ASCE MAGAZINE ISSN: 3073-1178

Revista ASCE Magazine, Periodicidad: Trimestral Enero - Marzo, Volumen: 5, Número: 1, Año: 2026 páginas 2517 - 2541

Doi: https://doi.org/10.70577/asce.v5i1.712

Recibido: 2026-01-24
Aceptado: 2026-02-13
Publicado: 2026-03-13

Marketing Predictivo: Implementación de Sistemas de Recomendación con Aprendizaje Automático para la Identificación de Patrones de Consumo en Entornos Digitales

Predictive Marketing: Implementation of Machine Learning-Based Recommendation System for the Identification of Consumption Patterns in Digital Environments

Autores

Ricardo Rubén Mora Torosine [1]
rmorat2@uteq.edu.ec
https://orcid.org/0009-0004-9430-0465
Universidad Técnica Estatal de Quevedo
Ecuador

Mauricio Rubén Franco Coello[2]
mfrancoc2@uteq.edu.ec
https://orcid.org/0009-0005-2756-9429
Universidad Técnica Estatal de Quevedo
Ecuador

Verónica Alexandra Arrata Corzo[3]
varratac@uteq.edu.ec
https://orcid.org/0000-0002-8609-4831
Universidad Técnica Estatal de Quevedo
Ecuador

Harold Elbert Escobar Terán[4]
hescobar@uteq.edu.ec
https://orcid.org/0000-0001-9165-6627
Universidad Técnica Estatal de Quevedo
Ecuador

Cómo citar

Mora Torosine, R. R., Franco Coello, M. R., Arrata Corzo, V. A., & Escobar Terán, H. E. (2026). Marketing Predictivo: Implementación de Sistemas de Recomendación con Aprendizaje Automático para la Identificación de Patrones de Consumo en Entornos Digitales. *ASCE MAGAZINE*, *5*(1), 2517–2541.

Resumen

En respuesta al crecimiento acelerado de las necesidades comerciales, hay una creciente dependencia de extraer información significativa de grandes cantidades de datos crudos para impulsar estrategias de marketing predictivo capaces de anticipar las preferencias del consumidor. El presente artículo describe la implementación de un sistema de recomendación de películas basado en técnicas de aprendizaje automático supervisado, específicamente filtrado colaborativo mediante el algoritmo K-Nearest Neighbors (KNN) con similitud coseno y el Framework Apache Mahout en Python, aplicado sobre el conjunto de datos de películas obtenido de la base de datos Yahoo Research Webscope, que consta de dos archivos: Yahoo! Movies User Ratings y Yahoo! Descriptive Content Information, v1.0. Para lo cual, se analizaron patrones estadísticos de consumo y se generaron recomendaciones personalizadas coherentes con el historial del usuario. Los resultados demuestran que el filtrado colaborativo basado en ítems permite identificar con precisión patrones de consumo latentes, ofrece recomendaciones relevantes que pueden potenciar estrategias de fidelización y conversión en entornos digitales.

Palabras clave: Marketing Predictivo, Filtrado Colaborativo, Sistema de Recomendación, Machine Learning, Aprendizaje Superviado, Patrones de Consumo.

.

Abstract

In response to the accelerated growth of commercial needs, there is an increasing reliance on extracting meaningful information from large volumes of raw data to drive predictive marketing strategies capable of anticipating consumer preferences. This article describes the implementation of a movie recommendation system based on supervised machine learning techniques, specifically collaborative filtering using the K-Nearest Neighbors (KNN) algorithm with cosine similarity and the Apache Mahout framework in Python, applied to a movie dataset obtained from the Yahoo Research Webscope database, consisting of two files: Yahoo! Movies User Ratings and Yahoo! Descriptive Content Information, v1.0. To this end, statistical consumption patterns were analyzed and personalized recommendations consistent with the user's history were generated. The results demonstrate that item-based collaborative filtering accurately identifies latent consumption patterns and provides relevant recommendations that can enhance customer retention and conversion strategies in digital environments.

Keywords: Predictive Marketing, Collaborative Filtering, Recommendation System, Machine Learning, Supervised Learning, Consumption Patterns.

.

Introducción

En la era del comercio digital, la capacidad de anticipar y personalizar la experiencia del consumidor se ha convertido en un diferenciador competitivo fundamental. Para ello, el marketing predictivo, se entiende como el conjunto de técnicas analíticas orientadas a pronosticar el comportamiento futuro del consumidor a partir de datos históricos; en el cual, hoy en día representa uno de los campos de mayor crecimiento en la intersección entre la ciencia de datos y en el marketing. las grandes compañías como Google, Amazon y Netflix utilizan sistemas o motores de recomendación para maximizar las ventas y mejorar la experiencia del usuario. Acorde a (Sharma, Shaikh, & Li, 2021) Netflix informa que alrededor del 75–80 % de lo que se ve en la plataforma proviene de recomendaciones, y estima que la personalización y el sistema de recomendación aportan más de 1.000 millones de dólares al año en reducción de churn y aumento de engagement. Dada la gran cantidad de información disponible en Internet y el creciente número de usuarios, es cada vez más importante para las empresas proporcionar información relevante según las preferencias individuales, identificando patrones en el comportamiento de búsqueda y consumo.

Los sistemas de recomendación se han posicionado como una herramienta estratégica en el ecosistema digital moderno. Según (Deldjoo, Schedl, Hidasi, Wei, & He, 2022) los sistemas de recomendación han evolucionado considerablemente gracias a la integración de técnicas de aprendizaje profundo, siendo capaces de capturar representaciones latentes complejas del comportamiento del usuario. En este sentido, el poder de los datos y el aumento de las ventas para las empresas de comercio electrónico se aprovechan mediante la implementación de estos sistemas en sus plataformas digitales.

Para ello, los sistemas de recomendación funcionan encontrando patrones de datos en conjuntos de datos mediante el estudio de las elecciones y preferencias de los consumidores, y luego prediciendo los resultados que mejor se adaptan a los intereses del usuario. Para (Agrawal, 2021) los sistemas de recomendación son algoritmos que sugieren elementos relevantes a los usuarios, existen varios tipos de sistemas de recomendación, cada uno con sus propias características y aplicaciones. En este caso, el filtrado colaborativo se basa en recopilar y analizar datos sobre el comportamiento del usuario. Esto incluye las actividades en línea del usuario y predice lo que le gustará en función de la similitud con otros usuarios. A su vez, el filtrado basado en contenido se enfoca en la descripción de un producto y un perfil del usuario. De la misma manera, los sistemas de recomendación híbridos, combinan el filtrado colaborativo y el filtrado basado en

contenido para hacer recomendaciones. Cada uno de estos sistemas tiene sus propias ventajas y desventajas, y la elección del sistema a utilizar dependerá de la aplicación específica y los datos disponibles.

De la misma forma, investigaciones recientes, como la de (Zhang, Yao, Sun, & Tay, 2019), han extendido estas técnicas mediante arquitecturas de aprendizaje profundo, logrando mejoras sustanciales en precisión, aunque el enfoque clásico basado en similitud coseno y KNN sigue siendo la base de la mayoría de implementaciones en producción por su interpretabilidad y eficiencia. Así mismo, uno de los trabajos que se tomaron como referencia fue de (Wu, Garg, & Bhandary, 2018) quienes implementaron un sistema de recomendación de películas mediante filtrado colaborativo con Apache Mahout sobre el dataset Yahoo Research Webscope, centrándose exclusivamente en la dimensión técnica del algoritmo.

