, como podemos ver en las figuras 12 13 y 14, los tres modelos tienen un lift superior a uno en el primer decil, es decir, para el 10% de la población con probabilidades más altas, los 3 modelos son capaces de hacer mejores predicciones que una clasificación aleatoria. Sin embargo, si tuviéramos que decir cuál de los tres modelos mostrados es el mejor clasificador, sería el modelo del producto tarjeta de crédito (figura 13) ya que en el primer decil clasifica 8 veces mejor que una clasificación aleatoria mientras que el modelo de la cuenta nomina lo hace 4 veces mejor, y el modelo del producto nomina pensión tan solo 1.5 veces mejor.

Por otro lado, podemos ver que los modelos de los productos cuentan nómina y nomina pensión (figura 12 y 14), tienen en el segmento 2 un lift acumulado superior al del primer decil. Esto se explica debido a que el segundo decil presenta más instancias de clase positiva que el primero, siendo esto un indicador de que el modelo está teniendo dificultades para identificar correctamente las instancias positivas más relevantes.

Figura 12. Lift ind_cno_fin_ult1

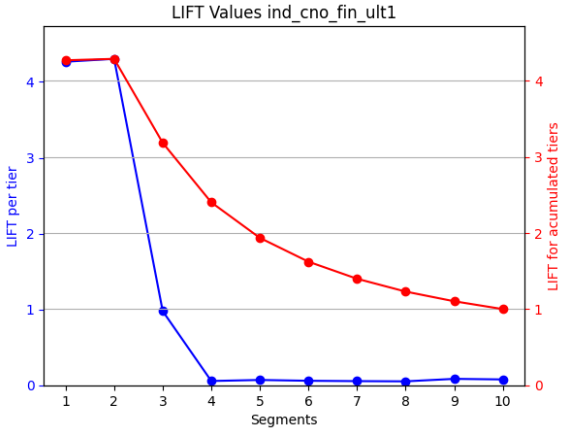


Figura 13. Lift ind_tjcr_fin_ult1

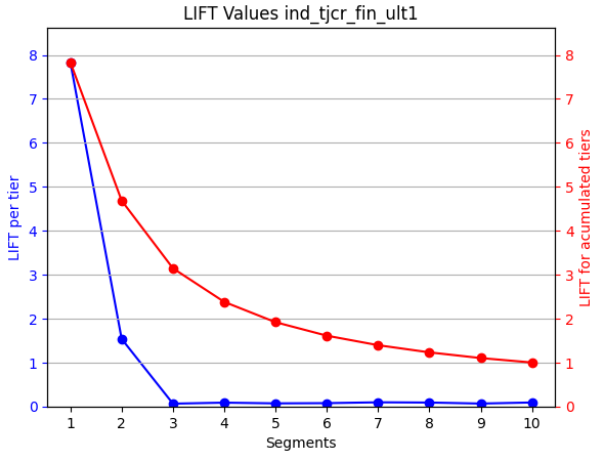
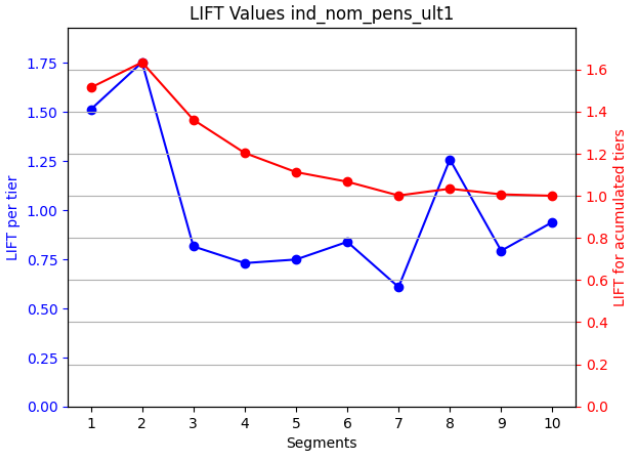


Figura 14. Lift ind_nom_pens_ult1



5.1.2 Evaluación del recomendador

Se utilizará la métrica Mapk (*Mean Average Precision at K*) para evaluar las recomendaciones del modelo híbrido. Esta se utiliza para evaluar la calidad y el rendimiento de un sistema de recomendación en función de la precisión de las recomendaciones. El MapK, mide la precisión de las primeras K recomendaciones hechas por un sistema de recomendación, tomando en cuenta la proporción de elementos relevantes que aparecen en las primeras K recomendaciones. Por lo tanto, recomendar un producto relevante para el usuario en la quinta posición de la recomendación, no tiene el mismo impacto que recomendar un producto relevante para el usuario en la primera o segunda posición de la recomendación.

Tabla 12. MapK Modelo Híbrido

K	Test
3	0.2876
5	0.2615
7	0.2214
9	0.2130

5.1.3 Comparación con Modelo Baseline

Tomaremos como modelo *baseline* la recomendación de los top 3, 5 7 y 9 productos más populares del dataset calcularemos el MAPk para cada valor de k correspondiente y lo compararemos con los resultados de nuestro modelo.

Podemos ver que, el modelo híbrido realiza recomendaciones ligeramente mejores a una recomendación estándar basada en la popularidad. Además, notamos que el resultado obtenido del MapK, disminuye a medida que K aumenta. Esto podría deberse a que, a medida que aumenta K aumentan los elementos no relevantes en las recomendaciones, y además porque en el modelo *baseline* se han recomendado los productos por popularidad y puede que los productos rankeados como 5 6 y 7 de popularidad tengan una gran diferencia en popularidad a los productos rankeados como 1 2 y 3; es decir que exista un salto de popularidad significativo entre un producto del *rank* y otro.

Tabla 13. MapK Modelo Baseline

K	Test
3	0.2398
5	0.2117
7	0.1946
9	0.1605

5.1.4 Importancia de Variables y Valores SHAP

A continuación, analizaremos la importancia de variables en alguno de nuestros modelos. Para la cuenta nómina (ind_cno_fin_ult1) podemos ver que algunas de las variables más significativas son la concatenación de la actividad del cliente con la del mes pasado con valor 0-0 (last_concat_ind_actividad_cliente_0_0) (nota. es la variable *encoded*), así como la concatenación de la relación del cliente con el banco con la del mes pasado con valor 1-1 (last_concat_tiprel_1mes_1_1), variables que indican si el cliente pertenece a las provincias de Islas Baleares y Ciudad Real, así como los valores de la misma tenencia del producto en los meses 1 2 y 3 anteriores al mes correspondiente (lag 1 2 y 3 de ind_cno_fin_ult1).

Por otro lado, podemos ver que algunas de las variables que tienen una relación directa con la target, es decir que a mayor valor de la variable más probable es que el cliente tenga el producto, son ind_recibo_fin_ult1, ind_nomina_fin_ult1 y los tres *lags* de la target, es decir que en el presente el cliente tenga el producto recibo y nómina y que en los últimos 3 meses haya tenido cuenta nómina.

