UNIVERSITAT DE BARCELONA

Trabajo de fin de grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

DESARROLLO DE UN SISTEMA DE RECOMENDACIÓN PARA UNA EMPRESA DE SERVICIOS ONLINE

Jia Hao Huang Xia

Director: Marc Soler Bages
Realizado a: Departamento de Matemáticas e Informática

Barcelona, 13 de juny de 2023

Abstract

A recommender is a system designed to provide personalized recommendations to users based on their preference, interest and history. A recommendation system analyzes these datas from the users profiles, data from the items available and the historical interaction to generate recommendations that are likely to be of interest to the users.

This project's purpose is the development of such a recommender for an online services company. The recommendation system should be able to provide each user's recommendations based on the history of other users similar to the user.

Resumen

Un recomendador es un sistema diseñado para proporcionar recomendaciones personalizadas a los usuarios basadas en sus preferencias, intereses e historial. Un sistema de recomendación analiza datos como perfiles de usuarios, información de los elementos disponibles y la interacción histórica para generar recomendaciones que sean de interés para los usuarios.

El propósito de este proyecto es desarrollar un recomendador para una empresa de servicios online. El sistema de recomendación deberá ser capaz de proporcionar recomendaciones a cada usuario basadas en el historial de otros usuarios similares a ellos.

Agradecimientos

En primer lugar, quiero dar las gracias a mi tutor, Marc Soler Bages, por su valiosa ayuda, apoyo y orientación durante el desarrollo de este proyecto.

En segundo lugar, le doy las gracias a Blai Ras, por su amabilidad y por la ayuda que me ha proporcionado durante este tiempo.

Y finalmente, también le quisiera dar las gracias a la empresa, cuyo nombre mantendré anónimo debido a motivos de privacidad, por darme la oportunidad de realizar este trabajo y por proporcionar los datos que se han usado para crear el sistema de recomendación.

Índice

1. Introducción

Antes de discutir sobre los sistemas de recomendación, hay que hablar sobre la **información** [1].

La información siempre ha sido una parte esencial de la naturaleza. Desde tiempos ancestrales, hemos ido guardando y pasando información a las futuras generaciones. Y no fue hasta hace cinco milenios, que se pudo convertir en poder, con la invención de la escritura, que se convirtió en el pilar del mundo moderno. Esta habilidad fue la que hizo posible preservar las ideas por muchísimo tiempo.

Fue en el siglo XX que Claude Shannon comprendió la esencia de la información. Ella se dio cuenta de que el contenido de la información de un mensaje depende únicamente de cuánto difiere de la media. No importa tanto el contenido o la longitud del mensaje, sino si es inesperado o no.

También descubrió que la información se puede convertir en números, o más precisamente, en sistemas binarios. Un bit (que representa un 0 o 1) puede representar sí o no, existente o no. Pero su poder radica en que con una cadena de estos bits podemos codificar cualquier cosa. Mediante la digitalización, la humanidad fue capaz de lograr que la información no solo fuera duradera, sino que también se pudiera almacenar en cantidades virtualmente infinitas y distribuirlas de manera extremadamente rápida.

A medida que las computadoras fueron ganando popularidad para el uso civil, las empresas de desarrollo y los investigadores se centraron en satisfacer las expectativas de sus usuarios cada vez con más eficiencia. Cada vez reduciendo más las molestias que podría tener un usuario al usar su producto. De esta manera iban surgiendo cada vez más soluciones que les ayudaban a comprender las necesidades de las personas y personalizar los servicios proporcionados por las computadoras.

Los sistemas de recomendación se basaron en investigaciones en ciencias cognitivas y recuperación de información, y su primera manifestación fue el sistema de comunicación Usenet creado por la Universidad de Duke a mediados de la década de 1970. En Usenet, los usuarios podían compartir contenido textual entre sí, clasificado en grupos y subgrupos para facilitar la búsqueda, pero no se construyó directamente sobre o se orientó a las preferencias de los usuarios.

La primera solución conocida de este tipo fue el bibliotecario de computadoras Grundy, que entrevistaba a los usuarios sobre sus preferencias y luego les recomendaba libros en función

de esa información. Según los datos recopilados, el sistema asignaba al usuario a un grupo estereotipado utilizando un método bastante primitivo, recomendando los mismos libros a todas las personas en el mismo grupo.

Aunque pueda parecer anticuado ahora mismo, para ese entonces esto era algo revolucionario para los servicios automatizados, ya que era personalizado. Y es importante destacar que, incluso en la actualidad, no todos los sitios webs de comercios electrónicos han alcanzado este hito.

Sin embargo, la solución de Grundy recibió críticas por parte del mundo científico. Según estudios de Nisbett y Wilson, las personas son muy débiles en el estudio y la descripción de sus propios procesos cognitivos. Estos a menudo resaltan sus atributos que les hacen destacar del resto del grupo, lo que dificulta los esfuerzos a estereotipar. O también puede pasar que quieran crear una imagen diferente de ellos mismos.

Y Heli Vaino, gerente de un centro comercial, equipó su centro comercial con equipos de wifi para rastrear a los visitantes dentro y en las inmediaciones del edificio, con el objetivo de recopilar información de los visitantes a través de sus acciones, en vez de lo que hablen.

Entonces surgieron los dos enfoques principales que muchos conocemos, el filtrado colaborativo y el filtrado basado en contenido. Para ver la definición de los dos enfoques, véase los capítulos 2.2.3 y 2.2.4.