Por lo tanto, El presente artículo extiende ese trabajo incorporando tres contribuciones originales: (1) la integración del algoritmo KNN con similitud coseno como complemento al filtrado colaborativo clásico de Mahout, permitiendo análisis de similitudes item-a-item más granulares; (2) el análisis estadístico profundo de patrones de consumo orientado a la toma de decisiones de marketing, dimensión ausente en el trabajo de referencia; y (3) la validación multi-perfil de recomendaciones personalizadas que demuestra la adaptabilidad del sistema a distintos segmentos de consumidores digitales.

Materiales y métodos

1. Métodos utilizados

Como se muestra en la **figura 1**, el sistema de recomendación funciona como un proceso dinámico de aprendizaje continuo. Cada vez que el usuario interactúa con la plataforma ya sea calificando explícitamente una película o dejando rastros de comportamiento a través de su historial de consumo, dicha información se convierte en la base para generar nuevas recomendaciones. El motor, construido con filtrado colaborativo mediante Apache Mahout y el algoritmo KNN con similitud coseno, analiza estos datos para identificar patrones y sugerir contenidos alineados con los intereses del usuario.

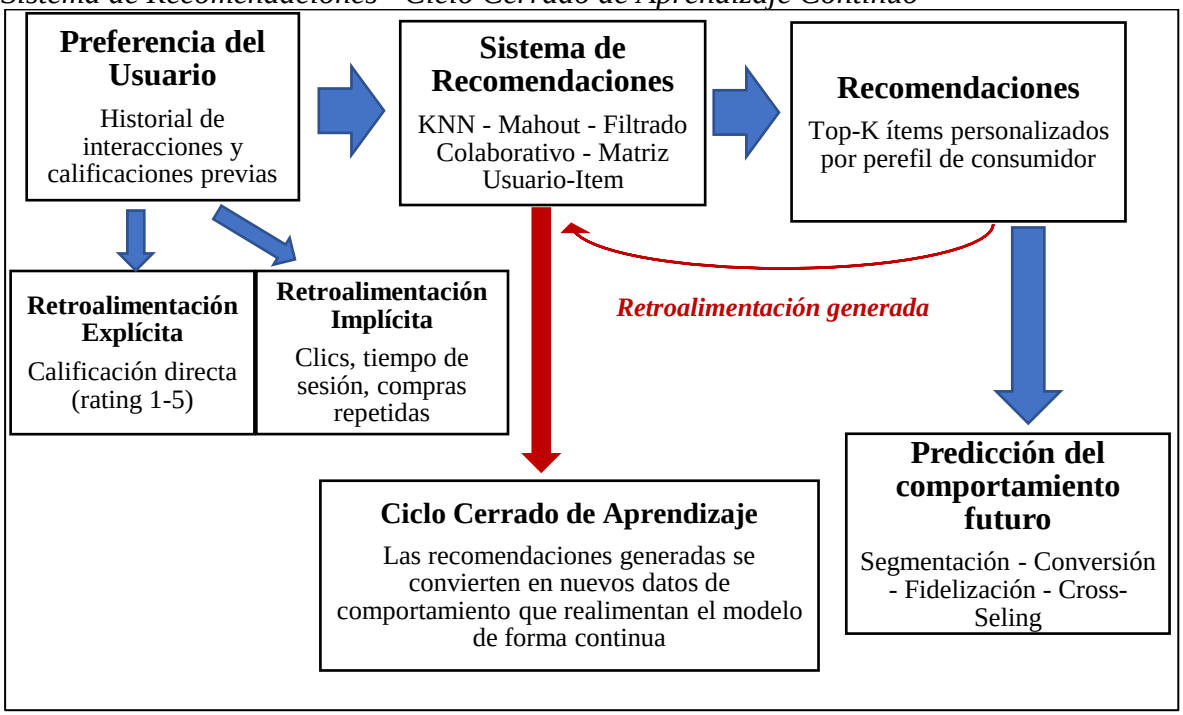
Sin embargo, las recomendaciones no representan el final del proceso, sino el inicio de una nueva fase de aprendizaje. Cada clic, valoración o visualización aporta nuevas señales que se incorporan nuevamente al sistema, actualizando la matriz usuario-ítem y ajustando los niveles

de similitud entre perfiles. De esta manera, el sistema evoluciona constantemente, refinando su capacidad de personalización con cada interacción (Fraihat, Shambour, Al-Betar, & Naser Makhadmeh, 2024).

Este ciclo de retroalimentación convierte la herramienta en un motor adaptativo, cuya precisión mejora progresivamente y que permite diseñar estrategias de marketing predictivo más efectivas, enfocadas en la fidelización, el aumento de la conversión y el impulso de ventas complementarias en entornos digitales; como lo sería la sugerencias de nuevas películas o series adaptadas a las preferencias del consumidor que el sistema a identificado, obteniendo así un mayor tiempo de retención del usuario en la plataforma.

Figura 1

Sistema de Recomendaciones - Ciclo Cerrado de Aprendizaje Continuo



Nota. Elaborado por autores. Basado de: (Yoo, Kang, & Tong, 2025)

1.1 Filtrado Colaborativo y Fundamento Matemático

El filtrado colaborativo (CF) opera sobre una matriz usuario-item M donde cada celda m_{ui} contiene la valoración del usuario u sobre el item i. La similitud entre dos usuarios x e y se calcula mediante la similitud coseno centrada, equivalente a la correlación de Pearson (Sgardelis, Margaris, Spiliotopoulos, & Vassilakis, 2025) :

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\,\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

.

Para lo cual, la predicción de valoración del usuario x sobre el item i se obtiene mediante:

$$\hat{r}_{xi} = \frac{\sum_{y \in N(x)} S_{xy} r_{yi}}{\sum_{y \in N(x)} |S_{xy}|}$$

En el caso del trabajo de (Wu, Garg, & Bhandary, 2018), ellos aplicaron este principio con el coeficiente de Pearson en Apache Mahout para filtrado basado en usuarios, y Log Likelihood Similarity para filtrado basado en items. El presente trabajo complementa esa implementación con KNN mediante similitud coseno directa sobre representaciones vectoriales dispersas, logrando mayor granularidad en el análisis de similitudes item-a-item.

1.1.1 Filtrado basado en usuarios

Las preferencias basadas en usuarios son comunes en el diseño de sistemas personalizados, asumiendo que las preferencias del usuario no son aleatorias al analizarlas históricamente. El proceso comienza con los usuarios dando calificaciones (1-5) a algunos elementos del catálogo. Estas calificaciones pueden ser explícitas o implícitas. Las calificaciones explícitas son cuando el usuario evalúa explícitamente el elemento en una escala o indica un pulgar hacia arriba/abajo. A menudo, las calificaciones explícitas son difíciles de obtener, y en estos casos, se recopilan calificaciones implícitas basadas en el comportamiento del usuario. Por ejemplo, si un usuario compra un producto más de una vez, indica una preferencia positiva. En el contexto de sistemas de películas, se puede inferir que, si un usuario ve toda la película, tiene cierta afinidad por ella. No hay reglas claras para determinar calificaciones implícitas.