De la misma manera, vemos que algunas de las variables con relación inversa con la target, es decir que a menor valor de la variable más probable es que el cliente tenga el producto son nomprov_ISLES_BALEARS, last_concat_ind_actividad_cliente_0_0, antigüedad e ind_cco_fin_ult1; por lo tanto clientes que sean de la provincia de islas Baleares, o que su actividad en el banco no sea ni haya sido en el mes pasado cero, o que tengan una antigüedad baja, o que no tengan el producto cuenta corriente en el mes presente, son más probables de tener el producto cuenta nómina al mes siguiente (que la target tome un valor de 1).

Figura 15. Feature importance ind_cno_fin_ult1

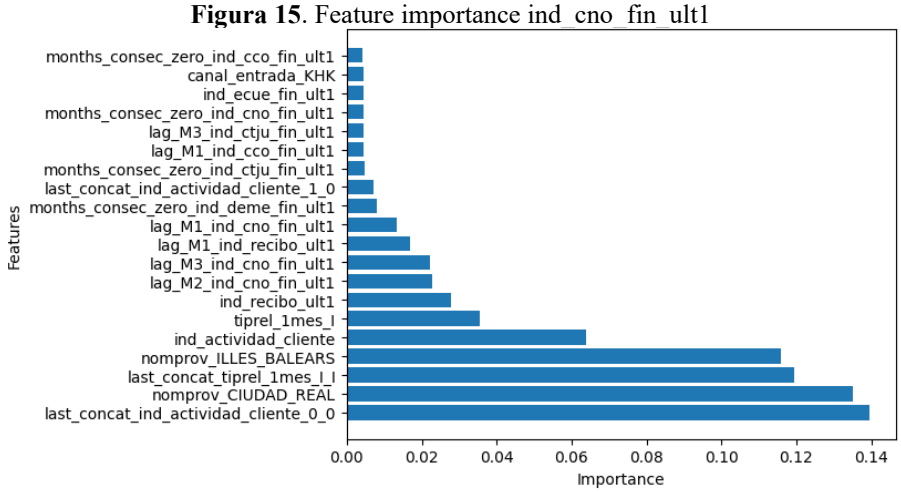
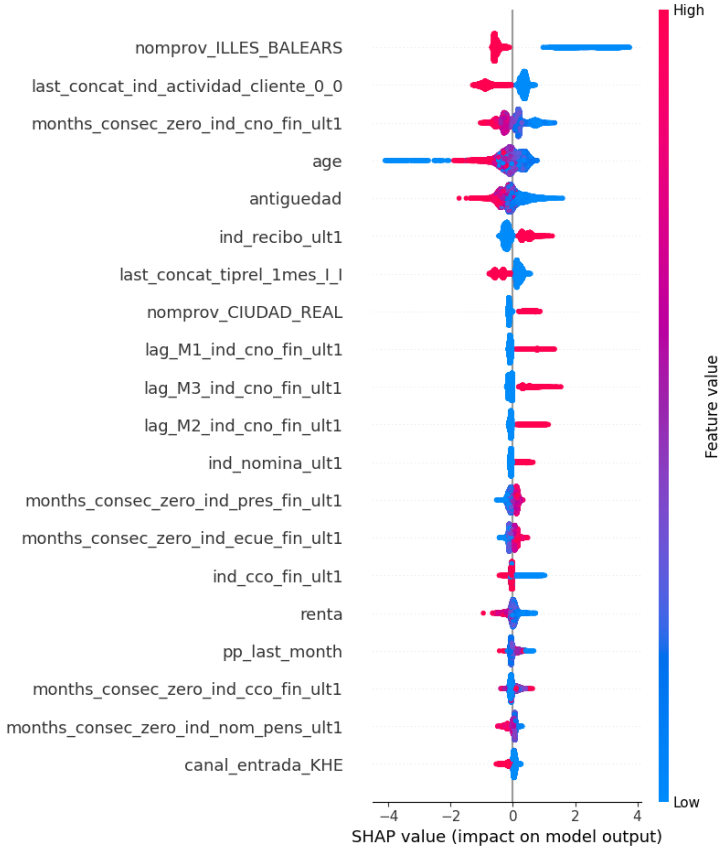


Figura 16. SHAP values ind_cno_fin_ult1



Analizando el producto Tarjeta de Crédito (Figuras 17 y 18) podemos ver que por lejos la variable más importante es que la que nos indica si el tipo de relación del cliente con el banco es activa o no (tiprel_1mes_A), seguida por la cantidad de productos adquiridos en el mes anterior (pp_last_month) y por la concatenación de la relación del cliente con el banco con la del mes pasado con valor 1-1 (last_concat_tiprel_1mes_1_1). De igual manera que el producto anterior, podemos ver que figuran como importantes los lags de la tenencia del producto en los 3 meses anteriores (lag 1 2 y 3 de ind_tjcr_fin_ult1), sin embargo, podemos ver que la tenencia del producto tres meses anteriores a la fecha es más significativa que la tenencia a los dos meses e incluso mucho más significativa que la tenencia en el mes anterior a la fecha. No obstante, al ver la Figura 18, nos damos cuenta de que estas tres últimas variables tienen una relación directa con la target, es decir que a mayores valores (1, porque es una variable binaria) más probable es que la target tome un valor de 1. Asimismo, podemos ver que, la variable months_concat_zero_ind_tjcr_fin_ult1 presenta una fuerte relación inversa con la target.