En la actualidad, la industria del entretenimiento está en constante crecimiento y evolución. Y una de las formas más populares de entretenimiento es el cine, el cual cuenta con una cantidad abundante de películas disponibles para el público. Sin embargo, con esta gran cantidad de posibles opciones, se puede hacer que la elección de una película sea una tarea difícil para muchos espectadores.

Es por eso que los sistemas de recomendación de películas han ganado popularidad en los últimos años. Estos sistemas ayudan a los consumidores a encontrar películas que puedan ser de su agrado, basándose en múltiples datos, como podría ser el historial de visualización o las calificaciones que ha dado a otras películas.

"*We live in an attention economy, where there's an overwhelming number of things, and recommender systems help us make decisions*" Nitya Mandyam, Senior Curriculum Developer en Codecademy.

1.1 Objetivo

El trabajo que quiero realizar tiene como objetivo el desarrollo de un sistema de recomendación de contenido multimedia para una empresa, y estará principalmente centrado en recomendar películas.

El principal objetivo es diseñar e implementar un sistema de recomendación de películas, y se utilizará el modelo de filtrado colaborativo basado en la similitud de usuarios, ya que, con los datos disponibles, este modelo es el que le puede sacar más provecho.

Este modelo de recomendación de películas tiene como objetivo proporcionar recomendaciones de películas personalizadas a los usuarios, basadas en las preferencias de otros usuarios similares.

Entre los objetivos que se ha propuesto, incluyen:

1. Recopilar y buscar información sobre trabajos anteriores que traten sobre un tema similar al nuestro: Estudiar bien estos trabajos y formular una idea de qué manera se querrá hacer el recomendador.
2. Recopilar y procesar datos de películas y de usuarios: Se debe recopilar información sobre las películas disponibles y sobre los usuarios que utilizarán el sistema de recomendación. Es importante procesar y analizar esta información para prepararla para el uso del modelo de filtrado colaborativo.
3. Diseñar e implementar el modelo de filtrado colaborativo: Se debe diseñar y programar el modelo que utilizará el sistema de recomendación. Esto implica la implementación de algoritmos y técnicas de análisis de datos para encontrar similitudes entre los usuarios y proporcionar recomendaciones basadas en estas similitudes.
4. Validar y optimizar el modelo de recomendación: Es importante evaluar la precisión y efectividad del modelo de recomendación y optimizarlo para mejorar su rendimiento. Esto significa que hay que realizar pruebas y validaciones.
5. Finalmente, intentar integrar el sistema de recomendación en la plataforma de la empresa.

En general, el objetivo de este trabajo es proporcionar una solución efectiva y personalizada para recomendar películas a los usuarios de la empresa, lo que mejorará la experiencia del usuario y aumentará la satisfacción del cliente.

1.2 Motivación

Una de las motivaciones principales para realizar este trabajo, como ya he mencionado anteriormente, es la falta de un sistema de recomendación en la empresa.

Esta empresa proporciona servicios online de entretenimiento a sus clientes, y su prioridad principal es la satisfacción de los usuarios de estos servicios.

Y un miembro del equipo de analistas de datos pensó que una de las cosas que se podría hacer para aumentar la satisfacción de los usuarios es la implementación de un sistema de recomendación de películas dentro de las plataformas, cuyo propósito sería recomendar posibles películas que podría llegar a interesarle al usuario y, de esta manera, prolongar el tiempo de conexión del usuario.

2. Estado del arte

2.1 Sistemas de recomendación

Un sistema de recomendación es una herramienta que se usa para sugerir productos o servicios al usuario. Su propósito principal es, a través de una serie de valoraciones y criterios sobre los datos del usuario o del ítem, predecir qué productos o servicios podrían ser del agrado del usuario, con el fin de mejorar su experiencia.

Los sistemas de recomendación se usan en una variedad de industrias, como por ejemplo la publicidad, la música, los libros, los juegos, las series y el cine. En el caso del cine, un sistema de recomendación de películas puede sugerir películas que se ajusten a los intereses y preferencias de los usuarios.

Existen diferentes tipos de técnicas que se pueden aplicar a un sistema de recomendación, como podemos observar en la Figura 1. Y dependiendo de los datos y herramientas que tengamos a nuestra disposición, usaremos una u otra.