Luego, para cada usuario, se encuentran cierto número definido de vecinos más cercanos. Se calcula la correlación entre las calificaciones de los usuarios utilizando el algoritmo de correlación de Pearson. La suposición es que, si las calificaciones de dos usuarios están altamente correlacionadas, entonces disfrutan de elementos y productos similares, lo que se utiliza para recomendar elementos a los usuarios.

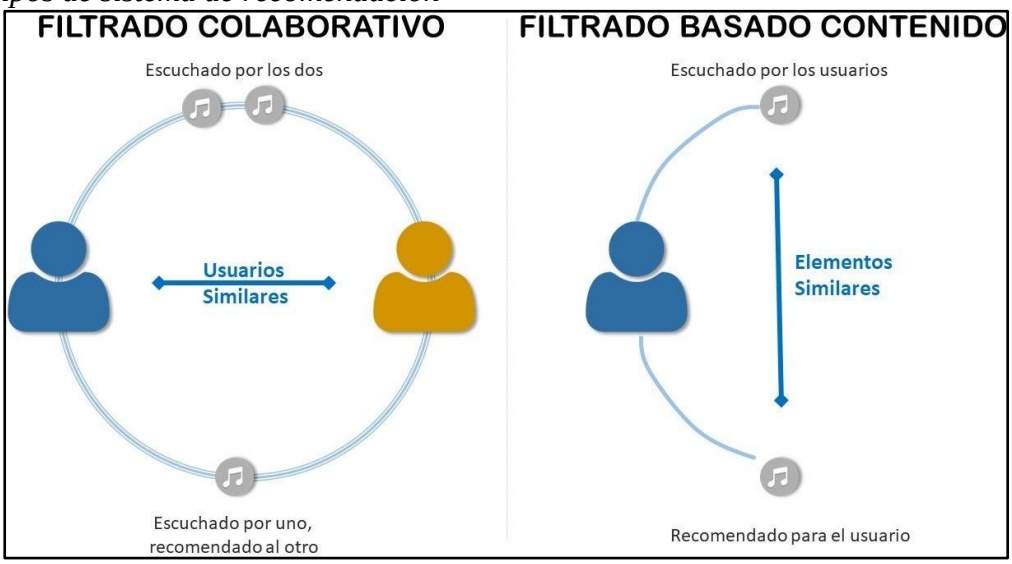
1.1.2 Filtrado basado en ítems:

A diferencia del método de filtrado basado en usuarios, el filtrado basado en ítems se centra en la similitud entre los elementos que gustan a los usuarios en lugar de los usuarios mismos. Los elementos más similares se calculan previamente. Luego, para realizar recomendaciones, se sugieren al usuario los elementos más similares al ítem objetivo.

La **Figura 2** explica aún más los dos tipos de sistemas de recomendación colaborativa. Este tipo de filtrado a menudo se denomina Filtrado Colaborativo Puro.

Figura 2

Tipos de sistema de recomendación



Fuente: (Jerez G, 2023)

Con respecto al trabajo de (Wu, Garg, & Bhandary, 2018) el cual presenta un sistema técnico sin aplicación al marketing. Por el que, el presente artículo remarca el problema como herramienta de marketing predictivo, analizando patrones de demanda por género, segmentación de audiencias y estrategias de fidelizacion en e-commerce. De la misma manera, el trabajo de referencia utiliza exclusivamente Mahout con Pearson y Log Likelihood. Por otra parte, este articulo incorpora NearestNeighbors de Scikit-learn con similitud coseno sobre matrices dispersas CSR, posibilitando análisis de similitud película-a-película más granular; mientras que la investigación referencia en mención, las evalúan cualitativamente con una tabla genérica. Por lo tanto, este articulo valida el sistema sobre múltiples perfiles de usuario distintos, demostrando la variabilidad y adaptabilidad de las recomendaciones según el historial individual de cada consumidor.

2. Entorno Tecnológico Utilizado

La implementación de los datos para su posterior análisis, se desarrolló en Python 3.12.0 sobre Anaconda, complementada con Apache Mahout para el procesamiento distribuido, dada su reproducibilidad y capacidad de escalar a entornos empresariales, como se presenta en la siguiente tabla:

Tabla 1
Software y librerías utilizadas en el entorno tecnológico

Software / librería	Version	Utilización
Python	3.12.0	Anaconda (procesamiento principal)
NumPy	1.26.2	Operaciones matriciales
Pandas	2.1.4	Manipulación de DataFrames
SciPy	1.11.4	Matriz dispersa CSR (csr_matrix)
Scikit-learn	1.3.2	Algoritmo KNN (NearestNeighbors)
Matplotlib	3.8.2	Visualización estadística
Seaborn	0.13.0	Análisis visual de distribuciones
Apache Mahout	s.f.	Filtrado colaborativo distribuido

2.1 Mahout

Apache Mahout es una biblioteca científica y altamente analítica basada en la Fundación Apache Software. Su objetivo es producir implementaciones gratuitas, distribuidas y escalables de algoritmos avanzados de aprendizaje automático utilizados en áreas como clustering, clasificación, filtrado colaborativo y coincidencia de patrones frecuentes. Muchas de las implementaciones utilizan la plataforma Apache Hadoop. Incluye algoritmos poderosos como Similaridad de Máxima Verosimilitud, Coeficiente de Pearson, Similaridad Coseno, entre otros. Aunque Mahout aún no está completamente desarrollado, sigue creciendo y ofrece una opción completa para incorporar aprendizaje automático en Big Data, gestionado por la plataforma subyacente de Hadoop.

Por lo que, lo largo de los años, se han desarrollado muchos sistemas de recomendación utilizando métodos de filtrado colaborativo, basado en contenido o híbridos. Estos sistemas se han implementado utilizando varios algoritmos de Big Data y aprendizaje automático. Acorde a investigaciones realizadas a través del uso de (Apache Mahout, 2018), se observan que el filtrado colaborativo es una de las aproximaciones más utilizadas para construir sistemas de recomendación. Muchos de estos sistemas emplean algoritmos de aprendizaje automático como el Clustering mediante K-Means, redes neuronales, entre otros, para recomendar elementos.

3. Conjunto de datos

El conjunto de datos utilizado fue obtenido de la base de datos Yahoo Research Webscope que consta de dos archivos: “Yahoo! Movies User Ratings” y “Yahoo! Descriptive Content Information, v1.0” (Yahooresearch, 2014). El primero contiene 211231 registros con ID de

usuario, ID de película y calificaciones. El segundo contiene 54058 registros con ID de película, título, género, directores, actores, entre otros. A su vez, se aplicó una limpieza en los datos dentro del archivo “Movies Descriptive Content Information” el cual tenía alrededor de 40 columnas, muchas de las cuales no eran necesarias para este análisis y fueron eliminadas. De la misma manera, el conjunto de datos también tenía valores en blanco y duplicados que necesitaban ser resueltos. Además, se eliminaron las entradas para películas en el archivo “Movies Users Ratings” que no correspondían a ninguna película en el archivo “Movies Descriptive Content Information”, esto con la finalidad de facilitar el procesamiento.