Figura 17. Feature importance ind_tjcr_fin_ult1

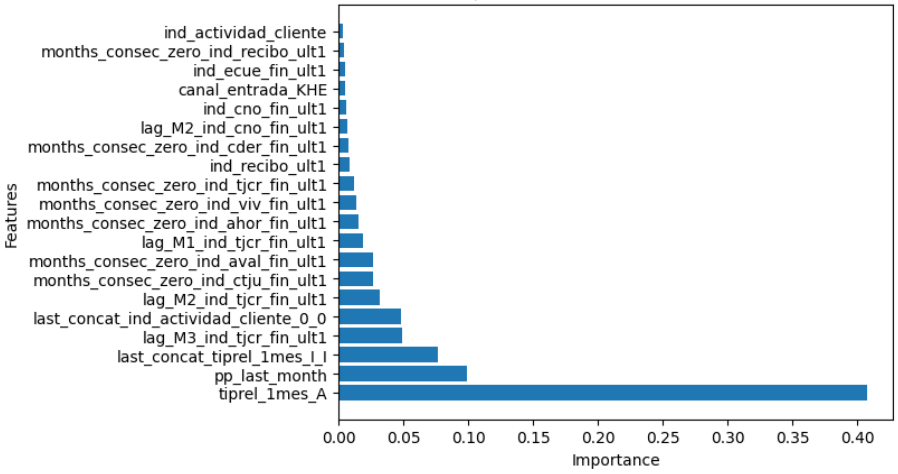
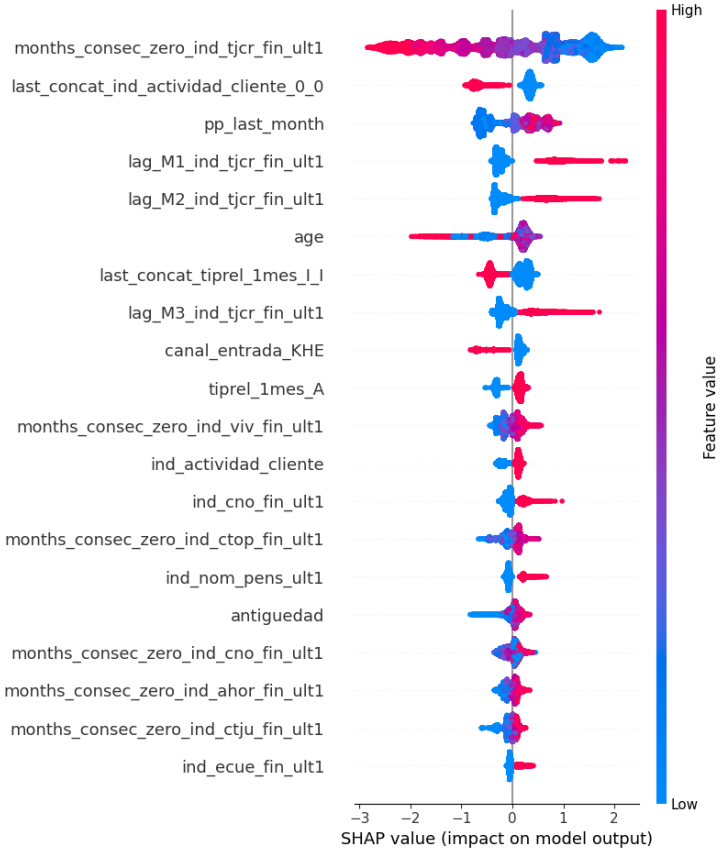


Figura 18. SHAP values ind_tjcr_fin_ult1



Finalmente, si analizamos el modelo del producto Nómina Pensión (Figuras 19 y 20), algo curioso sucede ya que vemos en la Figura 19 que solo la tenencia del producto en estudio en el mes anterior (lag_M1_ind_nom_pens_fin_ult1) se presenta como variable importante, sin embargo, esta no tiene una importancia significativa en comparación a la variable lag_M1_ind_cno_fin_ult1, que nos indica la tenencia del producto cuenta nómina en el mes anterior a la fecha. No obstante, el análisis de los valores SHAP en la Figura 21, podemos ver que la tenencia en el mes anterior del producto en estudio (lag_M1_ind_nom_pens_fin_ult1) tiene una fuerte relación directa con la variable target. Esto podría explicarse debido a que por la manera en que se ha determinado la importancia de las variables (Figura 19) esta no tiene en cuenta la interacción que pueda haber entre variables, sin embargo, SHAP (Figura 20) si es capaz de detectar estas interacciones, así como relaciones no lineales; por lo tanto, esta variable podría por su cuenta no ser relativamente importante, pero a la vez interactuar con variables más importantes que hagan que su contribución a la predicción sea significativa.

Figura 19. Feature importance ind_nom_pens_fin_ult1

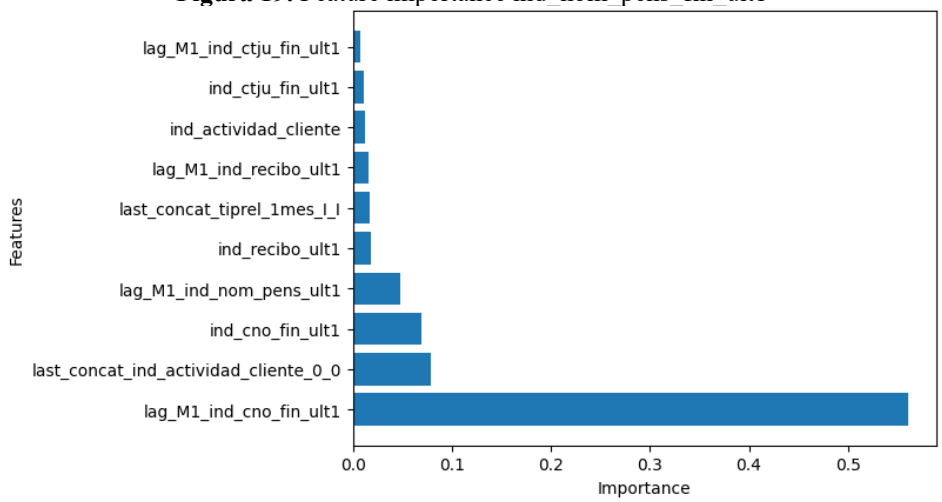
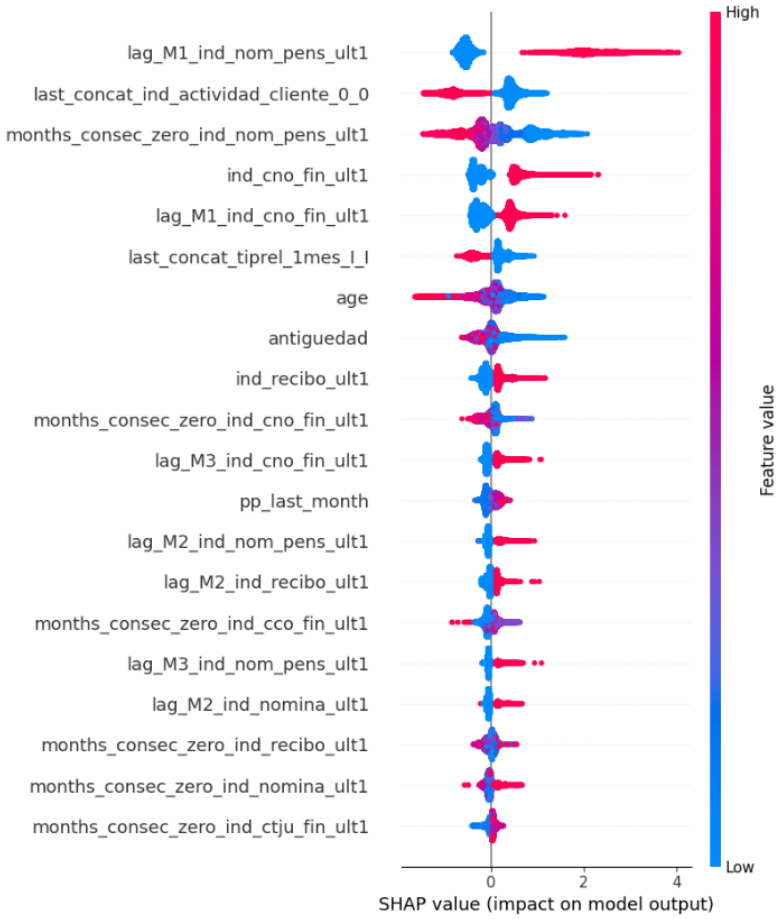