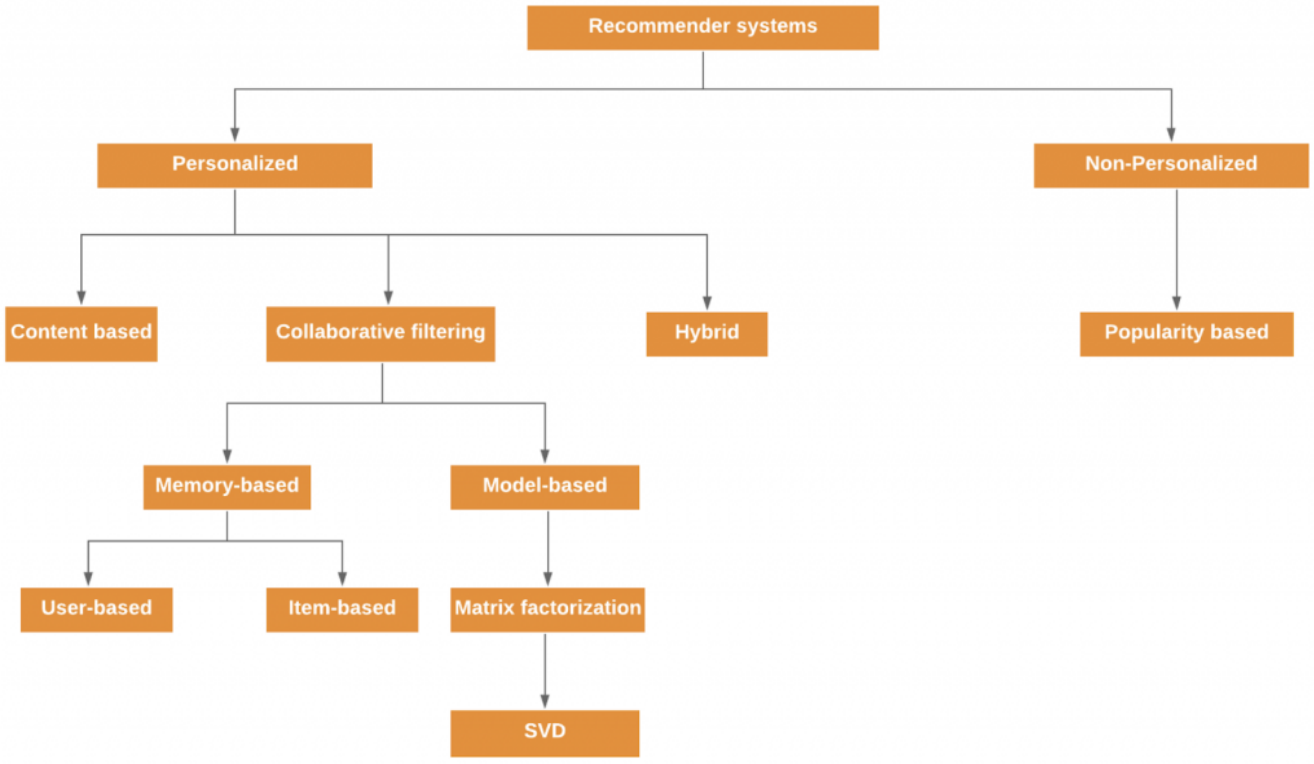

Figura 1: Clasificación de recomendadores

2.2 Modelos de sistemas de recomendación

Primero de todo hay que diferenciar los tipos de sistemas de recomendación que hay.

2.2.1 Basados en Popularidad

El modelo basado en la popularidad se trata del sistema más común [3]. El modelo provee recomendaciones basadas en el número de visitas, likes, valoraciones o compras. Las recomendaciones no están personalizadas para los usuarios, es decir, que los productos o servicios más populares son recomendados a todos los usuarios de la plataforma.

Ejemplos de plataformas que tienen este tipo de recomendador son YouTube, Netflix, Amazon o Spotify.

2.2.2 Basados en conocimiento

El modelo realiza recomendaciones basadas en información que conocen a priori o a través de la interacción con el usuario. Está basado en satisfacer las necesidades del usuario en vez de sus intereses. [3]

2.2.3 Filtrado colaborativo

El filtrado colaborativo también se trata de uno de los métodos más comunes en los recomendadores. Dentro de este modelo podemos encontrar las técnicas llamadas Memory-based, que utilizan toda la matriz de datos con sus calificaciones para generar una predicción. Tiene dos enfoques, el *filtrado colaborativo basado en usuarios* y el *filtrado colaborativo basado en elementos*.

Basado en usuarios

El filtrado colaborativo basado en usuarios tiene como objetivo principal predecir los intereses de un usuario mediante la información que se le ha proporcionado sobre el historial, preferencias e información de muchos usuarios. Básicamente, el modelo buscará usuarios con gustos similares al del usuario objetivo (Véase la Figura 2), y recomendará productos que les hayan gustado a estos usuarios, ya que, si dos usuarios tienen gustos parecidos, seguramente les gusten los mismos productos. Para encontrar un conjunto de usuarios parecidos, se usan técnicas como el Nearest Neighbor Search[1].

La ventaja de este sistema es que es de fácil implementación y brindan un alto nivel de cobertura. Y además es capaz de capturar características sutiles y no requiere tener una compresión del contenido del ítem.

La desventaja es que tendrá dificultades en recomendar productos nuevos, ya que, al ser nuevo el ítem, este tendrá una falta de interacción con los usuarios. Esto también se aplica

para los usuarios nuevos, porque, debido a la falta de un historial, no se podrá sugerir recomendaciones personalizadas.

Basado en elementos

Por otro lado, el filtrado colaborativo basado en elementos recomienda ítems basados en los ratings de los ítems realizados por otros usuarios en el sistema. La diferencia entre este filtrado y el anterior, es que no se realiza la búsqueda de vecindad de usuarios, sino que calcula directamente la similitud entre elementos.

Figura 2: Matriz de similitud basado en usuarios

2.2.4 Basados en contenido

Este tipo de modelo realiza recomendaciones basadas en la similitud de los ítems de un producto o servicio que le hayan interesado al usuario [4] y el perfil del usuario. La lógica detrás de este modelo es muy simple, si a un usuario le ha gustado la película Iron Man, entonces el sistema le recomendará películas del género superhéroe.

Según [5], a lo largo del tiempo, diferentes algoritmos han ido desarrollándose dentro de este ámbito. Donde la mayoría de los sistemas utilizan una técnica conocida como “singleShot”, la cual produce una única lista de recomendaciones para el usuario. Esta estrategia resultó muy exitosa durante muchos años, cuando la complejidad de los productos que se recomendaban en ese entonces eran muy simples, como películas, libros, música, etc. Sin embargo, se ha demostrado que no es tan adecuada para productos más complejos. Y ante la necesidad de un enfoque más sofisticado, surgió una nueva estrategia llamada Recomendadores Conversacionales.