Figura 3

Conjunto de datos Yahoo Research Webscope tras integración y limpieza

```
In [2]: #carga del conjunto de datos de calificación
        ratings = pd.read_csv("https://s3-us-west-2.amazonaws.com/recommender-tutorial/ratings.csv")
        ratings.head()
        # cargar conjunto de datos de películas
        movies = pd.read_csv("https://s3-us-west-2.amazonaws.com/recommender-tutorial/movies.csv")
        movies.head()
        ratings = ratings.merge(movies, how="left", on="movieId").drop_duplicates()
        print("Las dimensiones del df de estudio son: ", ratings.shape)
        ratings

Las dimensiones del df de estudio son:  (100836, 6)
```

Out[2]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	3	4.0	964981247	Grumpier Old Men (1995)	Comedy|Romance
2	1	6	4.0	964982224	Heat (1995)	Action|Crime|Thriller
3	1	47	5.0	964983815	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
4	1	50	5.0	964982931	Usual Suspects, The (1995)	Crime|Mystery|Thriller
...	...	...	...	...	...	...
100831	610	166534	4.0	1493848402	Split (2017)	Drama|Horror|Thriller
100832	610	168248	5.0	1493850091	John Wick: Chapter Two (2017)	Action|Crime|Thriller
100833	610	168250	5.0	1494273047	Get Out (2017)	Horror
100834	610	168252	5.0	1493846352	Logan (2017)	Action|Sci-Fi
100835	610	170875	3.0	1493846415	The Fate of the Furious (2017)	Action|Crime|Drama|Thriller

100836 rows × 6 columns

En la **figura 3** la variable "rating" hace referencia a la calificación dada por un usuario a una película en particular. Estas calificaciones suelen ser números que representan la opinión subjetiva del usuario sobre la película, indicando qué tan bien o mal la perciben. Por lo general, las calificaciones pueden estar en una escala numérica, por ejemplo: En una escala de 1 a 5, donde 1 puede representar una baja satisfacción y 5 una alta satisfacción. O también en una escala de 1 a 10, donde 1 podría ser una calificación muy baja y 10 la más alta.

4. Construccion del Modelo

4.1 Matriz Usuario-Item

Para la construcción del modelo se utilizó Framework Mahout para construir el sistema de recomendación. Así mismo, para el filtrado basado en usuarios, se utilizó la clase Similaridad del Usuario junto con el coeficiente de Correlación de Pearson, para determinar la similitud entre las calificaciones de los usuarios y, por lo tanto, las preferencias. De esta manera, se construyó una matriz usuario-item dispersa mediante csr_matrix de SciPy, con forma (M películas x N usuarios). Los mapeos bidireccionales entre IDs reales e índices matriciales permiten recuperar los identificadores originales tras el cálculo de similitudes. La representación dispersa es crítica dada la baja densidad inherente a los datasets de recomendación.

Figura 4

Creación de la matriz: Usuario-Artículo

```
def create_matrix(df):
    N = len(df["userId"].unique())    # Usuarios únicos
    M = len(df["movieId"].unique())   # Películas únicas

    user_mapper      = dict(zip(np.unique(df["userId"]),  range(N)))
    movie_mapper     = dict(zip(np.unique(df["movieId"]), range(M)))
    user_inv_mapper  = dict(zip(range(N), np.unique(df["userId"])))
    movie_inv_mapper = dict(zip(range(M), np.unique(df["movieId"])))

    user_index  = [user_mapper[i]  for i in df["userId"]]
    movie_index = [movie_mapper[i] for i in df["movieId"]]

    X = csr_matrix((df["rating"], (movie_index, user_index)), shape=(M, N))
    return X, user_mapper, movie_mapper, user_inv_mapper, movie_inv_mapper

    X, user_mapper, movie_mapper, user_inv_mapper, movie_inv_mapper  =  create_matrix (ratings)
```

Una matriz usuario-elemento es una estructura de datos básica en los sistemas de recomendación. En ella, las filas representan usuarios, las columnas ítems, y las celdas las interacciones; por ejemplo: ratings, clics, y compra, acorde a (Ramlatchan, Yang, Liu, & Li, 2018). En el cual sus funcionamientos y resultados obtenidos son los siguientes:

1. Para averiguar el número de usuarios únicos y de vídeos únicos del conjunto de datos, se calculan **N** y **M**.
2. Se producen cuatro diccionarios:

- **user_mapper**: Asigna distintos ID de usuario a índices (el ID de usuario 1 se convierte en el índice 0, por ejemplo).
- **movie_mapper:** Convierte los ID de película distintos en índices (el ID de película 1 se convierte en el índice 0, por ejemplo).
- **user_inv_mapper:** Invierte user_mapper y vuelve a asignar índices a ID de usuario.
- **movie_inv_mapper:** Invierte movie_mapper asignando índices a IDs de películas.

3. Para asignar los ID reales de usuario y película del conjunto de datos a sus índices correspondientes, se generan las listas **user_index** y **movie_index**.
4. Se crea una matriz dispersa **X** utilizando la función csr_matrix de SciPy. Para generar esta matriz se utilizan los índices de usuario y película que corresponden a los valores de calificación en el conjunto de datos. La forma de la misma es (**M, N**), donde **M** denota la cantidad de películas distintas y **N** denota la cantidad de consumidores distintos.

Dicho de otro modo, el código presentado en la **figura 4**, facilita la realización de cálculos y la creación de sistemas de recomendación basados en la representación estructurada de las valoraciones de los usuarios sobre las películas en los datos.

4.2 KNN con Similitud Coseno

Posteriormente, se implementó el filtrado basado en usuarios mediante UserSimilarity de Mahout con el coeficiente de Pearson, y el filtrado basado en ítems mediante ItemSimilarity con Log Likelihood Similarity. Los resultados de ítems se almacenan en HDFS para acceso escalable. A su vez, se implementó el algoritmo KNN con similitud coseno sobre la matriz dispersa mediante NearestNeighbors de Scikit-learn:

Figura 5

Creación de la matriz: Usuario-Artículo

```
def find_similar_movies(movie_id, X, k, metric="cosine", show_distance=False):
    movie_ind = movie_mapper[movie_id]
    movie_vec = X[movie_ind]
    k += 1  # +1 para excluir la propia película

    kNN = NearestNeighbors(n_neighbors=k, algorithm="brute", metric=metric)
    kNN.fit(X)
    movie_vec = movie_vec.reshape(1, -1)
    neighbour = kNN.kneighbors(movie_vec, return_distance=show_distance)

    neighbour_ids = [movie_inv_mapper[neighbour.item(i)] for i in range(k)]
    neighbour_ids.pop(0)  # excluir película de consulta
```

.

```
    return neighbour_ids
```

4.3 Recomendación Personalizada

La función **recommend_movies_for_user()** integra el análisis KNN con el historial del usuario, identificando la película mejor calificada y generando recomendaciones basadas en las peliculas más similares. Para lo cual, implementa el paradigma de marketing one-to-one donde cada consumidor recibe una propuesta diferenciada (Aggarwal, 2016).