Figura 20. SHAP values ind_nom_pens_fin_ult1



5.2 Filtrado Colaborativo

5.2.1 Factorización Matricial Generalizada (GMF):

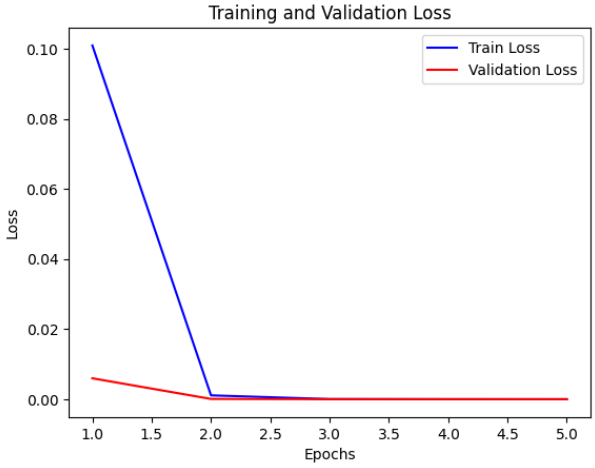
Se realizó iteró entre distintos valores de los hiperparametros *embedding size y learning rate* para encontrar la combinación optima de estos:

- *Embedding size*: [16, 32, 64]
- *Learning rate*: [0.001, 0.01, 0.1]

Finalmente se determinó que el modelo con el menor valor de la función de perdida en validación fue el que tenía como *embedding size* 16 y *learning rate* 0.001.

A continuación, en la Figura 21, podemos ver como varia la función de perdida en el conjunto de train y de validación en cada época de entrenamiento.

Figura 21. GMF: Disminución de la función de pérdida en cada época



Evaluación de las predicciones de cada producto

A continuación, procederemos a evaluar el recomendador utilizando el conjunto de *test*, el cual consiste en datos que el modelo no ha visto durante el entrenamiento.

En la Tabla 14, se presenta una comparativa de las precisiones y el *recall* obtenido para cada producto. Podemos ver que, la factorización matricial es capaz de identificar todas las interacciones positivas de todos los productos, incluso los que tienen una cantidad limitada de interacciones. No obstante, también podemos ver por lo general el modelo hace un mejor trabajo prediciendo las instancias que las negativas; por ello es que vemos la gran mayoría de productos a pesar de tener un alto *recall* tienen una precisión baja.

Tabla 14. Precisión productos GMF

Producto	Precision	Recall
ind_ahor_fin_ult1	0.01%	100%
ind_aval_fin_ult1	0.00%	100%
ind_cco_fin_ult1	77.69%	100%
ind_cder_fin_ult1	0.05%	100%
ind_cno_fin_ult1	10.33%	100%
ind_ctju_fin_ult1	1.26%	100%
ind_ctma_fin_ult1	1.01%	100%
ind_ctop_fin_ult1	17.20%	100%
ind_ctpp_fin_ult1	5.88%	100%
ind_deco_fin_ult1	0.08%	100%
ind_deme_fin_ult1	0.23%	100%
ind_dela_fin_ult1	5.79%	100%
ind_ecue_fin_ult1	10.85%	100%
ind_fond_fin_ult1	2.49%	100%
ind_hip_fin_ult1	0.81%	100%
ind_plan_fin_ult1	1.25%	100%
ind_pres_fin_ult1	0.32%	100%
ind_reca_fin_ult1	7.15%	100%
ind_tjcr_fin_ult1	6.23%	100%
ind_valo_fin_ult1	3.25%	100%
ind_viv_fin_ult1	0.54%	100%
ind_nomina_ult1	7.54%	100%
ind_nom_pens_ult1	8.33%	100%
ind_recibo_ult1	16.63%	100%

Evaluación del recomendador

En la tabla 15 se muestra el valor del MapK para diferentes valores de K, como 3, 5, 7 y 9, respectivamente. Contrariamente a los modelos anteriores, observamos que este resultado no disminuye a medida que aumenta K, sino que aumenta. Esta tendencia se explica por el hecho de que el modelo identifica de manera efectiva las instancias positivas en todos los casos y se enfoca en reducir la tasa de falsos negativos. Al lograr esto, el modelo minimiza la inclusión de recomendaciones irrelevantes en el conjunto de recomendaciones. En consecuencia, las recomendaciones proporcionadas tienen una mayor probabilidad de ser relevantes y de alta calidad, lo que a su vez mejora la precisión y, por ende, el valor del MapK.

Tabla 15. MapK GMF

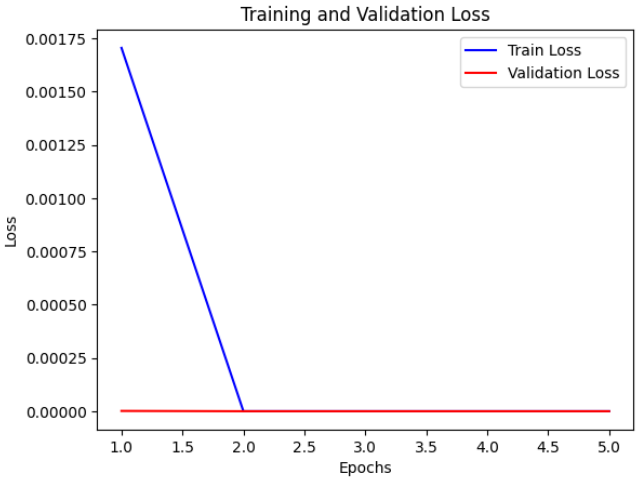
K	GMF
3	0.2297
5	0.2457
7	0.2695
9	0.2745

5.2.2 Factorización Matricial Neuronal desde cero (NMF):

En primer lugar, se entrena el modelo NMF desde cero, es decir con los componentes GMF y MLP sin haber encontrado cada uno de estos su combinación de hiperparametros optima.

A continuación, en la Figura 22 podemos ver como varia la función de perdida a lo largo de las épocas de entrenamiento.

Figura 22. NMF: Disminución de la función de pérdida en cada época



Evaluación de las predicciones de cada producto

A continuación, en la Tabla 16 podemos ver la precisión y el recall que obtiene el modelo para cada producto. Podemos ver que, los resultados son bastante parecidos a los que se obtienen en el modelo GMF; es decir el modelo es capaz de identificar todas las instancias positivas incluso las correspondientes a productos poco frecuentes, sin embargo, clasifica muchas instancias negativas como positivas. En otras palabras, el modelo ha sido capaz de reducir la cantidad de falsos negativos, a coste de aumentar la cantidad de falsos positivos.