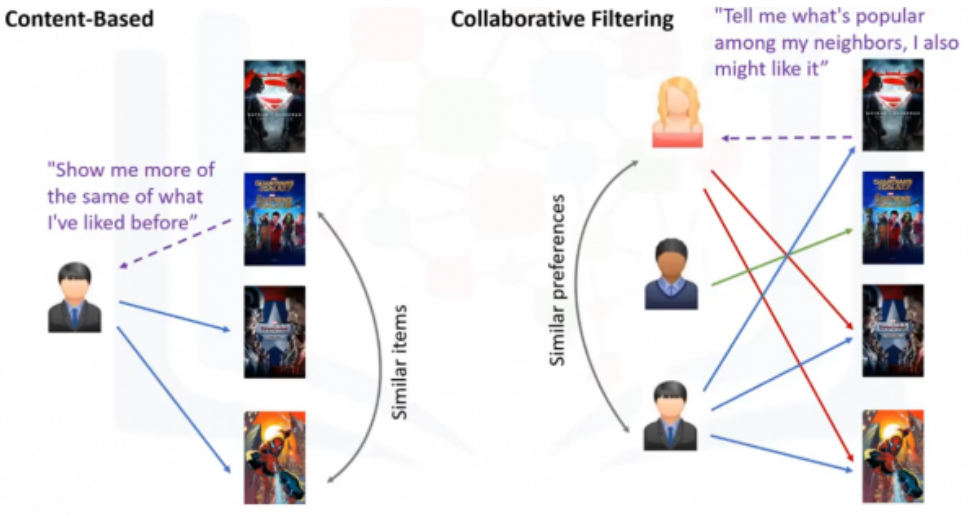

Figura 3: Filtrado colaborativo y Basado en contenido

2.2.5 Recomendadores Conversacionales

En este nuevo enfoque, según [5], los usuarios participan en un diálogo de recomendación, donde reciben recomendaciones y devuelven retroalimentación en forma de críticas sobre esas recomendaciones. De esta manera, permite al sistema refinar la búsqueda y ofrecer un conjunto de productos más adecuados a las preferencias del usuario.

En resumen, los recomendadores conversacionales son sistemas que guían al usuario a través del espacio de productos, ofreciendo sugerencias y solicitando feedback.

Algunos ejemplos destacados de recomendadores conversacionales son:

- **Value Elicitation:** Esta metodología se basa en un valor específico introducido por el usuario, lo que permite una búsqueda efectiva en función de una característica, como por ejemplo, un género = acción. La desventaja es que el usuario debe tener un amplio conocimiento sobre las características del producto o ítem a buscar, de lo contrario, este sistema resulta ineficaz.

- **Ratings-based feedback:** A diferencia del enfoque anterior, esta metodología no requiere un gran conocimiento por parte del usuario sobre las características del producto que busca. Los usuarios asignan una calificación sencilla, como por ejemplo, 4 estrellas de 5, para indicar su satisfacción con la recomendación. En este enfoque, el usuario no devuelve una retroalimentación detallada sobre las características del producto.

- **Preference-based feedback:** Esta metodología el usuario indica su recomendación preferida, en vez de clasificar un determinado conjunto de recomendaciones. Se trata de un enfoque de muy bajo coste de retroalimentación por parte del usuario, y además

requiere de un mínimo conocimiento de dominio, solamente la capacidad de distinguir si una recomendación es mala o buena.

- **Critiquing**: Los usuarios dan retroalimentaciones específicas sobre las recomendaciones. Donde expresan sus preferencias, en lugar de simplemente aceptar o rechazar. Por ejemplo, retroalimentaciones de este estilo "Dame más productos de este estilo, pero ...".

En la Figura 4 se puede observar la estructura de un recomendador conversacional. Y en la Figura 5 podemos ver los diferentes enfoques del recomendador.