Figura 6

Recomendación de películas en función de las preferencias del usuario

```
def recommend_movies_for_user(user_id, X, user_mapper, movie_mapper,
                              movie_inv_mapper, k=10):
    df1 = ratings[ratings["userId"] == user_id]
    if df1.empty:
        print(f"Usuario {user_id} no encontrado."); return

    # Ancla: pelicula mejor calificada por el usuario
    movie_id = df1[df1["rating"] == max(df1["rating"])]["movieId"].iloc[0]
    movie_titles = dict(zip(movies["movieId"], movies["title"]))
    similar_ids = find_similar_movies(movie_id, X, k)

    print(f"Desde que viste {movie_titles[movie_id]}, tambien te puede gustar:")
    for i in similar_ids:
        print(movie_titles.get(i, "No encontrada"))
```

Como se muestra en la **figura 6**, la función **recommend_movies_for_user** fue diseñada para generar recomendaciones personalizadas a partir de un enfoque basado en similitud entre ítems. Para su ejecución, recibe como parámetros los diccionarios **user_mapper, movie_mapper** y **movie_inv_mapper,** los cuales permiten mapear los identificadores originales de usuarios y películas a los índices correspondientes en la matriz usuario-ítem $\boldsymbol{X}$, además del identificador

del usuario objetivo (user_id) y un parámetro opcional k que determina el número de recomendaciones a generar (por defecto, 10). El procedimiento inicia filtrando el conjunto de datos de valoraciones para recuperar las calificaciones asociadas al usuario especificado; en caso de no encontrarse registros, el sistema detiene la ejecución e informa la inexistencia del usuario en la base de datos. Posteriormente, se identifica la película con mayor valoración otorgada por dicho usuario, utilizándola como punto de referencia para la recomendación. A partir de este ítem ancla, se invoca la función **find_similar_movies**, la cual emplea un enfoque de **k-Nearest Neighbors (k-NN)** para localizar películas con patrones de similitud cercanos, presumiblemente calculados mediante métricas como la similitud coseno.

Finalmente, el sistema presenta el título de la película mejor valorada y el conjunto de títulos recomendados. En caso de que alguna película no pueda ser localizada en el repositorio de metadatos, se genera un mensaje de advertencia indicando su ausencia. Este procedimiento permite estructurar un esquema de recomendación centrado en la preferencia explícita del usuario, optimizando la coherencia de las sugerencias generadas.

Resultados y discusión

Como primeros resultados se realizó un análisis descriptivo de los datos para identificar en primeros indicios sobre el comportamiento del consumidor, orientado a la toma de decisiones de marketing predictivo. Los patrones identificados son estratégicamente relevantes para la gestión de recomendaciones de los catálogos digitales y el diseño de campanas personalizadas.

Figura 7

Recomendación de películas en función de las preferencias del usuario

```
n_ratings = len(ratings)
n_movies = len(ratings['movieId'].unique())
n_users = len(ratings['userId'].unique())
print(f"Número de valoraciones: {n_ratings}")
print(f"Número de movieId's: {n_movies}")
print(f"Número de usuarios únicos: {n_users}")
print(f"Promedio de Valoración por usuario: {round(n_ratings/n_users, 2)}")
print(f"Promedio de Valoración por película: {round(n_ratings/n_movies, 2)}")
```

Número de valoraciones: 100836
Número de movieId's: 9724
Número de usuarios únicos: 610
Promedio de Valoración por usuario: 165.3
Promedio de Valoración por película: 10.37

Como se evidencia en la **figura 7**, El conjunto de datos analizado contiene 100.836 valoraciones realizadas por 610 usuarios sobre 9.724 películas, lo que muestra una base bastante amplia en términos de catálogo y participación. En promedio, cada usuario ha realizado alrededor de 165 valoraciones, lo que indica un nivel de actividad considerable y aporta información suficiente para identificar patrones de preferencia individuales. Por otro lado, cada película recibe en promedio unas 10 valoraciones, lo que sugiere que, aunque existe una buena cantidad total de datos, no todas las películas cuentan con el mismo nivel de interacción. Por lo que, esta diferencia puede generar cierta dispersión en la matriz usuario-ítem, algo común en sistemas de recomendación, pero que igualmente permite trabajar con modelos colaborativos siempre que se gestionen adecuadamente los posibles efectos de baja densidad en algunos ítems.

Figura 8

Distribución de calificaciones en el conjunto de datos Yahoo Webscope

```
from great_tables.data import gtcars
from great_tables import md, html
user_freq = ratings[['userId', 'movieId']].groupby('userId').count().reset_index()
user_freq.columns = ['userId', 'n_ratings']
# Sort the DataFrame by the number of ratings in descending order
user_freq = user_freq.sort_values("n_ratings", ascending=False).head(5)
# Create a display table showing the table tailor-made for examples: exibble
gt_tbl = user_freq[['userId', 'n_ratings']]
(
GT(gt_tbl)
.tab_header(title=html("<strong>Top 5 de Ratings</strong>"),
subtitle=html("Por usuario"),)
.tab_source_note(source_note="Fuente: Netflix")
)
```

Top 5 de Ratings

Por usuario

userId	n_ratings
414	2698
599	2478
474	2108
448	1864
274	1346

Fuente: Netflix

```
query_1 = ratings[ratings.userId==414].sort_values(["rating","timestamp"], ascending=False)
query_1.head(5)
```

.

	userId	movieId	rating	timestamp	title	genres
64889	414	106918	5.0	1522624529	Secret Life of Walter Mitty, The (2013)	Adventure|Comedy|Drama
62975	414	2019	5.0	1519593885	Seven Samurai (Shichinin no samurai) (1954)	Action|Adventure|Drama
64969	414	170705	5.0	1519593882	Band of Brothers (2001)	Action|Drama|War
64931	414	139385	5.0	1519593286	The Revenant (2015)	Adventure|Drama
64923	414	127108	5.0	1519593248	Brooklyn (2015)	Drama|Romance

Acorde a la **figura 8**, el análisis muestra que el usuario 414 es el más activo dentro del conjunto de datos, con 2.698 valoraciones registradas, seguido por otros usuarios con niveles también elevados de participación. Esta concentración de actividad en ciertos perfiles sugiere la presencia de usuarios altamente comprometidos, cuyos historiales pueden aportar información especialmente valiosa para el entrenamiento de modelos de recomendación, ya que ofrecen patrones de consumo más definidos y consistentes. De la misma forma, al revisar las cinco películas mejor calificadas por el usuario 414, todas con puntuación máxima (5.0), se observa una preferencia marcada por géneros como drama, aventura y acción, incluyendo producciones tanto clásicas como contemporáneas. Este patrón evidencia que el usuario mantiene criterios de valoración relativamente exigentes pero coherentes, lo que facilita la identificación de similitudes con otros perfiles y mejora la precisión potencial de las recomendaciones personalizadas.