Tabla 16. Precisión productos NMF desde cero

Producto	Precisión	Recall
ind_ahor_fin_ult1	0.03%	100%
ind_aval_fin_ult1	0.00%	100%
ind_cco_fin_ult1	82.14%	100%
ind_cder_fin_ult1	0.41%	100%
ind_cno_fin_ult1	36.77%	100%
ind_ctju_fin_ult1	1.26%	100%
ind_ctma_fin_ult1	5.94%	100%
ind_ctop_fin_ult1	48.9%	100%
ind_ctpp_fin_ult1	5.88%	100%
ind_deco_fin_ult1	0.1%	100%
ind_deme_fin_ult1	0.81%	100%
ind_dela_fin_ult1	8.12%	100%
ind_ecue_fin_ult1	10.85%	100%
ind_fond_fin_ult1	2.49%	100%
ind_hip_fin_ult1	0.81%	100%
ind_plan_fin_ult1	1.25%	100%
ind_pres_fin_ult1	0.32%	100%
ind_reca_fin_ult1	7.15%	100%
ind_tjcr_fin_ult1	60.55%	100%
ind_valo_fin_ult1	11.81%	100%
ind_viv_fin_ult1	0.93%	100%
ind_nomina_ult1	7.54%	100%
ind_nom_pens_ult1	8.33%	100%
ind_recibo_ult1	16.63%	100%

Evaluación del recomendador

No obstante, a pesar de que el modelo obtiene precisiones y *recalls* bastante similares a los que obtiene el modelo GMF, el MapK obtenido por este modelo es considerablemente más bajo (ver Tabla 17). Esto sucede ya que el modelo se encuentra sobreentrenado y predice de manera extrema las instancias de la clase positiva; es decir al visualizar las predicciones en vez de dar probabilidades distribuidas entre 0 y 1, da probabilidades bastante cercanas a 1. Por lo tanto, todo lo que el modelo va a predecir como instancia positiva va a tener una probabilidad casi igual a 1. Esto hace que, al momento de ordenar las probabilidades para recomendar ítems nuevos, el modelo mezcle instancias tanto verdaderas positivas como falsas positivas, haciendo al sistema más propenso a recomendar ítems menos relevantes en posiciones altas del ranking.

Tabla 17. MapK NMF desde cero

K	MapK
3	0.0072
5	0.0144
7	0.0192
9	0.0225

5.2.3 Factorización Matricial Neuronal (NMF) con modelos GMF y MLP pre entrenados: Entrenamiento de la MLP

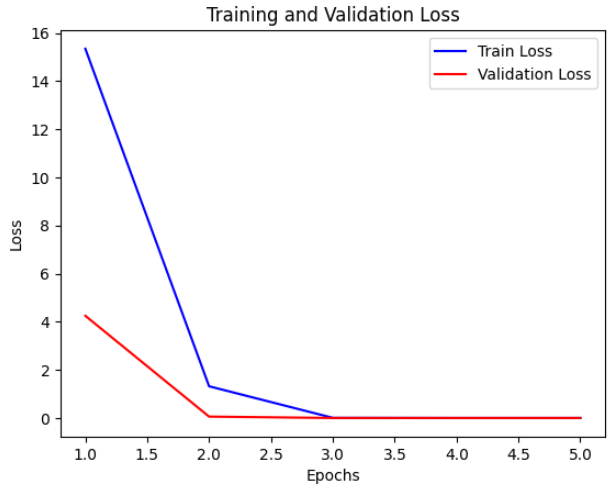
En primer lugar, con la finalidad de comprobar lo establecido por los autores He et al (2017) se realizó una búsqueda de los hiperparametros óptimos, al igual que se hizo con el modelo GMF:

- *Embedding size*: [16, 32, 64]
- *Learning rate*: [0.001, 0.01, 0.1]

Finalmente se determinó que el modelo con el menor valor de la función de perdida en validación fue el que tenía como *embedding size* 64 y *learning rate* 0.001.

A continuación, en la Figura 23 podemos ver como varia la función de perdida a lo largo de las épocas de entrenamiento para el mejor modelo MLP.

Figura 23. NMF pre-entrenado: Disminución de la función de pérdida en cada época



Ensamble NMF

A continuación, se toma el modelo GMF y MLP pre entrenados, y cada uno con su combinación optima de parámetros para ensamblar el modelo NMF. En las tablas 18 y 19 podemos ver los resultados del modelo. Notamos que, igual que los anteriores modelos construidos este es capaz de identificar todas las instancias

de la clase positiva, pero falla en algunos de la clase negativa dando resultados falsos positivos. No obstnate, comparando este modelo de ensamble con el GMF y el NMF armado desde cero, podemos ver que la cantidad de falsos positivos disminuyen en ciertos productos, especialmente en aquellos que cuentan con bastantes interacciones en el entrenamiento.

Tabla 18. Precisión productos NMF pre-entrenada

Producto	Precisión	Recall
ind_ahor_fin_ult1	0.03%	100%
ind_aval_fin_ult1	0.00%	100%
ind_cco_fin_ult1	88.73%	100%
ind_cder_fin_ult1	7.82%	100%
ind_cno_fin_ult1	44.06%	100%
ind_ctju_fin_ult1	2.54%	100%
ind_ctma_fin_ult1	7.88%	100%
ind_ctop_fin_ult1	52.4%	100%
ind_ctpp_fin_ult1	9.03%	100%
ind_deco_fin_ult1	0.19%	100%
ind_deme_fin_ult1	1.35%	100%
ind_dela_fin_ult1	12.24%	100%
ind_ecue_fin_ult1	16.60%	100%
ind_fond_fin_ult1	2.55%	100%
ind_hip_fin_ult1	0.83%	100%
ind_plan_fin_ult1	1.24%	100%
ind_pres_fin_ult1	0.76%	100%
ind_reca_fin_ult1	7.67%	100%
ind_tjcr_fin_ult1	86.74%	100%
ind_valo_fin_ult1	21.44%	100%
ind_viv_fin_ult1	0.93%	100%
ind_nomina_ult1	11.96%	100%
ind_nom_pens_ult1	15.31%	100%
ind_recibo_ult1	20.63%	100%

Finalmente, podemos ver que el valor obtenido para el MAPk es ligeramente superior a los obtenidos por los anteriores modelos.