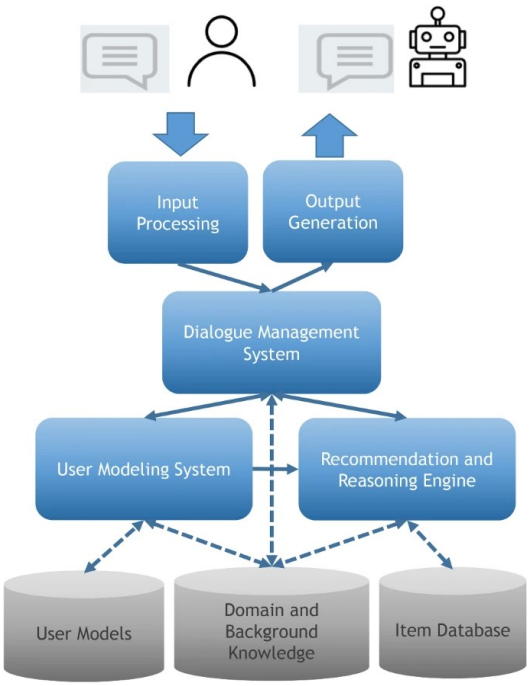

Figura 4: Estructura de un recomendador conversacional

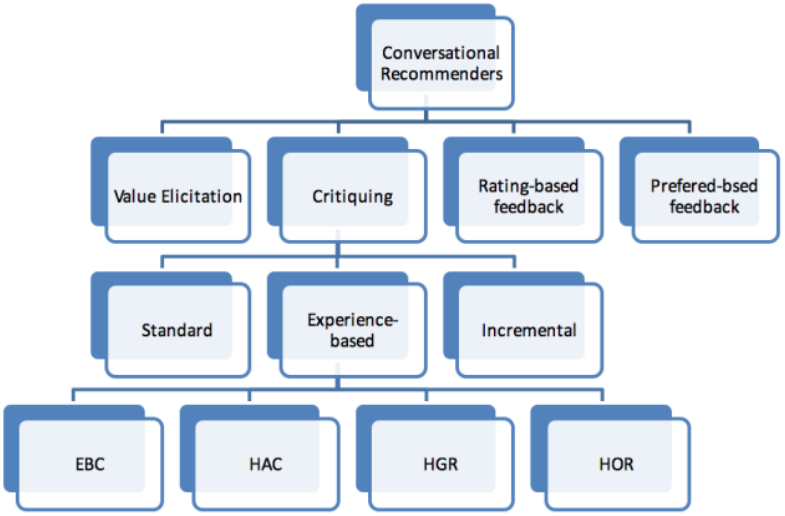

Figura 5: Clasificación de los Recomendadores Conversacionales

2.2.5 Modelos Híbridos

Se trata de un modelo que combina diferentes enfoques con el objetivo de juntar sus mejores características y mejorar su rendimiento, de esta manera poder generar mejores recomendaciones [6].

Los sistemas de recomendación híbrida pueden dividirse en dos grupos:

- **De combinación lineal:** Son aquellos que crean una lista de recomendaciones sin combinarlas para crear una predicción combinada.
- **De combinación secuencial:** Donde la salida de una técnica de recomendación es la entrada a otra técnica.

2.3 Base de Datos

Para poder llegar a recomendar algún producto o servicio a un usuario, los sistemas de recomendación necesitan acceso a grandes cantidades de datos sobre los usuarios y los elementos que se están recomendando. Aquí es donde entran las bases de datos.

Una base de datos es un conjunto organizado de datos que se almacenan y se acceden de manera eficiente. Para un sistema de recomendación, las bases de datos son el almacén donde se guardan todo tipo de información relacionado con el usuario, ítem y las interacciones entre ellos.

Por ejemplo, si un sistema de recomendación de películas quiere proporcionar recomendaciones a un usuario, necesitará información sobre el usuario, como por ejemplo su historial, sus gustos, qué películas le gustó, etc. Además, también necesitará información sobre las películas disponibles en su base de datos, como los títulos, géneros, actores, directores y más.

Una vez que el sistema de recomendación tenga acceso a una base de datos, podrá utilizar técnicas de aprendizaje para analizar los patrones en los datos y hacer recomendaciones precisas al usuario.

En mi caso, para este trabajo usaré la base de datos que usa la empresa, donde están almacenados todos los datos de la plataforma, tanto del usuario como de los vuelos, entre otras cosas. Se me ha otorgado derecho a descargar cualquier tipo de datos que crea necesario para el desarrollo del sistema de recomendación.

Y, viendo la información que se puede recopilar de la base de datos, he llegado a la conclusión de que el modelo que más puede aprovechar estos datos es Filtrado colaborativo basado en la similitud de los usuarios.

2.4 Datos

Como ya hemos mencionado antes, los datos que usaremos para entrenar el modelo se cogerá de la base de datos de la empresa, a través de Kibana, que usa Elasticsearch.

2.4.1 Elasticsearch

ElasticSearch es un motor de búsqueda y análisis de datos distribuidos que está basado en Apache Lucene[2]. Fue diseñado para el almacenamiento, búsqueda y análisis de grandes volúmenes de datos en tiempo real.

2.4.2 Kibana

Kibana es una plataforma de visualización y análisis de datos que se usa junto al ElasticSearch. Esta herramienta nos ofrece una serie de ventajas y funcionalidades:

- **Visualización**: Esta es la funcionalidad principal de Kibana y permite crear visualizaciones interactivas y dinámicas a partir de los datos almacenados en Elasticsearch. Es posible crear gráficos, diagramas, mapas, tablas y otros tipos de visualizaciones.
- **Exploración**: Kibana facilita la exploración de datos en tiempo real, permite realizar búsquedas y filtrar datos de forma interactiva para descubrir patrones, tendencias y anomalías en los datos.
- **Personalizable**: Es personalizable y flexible, ya que permite crear visualizaciones personalizadas a la necesidad del usuario.

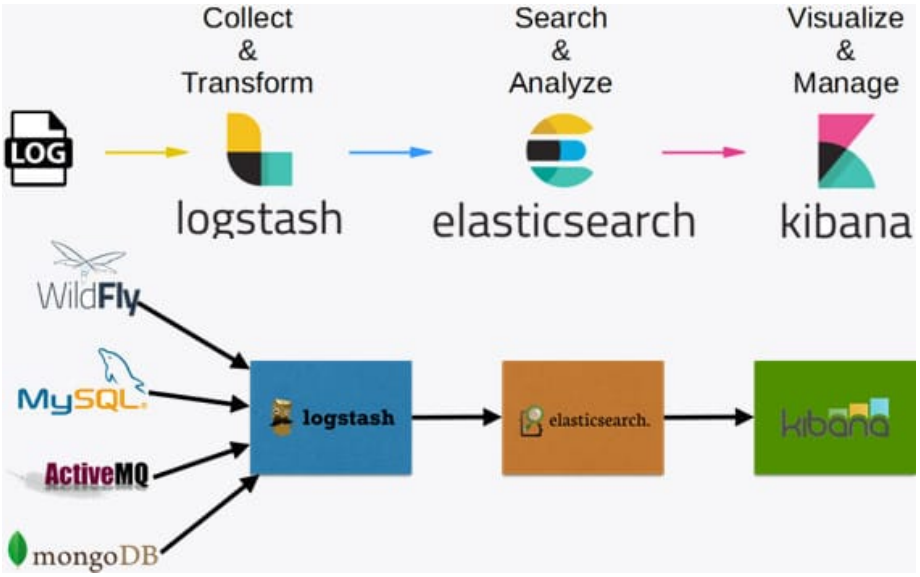

Figura 6: Arquitectura del Elasticsearch y Kibana

2.5 Datos disponibles

De la información del usuario, hemos pensado que puede haber múltiples factores que podrían influenciar el gusto de una persona. Y es por eso que tenemos estas columnas del usuario:

- ID del usuario: El identificador del usuario.
- Género
- Edad
- Nacionalidad

Y de la película tenemos:

- Nombre de la película
- Duración de la película
- Progreso de la película por usuario
- Likes
- Número de reproducciones

Y del vuelo:

- Rutas del vuelo (Origen y destino)

Como podréis ver, no tenemos una columna para la puntuación del usuario, cosa que no suele ser común con plataformas que proveen entretenimiento tipo películas. Como consecuencia, hemos tenido que buscar otra alternativa para evaluar si al usuario le ha gustado la película o no.

Esta se ha calculado usando las columnas de "Duración de la película" ("Duration") y "Progreso de la película por usuario" ("Progress"). Si el usuario ha visto un cierto porcentaje de la película, tomamos como que le ha gustado. Este valor puede variar dependiendo de la aerolínea y del porcentaje visto por la mayoría de los usuarios.

	Name	Day	User ID	Gender	Birthyear	Country	Origin ICAO	Destination ICAO	Progress	Duration
0	The Witches	2023-04-01	2336	Female	1-Jan-1979 12:00 AM	Mexico	MMQT	MMUN	3,276.40	6,000.00
1	The Witches	2023-04-01	3958	Female	1-Jan-1979 12:00 AM	Mexico	MMMY	MMQT	364.878	6,000.00
2	The Witches	2023-04-01	1114	Female	1-Jan-1982 12:00 AM	Mexico	MMUN	SKBO	2,779.64	6,000.00
3	The Witches	2023-04-01	4218	Female	1-Jan-1987 12:00 AM	Mexico	MMMY	MMQT	159.852	6,000.00
4	The Witches	2023-04-01	446	Female	1-Jan-1979 12:00 AM	Mexico	MMQT	MMUN	3,378.05	6,000.00

Figura 8: Ejemplo del dataset

2.5.1 Datos sensibles

Cabe mencionar que, debido a que esto se trata de un trabajo que se guardará en la web de la facultad de informática, se han modificado los datos por motivos de privacidad. En otras palabras, hay datos del dataset[3] que han sido modificadas o rehechas para que no se muestre nada que pueda llegar a ser sensible o comprometedor. Los datos que se han usado son lo suficientemente genéricos como para no causar ningún problema a la privacidad de los usuarios.

2.6 Análisis de datos

Una vez que se ha completado la descarga de los datos, es necesario llevar a cabo un proceso de selección para determinar qué variables o características tienen más relevancia. Es decir, es necesario identificar qué datos son más importantes para construir la matriz de similitud.

Para llevar a cabo esta tarea, podemos hacer un estudio y analizar los datos que tenemos disponibles. Esto implica aplicar técnicas de análisis de datos para evaluar y ponderar la importancia de cada variable. Existen diversas aproximaciones para realizar esta tarea, pero para mi caso he mirado métodos de análisis de correlación.

- **Análisis de correlación:** Este tipo de análisis es simplemente evaluar la relación entre las variables y, en nuestro caso, la preferencia del usuario. Esto dará una idea de cuánto cada dato está relacionado con los gustos de los usuarios. Los datos con una correlación más alta se considerarán como más importantes. Existen múltiples tipos de análisis de correlación, como por ejemplo Pearson o Spearman [7].

2.6.1 Correlación de Pearson

Uno de los métodos más comunes es el coeficiente de correlación de Pearson, que calcula la relación lineal entre las variables. La correlación de Pearson se calcula dividiendo la covarianza de las variables entre el producto de sus desviaciones estándar. Su valor puede variar entre -1 y 1 como podemos observar en la Figura 9, donde 1 indica una correlación positiva, -1 una correlación negativa y 0 indica que no hay correlación [8].

Con este método se podría evaluar la relación entre las variables cuantitativas[4], como la edad del usuario o el día de visualización.

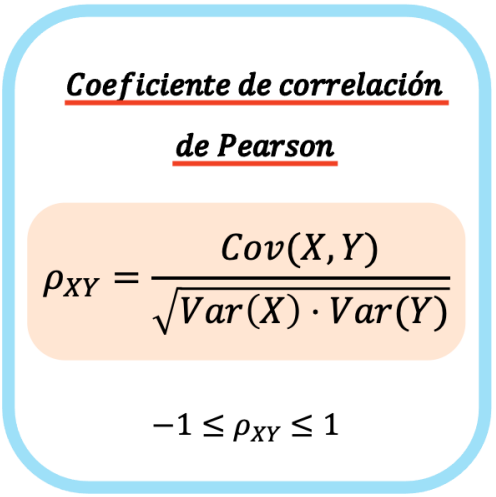

Figura 9: Correlación de Pearson

2.6.2 Correlación de Spearman

La correlación de Spearman se utiliza para medir la correlación entre dos variables ordinales o variables no paramétricas. A diferencia del coeficiente de correlación de Pearson, este no asume una relación lineal entre las variables, sino una relación de clasificación. En lugar de utilizar los valores numéricos de las variables, utiliza los rangos de las observaciones. Se calcula de manera similar al coeficiente de Pearson, pero se basa en los rangos de las observaciones en lugar de los valores brutos. Este coeficiente también puede variar entre -1 y 1, donde 1 indica una correlación positiva perfecta, -1 una correlación negativa perfecta y 0 indica que no hay correlación. [9]