Figura 9

Valoración del ranking de películas de mejor a peor calificadas

```
# Encuentre las películas mejor y peor valoradas:
mean_rating = ratings.groupby('movieId')[['rating']].mean()
print("================================================================================")
# Películas peor valoradas
lowest_rated = mean_rating['rating'].idxmin()
print("Película peor valorada:\n", movies.loc[movies['movieId'] == lowest_rated])
print("================================================================================")
# Películas mejor valoradas
highest_rated = mean_rating['rating'].idxmax()
print("Película mejor valorada:\n", movies.loc[movies['movieId'] == highest_rated])
print("================================================================================")
```

.

```
================================================================================
Película peor valorada:
       movieId         title   genres
2689      3604  Gypsy (1962)  Musical
================================================================================
Película mejor valorada:
     movieId            title           genres
48        53  Lamerica (1994)  Adventure|Drama
================================================================================
```

Top de las 10 Películas mejor puntuadas:

```
reporte_1 = ratings.groupby(["title"]).sum(["rating"]).reset_index().sort_values("rating",
ascending=False).head(10)
reporte_1
```

	title	userId	movieId	rating	timestamp
7593	Shawshank Redemption, The (1994)	95829	100806	1404.0	376924839127
3158	Forrest Gump (1994)	101385	117124	1370.0	386165236681
6865	Pulp Fiction (1994)	90621	90872	1288.5	349204311001
5512	Matrix, The (1999)	85236	714738	1165.5	350270041779
7680	Silence of the Lambs, The (1991)	85535	165447	1161.0	320035674330
8001	Star Wars: Episode IV - A New Hope (1977)	76484	65260	1062.0	298509531263
1337	Braveheart (1995)	73375	26070	955.5	259448308280
3011	Fight Club (1999)	69737	645062	931.5	282265436424
7421	Schindler's List (1993)	69481	115940	929.5	257743980936
4662	Jurassic Park (1993)	74065	114240	892.5	257876752969

Dentro de la **figura 9** se realizó el análisis de las valoraciones promedio por película permite identificar tanto el extremo inferior como el superior en términos de percepción general de los usuarios. En este caso, Gypsy (1962) aparece como la película con menor calificación promedio, mientras que Lamerica (1994) destaca como la mejor valorada. Este contraste evidencia cómo, dentro de un mismo conjunto de datos, pueden coexistir obras con niveles de aceptación muy distintos, lo que refleja la diversidad de gustos y criterios de evaluación presentes en la comunidad de usuarios. Así mismo, se puede observar el ranking de las diez películas con mayor puntuación acumulada, se aprecia la predominancia de títulos ampliamente reconocidos como The Shawshank Redemption, Forrest Gump y Pulp Fiction, lo que sugiere que producciones con alto impacto cultural tienden a concentrar tanto un mayor número de valoraciones como calificaciones elevadas. No obstante, al tratarse de una suma total de ratings

y no de un promedio, estos resultados también pueden estar influenciados por la popularidad y el volumen de interacciones, más que exclusivamente por la calidad percibida.

También se lo puede visualizar en formato de gráfico de barras y segmentarlo por el tipo de género de peliculas entre las más y menos vistas por los usuarios de la plataforma, como se evidencia en la **figura 10.** Este tipo, de análisis ayuda a las estrategias del marketing en identificar ya sea por una mayor segmentación geográfica o incluso por temporadas las categorías más visualizadas, y de esa manera enfocar campañas publicitarias a través del sistema de recomendación peliculas y series a fines de los usuarios segmentados, provocando una mayor retención del cliente en la plataforma.

Figura 10

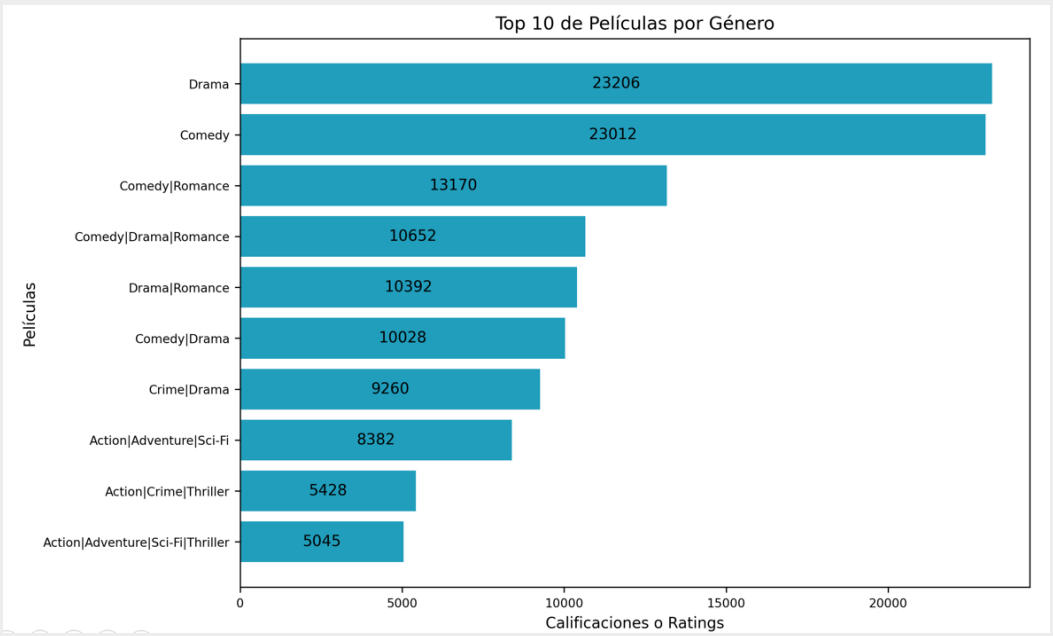
Ranking de las diez películas categorizadas por Género

```
def plot_horizontal_barh(df, variable_x, variable_y, title='Gráfico', xlabel=None, ylabel=None,
save_path=None):
plt.figure(figsize=(10, 6))
bars = plt.barh(df[variable_x], df[variable_y], color='#219ebc')
for bar in bars:
plt.text(bar.get_width() / 2, bar.get_y() + bar.get_height() / 2,
f'{bar.get_width():.0f}',
va='center', ha='center', fontsize=10, color='black')
plt.xlabel(xlabel if xlabel else variable_x)
plt.ylabel(ylabel if ylabel else variable_y)
plt.title(title)
plt.gca().invert_yaxis()
plt.tight_layout()
plt.tick_params(axis='x', which='both', labelsize=8)
plt.tick_params(axis='y', which='both', labelsize=8)
plt.figtext(0.1, 0.01, 'Elaborado por autores', ha='left', va='bottom', fontsize=10)
if save_path:
plt.savefig(save_path, dpi=300, bbox_inches='tight')
# print(f"Figura guardada en: {save_path}")
plt.show()

reporte_2 = ratings.groupby(["genres"]).sum(["rating"]).reset_index().sort_values("rating",
ascending=False).head(10)

plot_horizontal_barh(reporte_2,
'genres',
'rating',
title='Top 10 de Películas por Género',
xlabel='Calificaciones o Ratings',
ylabel='Películas',
save_path='img/figure_3.png')
```

.



La concentración de preferencias en Drama y Comedia permite diseñar estrategias de posicionamiento de catálogo y orientar la inversión en adquisición de contenido. Este hallazgo es coherente con la teoría de la demanda concentrada en catálogos digitales documentada por (Deldjoo, Schedl, Hidasi, Wei, & He, 2022), donde una fracción pequeña de ítems genera la mayoría de las interacciones del sistema.