Tabla 19. MapK NMF desde cero

K	MapK
3	0.2456
5	0.2598
7	0.2701
9	0.2794

Capítulo 6. Conclusiones, limitaciones y trabajo futuro

6.1 Conclusiones

- El desarrollo de un modelo recomendador de productos bancarios utilizando técnicas de Machine Learning y Deep Learning puede ser una iniciativa estratégica para las entidades financieras. Este modelo puede ayudar a mejorar la satisfacción del cliente y fortalecer las relaciones con el banco al ofrecer recomendaciones personalizadas y relevantes.
- La recopilación y análisis de datos relevantes, como historiales de transacciones, perfiles de clientes y características de productos bancarios, es fundamental para entrenar y ajustar los modelos que se vayan a probar para el sistema de recomendación. Estos datos proporcionan información valiosa sobre las preferencias y necesidades individuales de los clientes, lo que permite realizar recomendaciones más precisas.
- Durante la investigación, se exploraron diferentes algoritmos y técnicas de recomendación utilizados en el ámbito de la analítica de datos y el aprendizaje automático. Esto permitió evaluar su aplicabilidad y eficacia en el contexto de productos bancarios.
- El modelo híbrido, aunque presentó algunas limitaciones en términos de falta de datos para ciertos productos y resultó ser más costoso de construir, brindó una valiosa oportunidad para comprender el ciclo de vida de un proyecto de ciencia de datos que puede ser aplicado en entidades bancarias. Desde la etapa de construcción de la variable target y de la población base, hasta la ingeniería de variables, modelado e integración, este enfoque permitió adquirir una comprensión más profunda de los desafíos y procesos involucrados en el desarrollo de un sistema recomendador.
- Los métodos de filtrado colaborativo (GMF y NMF) por lo general hacen un buen trabajo identificando instancias de la clase positiva, no obstante, en este proceso suelen identificar incorrectamente muchas instancias de la clase negativa como positiva, por ello obtiene buenos resultados de recall pero pésimos resultados en la precisión.
- El método de factorización matricial neural utilizando redes neuronales de Keras demostró ser el enfoque más eficiente para la construcción del modelo recomendador, la cual gracias a combinar la linealidad de la factorización matricial con la no linealidad del perceptrón multicapa fue capaz de identificar interacciones incluso donde la informacion era limitada, logrando detectar todas las instancias de la clase positiva correctamente y disminuyendo los falsos positivos.
- Por otro lado, la factorización matricial neural la cual se construyó desde cero entrenando a los componentes de GMF y MLP en conjunto presenta resultados altamente sobre ajustados, mientras

que por otro lado el modelo que entrenó por separado los componentes GMF y MLP logró mejorar el rendimiento del modelo.

- Dicho esto, el tamaño del *embedding* y la tasa de aprendizaje son parámetros clave en un modelo; este primero afecta la capacidad del modelo para capturar relaciones complejas, mientras que el segundo influye en la velocidad y estabilidad del entrenamiento. Estos parámetros deben ajustarse específicamente para cada modelo, ya que diferentes algoritmos pueden requerir configuraciones personalizadas. Compartir los mismos valores en dos modelos no siempre es lo mejor, por lo que es necesario realizar experimentos y pruebas exhaustivas para encontrar los valores óptimos que maximicen el rendimiento del modelo en cada caso.
- La métrica utilizada para comparar los modelos recomendadores, MapK presenta varias limitaciones al medir el rendimiento de un sistema de recomendación, algunas de ellas siendo su sensibilidad a la posición de los elementos relevantes en el ranking, la falta de consideración de la diversidad de las recomendaciones, la falta de consideración de la relevancia de los elementos no recuperados y la falta de consideración de la temporalidad de las recomendaciones. Asimismo, esta métrica es bastante susceptible a usuarios con cero nuevos productos en el momento de la evaluación dado, ya que en este caso considera que la precisión promedio (*Average Precision at K*) de este usuario es 0; y esto puede tener un impacto significativo en el cálculo del MapK.
- Es por ello que, analizar el MapK por sí solo puede ser difícil. Pese a que un mayor valor de esta métrica siempre indica un mejor sistema de recomendacion, muchas veces se suelen obtener valores bajos y esto no significa un bajo rendimiento del sistema. Es por este motivo que, es recomendase recomienda comparar el MapK de nuestro sistema contra uno de un modelo base.

6.2 Limitaciones

- La principal limitación de este trabajo han sido los recursos computacionales. Se ha estado trabajando con millones de datos, por lo cual desarrollar el proyecto en un ordenador no era posible. Por lo cual, se optó por obtener una suscripción a Google Colab Pro, para poder tener acceso a los servidores de Google con mayores capacidades de memoria y unidades de procesamiento gráfico (GPU), principalmente para acelerar los procesos de ingeniería de variables y de entrenamiento de las redes neuronales. Sin embargo, las unidades de GPU pueden agotarse temporalmente debido a la alta demanda de los usuarios en el momento dado, retrasando el avance del proyecto y limitando las pruebas y resultados que se quisieron realizar.

- Asimismo, otra limitación que se tuvo en el desarrollo de este trabajo fue la falta de tiempo. Originalmente se tenía planeado que el trabajo fuera desarrollado en el horario de las prácticas profesionales, sin embargo, debido a que el proyecto fue externo a las labores realizadas en las prácticas con la empresa, tuvo que ser desarrollado en otros horarios y con recursos propios.

6.3 Recomendaciones y Trabajo Futuro

- En la construcción del modelo híbrido, se recomienda realizar un análisis de correlaciones previo a la construcción del modelo, así como limitar la cantidad de variables del modelo, de manera que logramos mejorar la interpretabilidad, estabilidad, eficiencia y capacidad de generalización de los modelos, evitando problemas asociados con la complejidad excesiva y la multicolinealidad tanto entre variables predictivas como entre variables predictivas y la target.
- Se recomienda analizar más a profundidad la situación de cada producto, así como si es relevante o no incluirlo en las recomendaciones, ya que como pudimos ver muchos de estos se encuentran en declive y a lo mejor computacionalmente es más costoso de lo que puede llegar a aportar la recomendación de ese producto.
- En la parte de filtrado colaborativo con redes neuronales, asegurarse de que en las interacciones los productos estén balanceados tanto en los conjuntos de train, como validación y test, con la finalidad de garantizar la generalización del modelo, evitar sesgos, permitir una evaluación justa y precisa, y mejorar la capacidad de personalización en las recomendaciones.
- En la construcción del modelo Factorización Matricial Neuronal, realizar una búsqueda más exhaustiva de la mejor combinación de hiperparametros, así como probar distintas técnicas para reducir el sobre entrenamiento de la red.
- Con respecto a los modelos de filtrado colaborativo basado en modelos (GMF y NMF), realizar pruebas con usuarios nuevos, con la finalidad de analizar el rendimiento de los modelos frente al problema de *cold-start*; asimismo, construir variaciones de estos modelos incluyendo características de usuarios y productos con la finalidad de disminuir el efecto de este problema, y comparar resultados.
- Para trabajos futuros, sería beneficioso considerar la utilización de *TensorFlow Recommenders* debido a las numerosas funcionalidades y beneficios que ofrece. Esta biblioteca proporciona modelos predefinidos, algoritmos de aprendizaje automático optimizados y herramientas de evaluación para el desarrollo de sistemas de recomendación. Al aprovechar estas características, se podrían mejorar los resultados obtenidos en este proyecto. *TensorFlow Recommenders* permite una

implementación más eficiente y efectiva de sistemas de recomendación, facilitando la representación de datos, ofreciendo modelos flexibles y permitiendo una evaluación precisa del rendimiento. Además, si se tuviera en mente construir un modelo con mira a la productivización, TensorFlow es una opción sólida y confiable para esto, ofreciendo escalabilidad, flexibilidad, un ecosistema activo y compatibilidad con hardware diverso.