$$r_s = 1 - \frac{6\sum D^2}{n\left(n^2 - 1\right)}$$

Figura 10: Correlación de Spearman

2.6.3 Similitud del coseno

El coeficiente de similitud del coseno es una métrica utilizada muy comúnmente en la matriz de similitud. Es ampliamente utilizada en la recuperación de información y estudios relacionados. Este coeficiente modela un documento de texto como un vector de términos. La similitud entre dos documentos se puede calcular mediante el valor del coseno entre los vectores de términos de los dos documentos. En el caso de un motor de búsqueda, los valores de similitud entre la consulta del usuario y los documentos se ordenan de mayor a menor. Un puntaje más alto significa una mayor relevancia entre el documento y la consulta [10].

$$\text{soft\_cosine}_1(a, b) = \frac{\sum_{i,j}^{N} s_{ij} a_i b_j}{\sqrt{\sum_{i,j}^{N} s_{ij} a_i a_j} \sqrt{\sum_{i,j}^{N} s_{ij} b_i b_j}},$$

Figura 11: Similitud del coseno

3. Herramientas utilizadas

En este capítulo hablaremos sobre las herramientas que hemos usado para la realización de este proyecto, como librerías, lenguaje, aplicación, etc. Y también se hablará sobre cómo se ha realizado el sistema de recomendación.

Como hemos mencionado al principio, el proyecto consiste en el desarrollo de un sistema de recomendación de películas para una empresa de servicios online. Nos centraremos solamente en la parte del backend y datos. El recomendador debe de ser capaz de dar sugerencias de películas para los usuarios, que son los pasajeros.

A lo largo del capítulo, explicaremos detalladamente sobre cada herramienta que hemos ido usando a lo largo del desarrollo de este sistema de recomendación.

3.1 Python

Para el desarrollo del proyecto, he decidido usar el lenguaje de programación Python.

Según [11], los lenguajes de programación son la herramienta básica de construcción de programas. Python ha ido ganando popularidad en comunidades como la de software libre, la ciencia y la educación, gracias a su simplicidad y capacidad de enfocar en los problemas actuales de manera eficaz.

Python cuenta con una multitud de funcionalidades para la programación orientada a objetos, imperativa y funcional, por lo que se le considera como un lenguaje multiparadigma[5]. Está basado en el lenguaje ABC y otros lenguajes también tuvieron influencias, como C, según su propio autor [12].

Se le considera un lenguaje de alto nivel debido a que incluye estructuras de datos integradas, como listas, diccionarios, conjuntos y tuplas. Estas estructuras permiten realizar tareas complejas en pocas líneas de código y de manera legible.

La razón por la que he escogido Python para este trabajo, es porque este cuenta con una amplia selección de librerías y herramientas diseñadas para sistemas de recomendación. Como por ejemplo las librerías Pandas, NumPy y scikit-learn.

Por no decir de la gran comunidad que se ha estado formando estos años, con una amplia cantidad de desarrolladores, foros y documentación disponibles en línea. Por lo tanto, no habrá problemas en encontrar soluciones para los problemas que puedan ir surgiendo a lo

largo del progreso, ya que, no sería raro que otro usuario haya tenido el mismo problema y haya recibido una solución.

Otra razón es que Python integra muy bien otras tecnologías utilizadas para el desarrollo de sistemas de recomendación, como por ejemplo las bases de datos, frameworks y servicios en la nube.

Y finalmente, Python es el lenguaje de programación que más he utilizado a lo largo de mi carrera, tanto en la universidad como en el trabajo. Es el lenguaje con el que estoy más familiarizado. Por no decir que de entre todos, es el que me parece más simple y flexible. De hecho, se le conoce por su simple sintaxis y su flexibilidad.

3.2 Jupyter Notebook

Jupyter Notebook es una plataforma de código abierto que se usa para crear y compartir documentos que ofrecen la posibilidad de ejecutar e integrar código en vivo, visualizar datos, ecuaciones y explicaciones en forma de texto. Jupyter Notebook es un proyecto derivado del Proyecto IPython, que solía tener su propio proyecto llamado IPython Notebook. El nombre Jupyter proviene de los lenguajes de programación principales que admite la plataforma, que son Julia, Python y R [13]. Véase la Figura 12 para ver su arquitectura.

He usado Jupyter Notebook porque es la plataforma con la que estoy más familiarizado. Además, permite combinar código, visualizaciones y explicaciones en un mismo documento, cosa que con otras plataformas no serían posibles. A la hora del desarrollo, Jupyter me permite ver los resultados que vaya generando gracias a las diferentes herramientas que tiene integrada. Por no decir de la gran comunidad que tiene.