Como resultados mediante KNN con similitud coseno, permite analizar la estructura de vecindad en el espacio de calificaciones, demostró ser efectivo para identificar películas con perfiles de calificación similares. En el que, el análisis de la película "Batman Forever (1995)" (ID: 153) produjo los siguientes 10 vecinos más cercanos en el espacio de calificaciones:

Figura 11

Análisis de Similitudes – Aplicación del Método de KNN

```
def find_similar_movies(movie_id, X, k, metric='cosine', show_distance=False):
neighbour_ids = []
movie_ind = movie_mapper[movie_id]
movie_vec = X[movie_ind]
k+=1
kNN = NearestNeighbors(n_neighbors=k, algorithm="brute", metric=metric)
kNN.fit(X)
movie_vec = movie_vec.reshape(1,-1)
neighbour = kNN.kneighbors(movie_vec, return_distance=show_distance)
for i in range(0,k):
```

.

```
n = neighbour.item(i)
neighbour_ids.append(movie_inv_mapper[n])
neighbour_ids.pop(0)
return neighbour_ids
movie_titles = dict(zip(movies['movieId'], movies['title']))
# print(movie_titles)
movie_id = 153
similar_ids = find_similar_movies(movie_id, X, k=10)
movie_title = movie_titles[movie_id]
print(f"Since you watched {movie_title}")
for i in similar_ids:
print(movie_titles[i])
```

```
Since you watched Batman Forever (1995)
Batman (1989)
True Lies (1994)
Ace Ventura: Pet Detective (1994)
Cliffhanger (1993)
Dances with Wolves (1990)
Die Hard: With a Vengeance (1995)
Stargate (1994)
Aladdin (1992)
Clear and Present Danger (1994)
Interview with the Vampire: The Vampire Chronicles (1994)
```

La función find_similar_movies() implementa el algoritmo K-Nearest Neighbors (KNN) utilizando similitud coseno para identificar las películas más parecidas a una película objetivo. El algoritmo KNN es un método de aprendizaje supervisado perezoso (lazy learning) que no construye un modelo explícito durante el entrenamiento, sino que computa las distancias en el momento de la consulta (Koren, Rendle, & Bell, 2021). Por lo tanto, toma como entrada el ID de una película (movie_id), la matriz de características X, el número de vecinos (k), la métrica de distancia a utilizar (metric), y un indicador para mostrar o no las distancias (show_distance). Por lo que, en primer lugar, convierte el movie_id en el índice correspondiente dentro de la matriz mediante un sistema de mapeo inverso. Posteriormente, utiliza la biblioteca scikit-learn para aplicar el algoritmo k-NN, calculando las distancias entre la película seleccionada y el resto de las películas representadas en X. Como resultado, devuelve una lista con los identificadores de las películas más cercanas, excluyendo la película original del conjunto de recomendación. En caso de activarse la opción correspondiente, también puede mostrar las distancias asociadas a cada vecino. A modo de aplicación, se ejemplifica el procedimiento utilizando la película con ID 153, imprimiendo primero el título de la película base y luego los títulos de las películas identificadas como similares.

El modelo captura coherentemente la dimensión temporal (peliculas de los 90) y de genero (acción/aventura), evidenciando que la co-ocurrencia de valoraciones en el espacio de usuarios es un indicador proxy eficiente de similitud de contenido. Según (Koren, Rendle, & Bell, 2021), este comportamiento denominado *latent factor alignment* es la razón fundamental por la que el filtrado colaborativo opera con alta precisión sin acceso a los metadatos del contenido.

Figura 12

Recomendación de Películas en Función de las Preferencias del Usuario

```
def recommend_movies_for_user(user_id, X, user_mapper, movie_mapper, movie_inv_mapper, k=10):
df1 = ratings[ratings['userId'] == user_id]
if df1.empty:
print(f"User con ID {user_id} no existe.")
return
movie_id = df1[df1['rating'] == max(df1['rating'])]['movieId'].iloc[0]
movie_titles = dict(zip(movies['movieId'], movies['title']))
similar_ids = find_similar_movies(movie_id, X, k)
movie_title = movie_titles.get(movie_id, "Película no encontrada")
if movie_title == "Película no encontrada":
print(f"Película con ID {movie_id} no encontrada.")
return
print(f"Desde que viste {movie_title}, también te puede gustar:")
for i in similar_ids:
print(movie_titles.get(i, "Película no encontrada"))

# Vamos a filtrar a la movie_id 153 de Batman Forever y los usuarios que le dieron el mayor puntaje:
query_2 = ratings[(ratings['movieId'] == 153) & (ratings['rating'] == max(ratings['rating']))]["userId"].tolist()
type(query_2)

for user_id in query_2:
print(f"\nRecomendaciones para el Usuario con ID {user_id}:\n")
recommend_movies_for_user(user_id, X, user_mapper, movie_mapper, movie_inv_mapper, k=10)
print("="*50)
```

```
Recomendaciones para el Usuario con ID 43:

Desde que viste Toy Story (1995), también te puede gustar:
Toy Story 2 (1999)
Jurassic Park (1993)
Independence Day (a.k.a. ID4) (1996)
Star Wars: Episode IV - A New Hope (1977)
Forrest Gump (1994)
Lion King, The (1994)
Star Wars: Episode VI - Return of the Jedi (1983)
Mission: Impossible (1996)
Groundhog Day (1993)
Back to the Future (1985)
==================================================

Recomendaciones para el Usuario con ID 99:

Desde que viste Apollo 13 (1995), también te puede gustar:
Jurassic Park (1993)
Fugitive, The (1993)
Forrest Gump (1994)
True Lies (1994)
Braveheart (1995)
Dances with Wolves (1990)
Shawshank Redemption, The (1994)
Batman (1989)
Aladdin (1992)
Outbreak (1995)
==================================================
```

```
==================================================

Recomendaciones para el Usuario con ID 234:

Desde que viste Toy Story (1995), también te puede gustar:
Toy Story 2 (1999)
Jurassic Park (1993)
Independence Day (a.k.a. ID4) (1996)
Star Wars: Episode IV - A New Hope (1977)
Forrest Gump (1994)
Lion King, The (1994)
Star Wars: Episode VI - Return of the Jedi (1983)
Mission: Impossible (1996)
Groundhog Day (1993)
Back to the Future (1985)
==================================================

Recomendaciones para el Usuario con ID 321:

Desde que viste Jumanji (1995), también te puede gustar:
Lion King, The (1994)
Mrs. Doubtfire (1993)
Mask, The (1994)
Jurassic Park (1993)
Home Alone (1990)
Nightmare Before Christmas, The (1993)
Aladdin (1992)
Beauty and the Beast (1991)
Ace Ventura: When Nature Calls (1995)
Santa Clause, The (1994)
==================================================

Recomendaciones para el Usuario con ID 458:

Desde que viste Get Shorty (1995), también te puede gustar:
Dave (1993)
Fugitive, The (1993)
Four Weddings and a Funeral (1994)
Sleepless in Seattle (1993)
Quiz Show (1994)
Ed Wood (1994)
In the Line of Fire (1993)
True Lies (1994)
Batman (1989)
Dances with Wolves (1990)
==================================================
```