- Finalmente, para trabajos futuros y a mira de una productivización del sistema recomendador de productos, se recomienda analizar la estabilidad de las predicciones del modelo en el tiempo, así como monitorear KPI's de importancia de forma periódica.

REFERENCIAS

Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer.

Banerjee, S. (2020). Keras documentation: Collaborative Filtering for Movie Recommendations. Retrieved June 18, 2023, from Keras.io website: https://keras.io/examples/structured_data/collaborative_filtering_movielens/

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web - WWW '17*. https://doi.org/10.1145/3038912.3052569

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). Recommender Systems: An Introduction. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511763113

Karatzoglou, A., & Hidasi, B. (2017). Deep learning for recommender systems. In Proceedings of the Eleventh ACM Conference on Recommender Systems (pp. 353-354). ACM.

Pathairush Seeda. (2021, October 13). A Complete Guide To Recommender Systems — Tutorial with Sklearn, Surprise, Keras, Recommenders. Retrieved June 18, 2023, from Medium website: https://towardsdatascience.com/a-complete-guide-to-recommender-system-tutorial-with-sklearn-surprise-keras-recommender-5e52e8ceace1

Rink, K. (2023, January 18). Mean Average Precision at K (MAP@K) clearly explained | Towards Data Science. Retrieved June 18, 2023, from Medium website: https://towardsdatascience.com/mean-average-precision-at-k-map-k-clearly-explained-538d8e032d2

Rocca, B. (2019, June 2). Introduction to recommender systems - Towards Data Science. Retrieved June 25, 2023, from Medium website: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

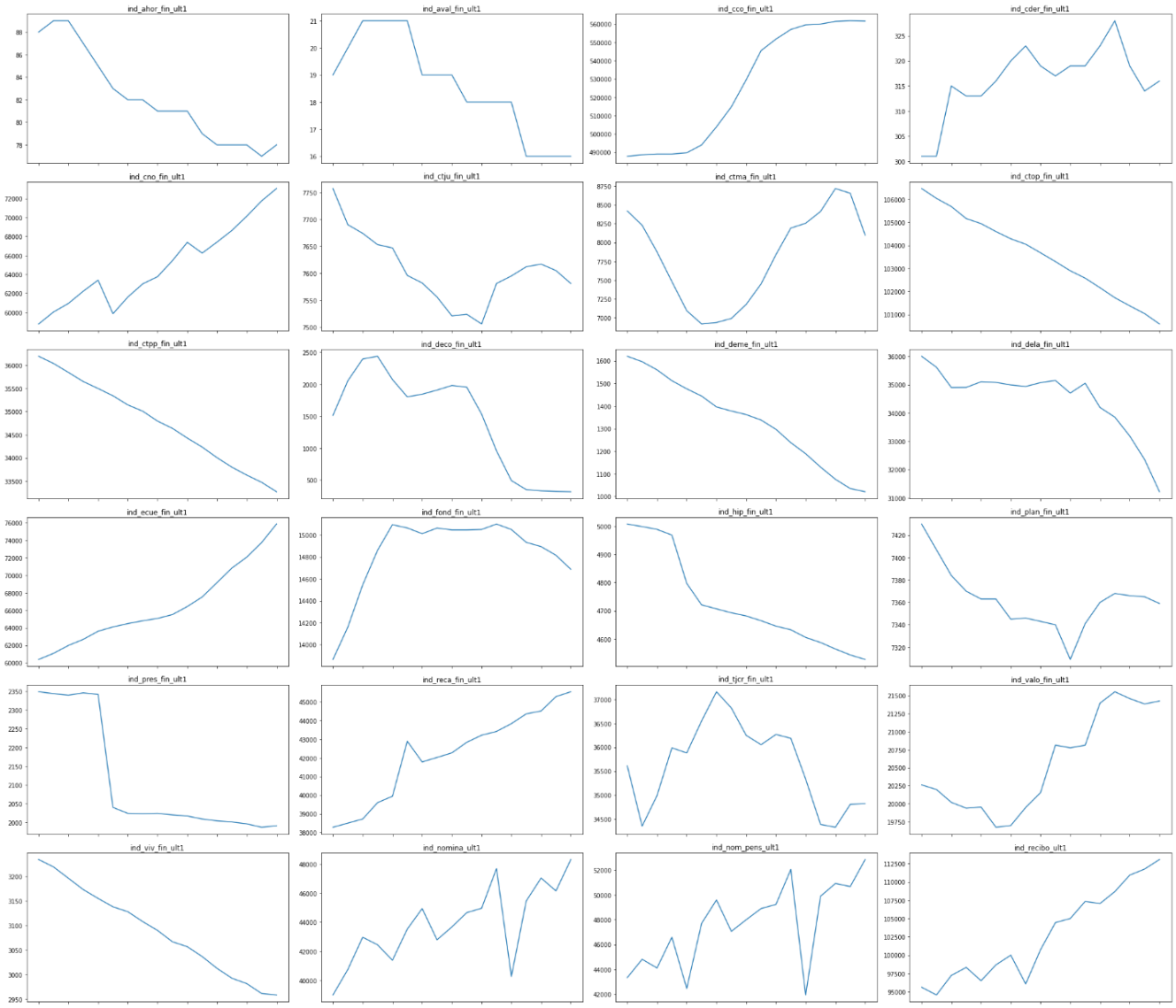
Santander Product Recommendation | Kaggle. (2023). Retrieved June 26, 2023, from Kaggle.com website: https://www.kaggle.com/competitions/santander-product-recommendation/data

Santander Product Recommendation - Tom Van de Wiele. (2016). Retrieved June 26, 2023, from Github.io website: https://ttvand.github.io/Second-place-in-the-Santander-product-Recommendation-Kaggle-competition/

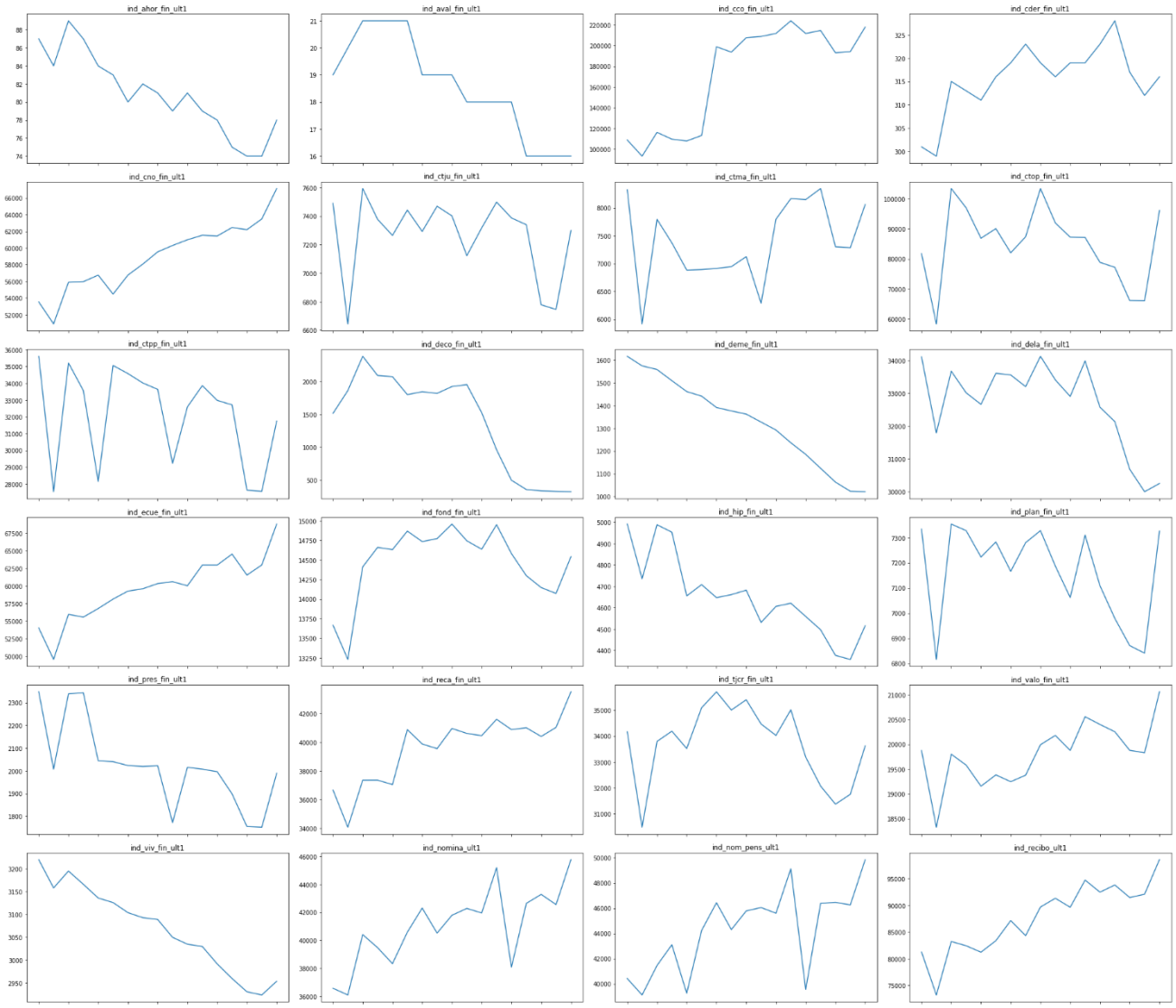
Zaragoza, J., & Cyberclick. (2023). Data Science: cómo crear un sistema de recomendación con machine learning. Retrieved June 26, 2023, from Cyberclick.es website: https://www.cyberclick.es/numerical-blog/data-science-como-crear-un-sistema-de-recomendacion-con-machine-learning

ANEXOS

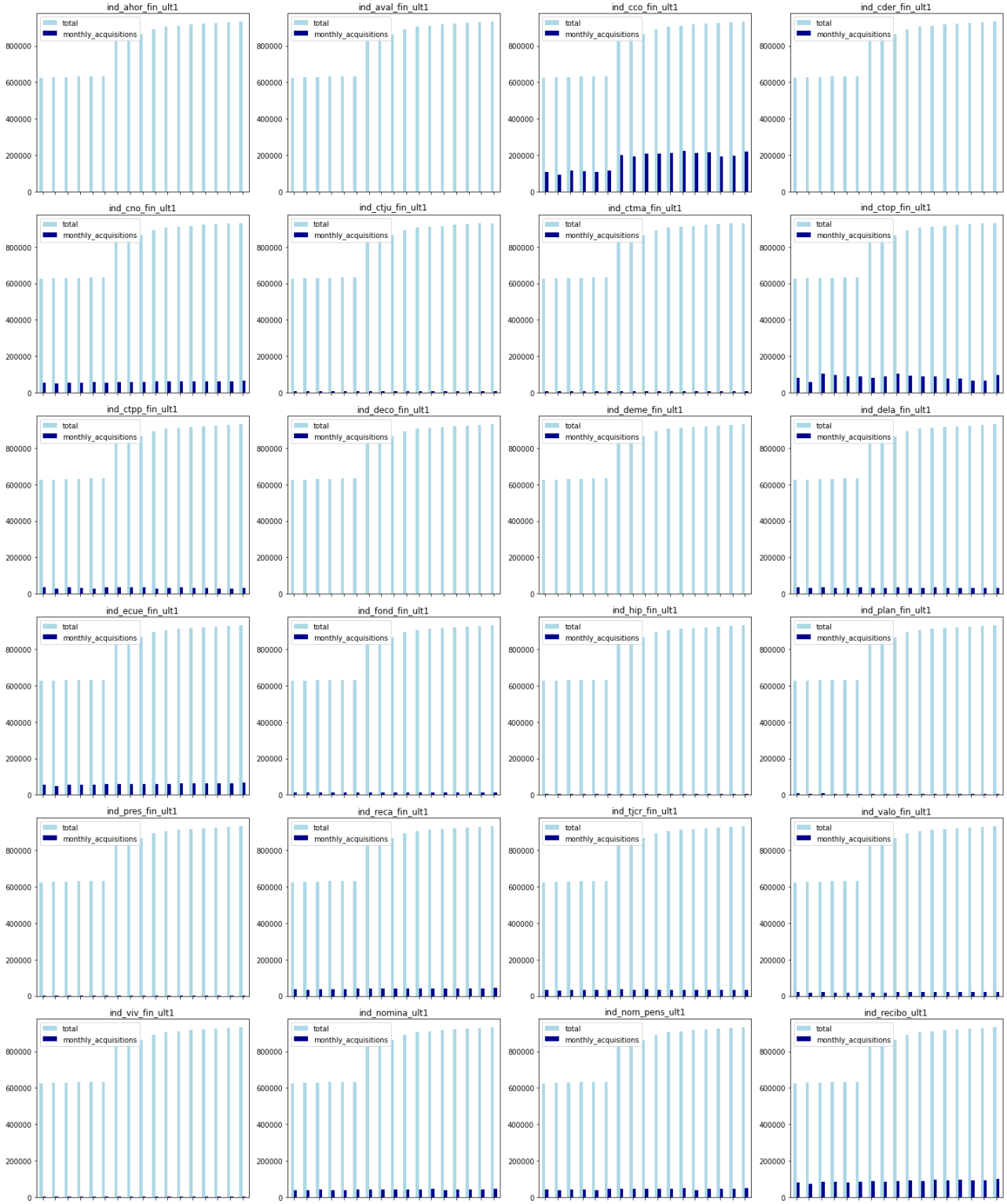
ANEXO 1: Evolución de la tenencia de productos



ANEXO 2: Evolución de la adquisición de productos



ANEXO 3: Proporción de la adquisición de productos



ANEXO 4: LIFT y LIFT Acumulado de productos