La función recommend_movies_for_user tiene como objetivo generar recomendaciones personalizadas a partir del historial de valoraciones de un usuario específico. Para ello, recibe como parámetros los diccionarios user_mapper, movie_mapper y movie_inv_mapper, que

permiten traducir los identificadores originales de usuarios y películas a los índices correspondientes dentro de la matriz usuario-ítem *X*, además del user_id objetivo y un parámetro opcional *k* que define el número de recomendaciones a generar (pordefecto, 10). El procedimiento inicia filtrando el DataFrame ratings para recuperar las valoraciones asociadas al usuario; si no existen registros, la función notifica que el usuario no se encuentra en la base de datos y finaliza su ejecución. Posteriormente, identifica la película con mayor puntuación otorgada por dicho usuario y utiliza su identificador como referencia para invocar la función find_similar_movies, la cual aplica el algoritmo k-NN sobre la matriz de características para localizar películas con patrones similares. Finalmente, se imprime el título de la película mejor valorada y los títulos de las películas recomendadas; en caso de que alguna no se encuentre en el DataFrame de películas, se genera un mensaje de advertencia. Como ejemplo de aplicación, se ilustra el uso del método con la película de ID 153, mostrando primero el título original y luego los títulos de las películas identificadas como similares en función de sus características.

Los resultados en la **figura 12**, evidencian que el sistema genera recomendaciones coherentes con el género y la era de las películas consumidas. Usuarios con preferencia por el drama (Usuario 99 con Apollo 13) reciben recomendaciones de dramas y thrillers de los 90, mientras que usuarios con gusto por la acción (Usuario 486 con Heat) reciben propuestas del género policial. Esto es consistente con los hallazgos de (Yi, Yang, Hong, Zhiyuan Cheng, & Heldt, 2019), quienes reportan que los sistemas basados en el ítem mejor valorado del usuario capturan adecuadamente las preferencias de género.

Conclusión

En conclusión, el presente artículo implementó un sistema de recomendación de películas basado en filtrado colaborativo mediante Apache Mahout y KNN con similitud coseno, aplicado al dataset Yahoo Research Webscope, con enfoque en el marketing predictivo y la identificación de patrones de consumo en entornos digitales. Tomando como referencia metodológica el trabajo de (Wu, Garg, & Bhandary, 2018). Para lo cual, se realizó la integración del algoritmo KNN con similitud coseno sobre matrices dispersas CSR, el análisis estadístico profundo de patrones de consumo con orientación a marketing, y la validación multi-perfil que demuestra la adaptabilidad del sistema a distintos segmentos de consumidores. Por lo tanto, los resultados evidencian que el filtrado colaborativo basado en items identifica con precisión patrones latentes de consumo y genera recomendaciones adaptadas al perfil individual de cada

usuario. La concentración de demanda en los géneros Drama y Comedia, la variabilidad inter-perfil de las recomendaciones y la coherencia de los ítems sugeridos validan la capacidad del sistema como herramienta de marketing one-to-one escalable.

Referencias Bibliográficas

Aggarwal, C. (2016). *Sistemas de recomendación.* Springer. doi:https://doi.org/10.1007/978-3-319-29659-3

Agrawal, S. K. (2021). *www.analyticsvidhya.com*. Obtenido de Recommendation System - Understanding: www.analyticsvidhya.com/blog/

Apache Mahout. (2 de Abril de 2018). *https://mahout.apache.org/*. Obtenido de https://mahout.apache.org/

Deldjoo, Y., Schedl, M., Hidasi, B., Wei, Y., & He, X. (2022). Sistemas de recomendación multimedia: Algoritmos y desafíos. En F. Ricci, L. Rokach, & B. Shapira, *Manual de sistemas de recomendación* (págs. 1015-1055). Nueva York: Springer. doi:https://doi.org/10.1007/978-1-0716-2197-4_26

Fraihat, S., Shambour, Q., Al-Betar, M. A., & Naser Makhadmeh, S. (8 de Diciembre de 2024). Variational Autoencoders-Based Algorithm for Multi-Criteria Recommendation Systems. *Algorithms, 17*(561). doi:https://doi.org/10.3390/a17120561

Jerez G, J. C. (28 de Junio de 2023). *https://medium.com*. Obtenido de Los Sistemas de Recomendación y la Ciencia de Datos: https://medium.com/@jcjerez_77135/los-sistemas-de-recomendaci%C3%B3n-y-la-ciencia-de-datos-1b2fa965f47b

Koren, Y., Rendle, S., & Bell, R. (2021). *Advances in Collaborative Filtering.* Springer. doi:https://doi.org/10.1007/978-1-0716-2197-4_3

Ramlatchan, A., Yang, M., Liu, Q., & Li, M. (Diciembre de 2018). A Survey of Matrix Completion Methods for Recommendation Systems. *BIG DATA MINING AND ANALYTICS, I*(4), 308-323. doi:10.26599/BDMA.2018.9020008

Sgardelis, K., Margaris, D., Spiliotopoulos, D., & Vassilakis, C. (Junio de 2025). An evaluation review of user similarity metrics in sparse collaborative filtering datasets. *International Journal of Data Science and Analytics*. doi:https://doi.org/10.1007/s41060-025-00846-4

Sharma, R. S., Shaikh, A. A., & Li, E. (6 de Mayo de 2021). Designing Recommendation or Suggestion Systems:looking to the future. *Electronic Markets, 31*, 243–252. doi:https://doi.org/10.1007/s12525-021-00478-z

Wu, C.-S. M., Garg, D., & Bhandary, U. (2018). Sistema de recomendación de películas mediante filtrado colaborativo. *9.ª Conferencia Internacional IEEE sobre Ingeniería de Software y Ciencias de los Servicios (ICSESS)*, 11-15. Pekín, China. doi:doi: 10.1109/ICSESS.2018.8663822

Yahooresearch. (24 de Febrero de 2014). *Webscope*. Obtenido de https://yahooresearch.tumblr.com: https://yahooresearch.tumblr.com/post/77697901734/welcome-to-webscope

Yi, X., Yang, J., Hong, L., Zhiyuan Cheng, D., & Heldt, L. (2019). *Modelado neuronal con corrección del sesgo de muestreo para recomendaciones de elementos de corpus grandes.* Proceedings of the 13th ACM Conference on Recommender Systems (RecSys). doi:https://doi.org/10.1145/3298689.3346996

Yoo, H., Kang, S., & Tong, H. (2025). Sistemas de recomendación continua. *CIKM '25: Actas de la 34ª Conferencia Internacional de la ACM sobre Gestión de la Información y el Conocimiento* (págs. 6857 - 6860). Association for Computing Machinery. doi:https://doi.org/10.1145/3746252.3761452

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Sistema de recomendación basado en aprendizaje profundo: una encuesta y nuevas perspectivas. *ACM Computing Surveys, 52*(5), 1-38. doi:https://doi.org/10.1145/3285029

Conflicto de intereses:

Los autores declaran que no existe conflicto de interés posible.

Financiamiento:

No existió asistencia financiera de partes externas al presente artículo.

Agradecimiento:

Universidad Técnica Estatal de Quevedo – Facultad de Ciencias Empresariales

Nota:

El artículo no es producto de una publicación anterior.