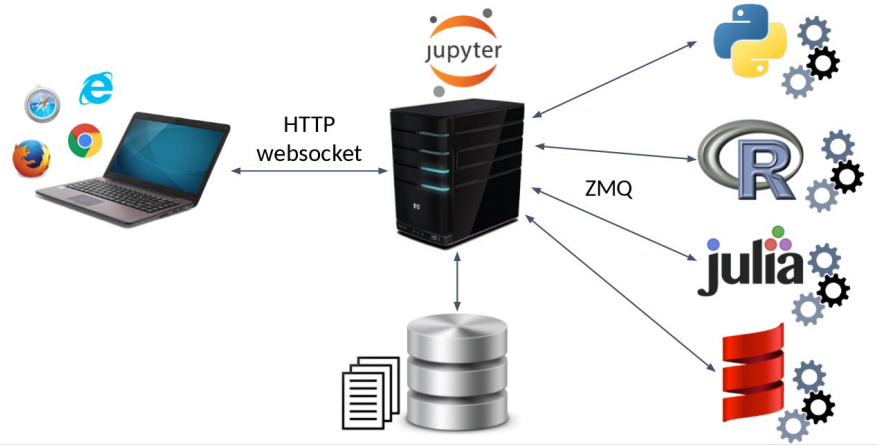

Figura 12: Arquitectura del Jupyter Notebook

3.3 Librerías

En esta sección se van a introducir las librerías principales que se han usado para el desarrollo del recomendador.

3.3.1 Pandas

Pandas es una biblioteca de software escrita en el lenguaje de programación Python, diseñada para la manipulación y análisis de datos [14]. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series de tiempo.

Su uso principal reside en el análisis de datos y la manipulación asociada de datos tabulares en DataFrames[9]. Permite importar datos desde diversos formatos de archivo como valores separados por comas (los archivos .csv), JSON, tablas o consultas de bases de datos SQL y Microsoft Excel. Pandas también permite realizar diversas operaciones de manipulación de datos, como fusionar, remodelar, seleccionar, así como funciones de limpieza y transformación de datos.

La evolución de Pandas en Python incorporó numerosas funcionalidades similares al manejo de DataFrames que ya existían en el lenguaje de programación R. La biblioteca de Pandas se basa en NumPy, otra biblioteca diseñada para el manejo eficiente de arreglos en lugar de incluir características específicas para la manipulación de DataFrames.

3.3.2 Numpy

NumPy es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel [15].

Según [16], NumPy se basa en el exitoso objeto de matriz numérica llamado Numeric y es su sucesor. Su objetivo es crear el fundamento de un entorno útil para la computación científica.

NumPy proporciona dos objetos fundamentales: un objeto de matriz N-dimensional (*ndarray*[6]) y un objeto de función universal (*ufunc*[7]). Una matriz N-dimensional es una colección homogénea de "elementos" indexados usando N enteros. Hay dos piezas esenciales de información que definen una matriz N-dimensional, su forma y el tipo de elemento del que está compuesta.

La forma de la matriz es una tupla de N enteros (uno para cada dimensión) que proporciona información sobre cómo puede variar el índice a lo largo de esa dimensión. La otra información importante que describe una matriz es el tipo de elemento del que está

compuesta. Debido a que cada *ndarray* es una colección homogénea de datos exactamente del mismo tipo, cada elemento ocupa el mismo tamaño de bloque de memoria y cada bloque de memoria en la matriz se interpreta de exactamente la misma manera.

3.3.3 Sklearn - LabelEncoder

La función LabelEncoder proviene de la librería Sklearn de Python. Se utiliza para codificar las etiquetas de una característica categórica en valores numéricos que van desde 0 hasta el número de clases - 1. [17]

3.3.4 Datetime

Datetime es una librería de Python que proporciona clases para manipular fechas y tiempo. [18]

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from datetime import datetime, timedelta
```

Figura 13: Librerías usadas

4. Implementación

Para empezar, se ha descargado el dataset que se va a utilizar para entrenar el sistema de recomendación. Como hemos mencionado anteriormente, el dataset que se ha usado fue proporcionado por una empresa de servicios online, y estos datos se sacan de Kibana.

Para más información sobre los datos podéis consultar el capítulo 2.6.

4.1 Datasets

El dataset que nos hemos descargado contiene datos sobre los usuarios, película y vuelo. Le hemos puesto de nombre *data,* tiene 4665 filas y contiene las siguientes columnas:

- User ID: *String* identificador de un usuario.
- Day: *Datetime* día que el usuario ha visto la película.
- Name: *String* nombre de la película.
- Gender: *String* género del usuario. Esta puede ser:
    - Male
    - Female
    - Not specified
- Birthyear: *Datetime* fecha de nacimiento del usuario.
- Country: *String* nacionalidad del usuario.
- Origin ICAO: *String* aeropuerto origen del usuario.
- Destination ICAO: *String* aeropuerto destino del usuario.
- Progress: *Float* segundos vistos de la película por el usuario.
- Duration: *Float* duración de la película en segundos.

Estas son las columnas que tiene el dataset cuando se descarga de Kibana. En la Figura 14 podemos ver un ejemplo de un dataset.

Además de *data,* hemos descargado otra tabla, que llamaremos *icao_iata*. Básicamente, se trata de una tabla separada de *data* que contiene los aeropuertos en formato ICAO e IATA de

la aerolínea correspondiente. Su propósito es convertir los códigos ICAO a IATA, ya que en la empresa solo se usa IATA. Esta tabla tiene las siguientes columnas:

- ICAO: *String* código ICAO del aeropuerto.
- IATA: *String* código IATA del aeropuerto.

En la Figura 15 podemos observar un ejemplo de la tabla *icao_iata*.

Name	Day	User ID	Gender	Birthyear	Country	Origin ICAO	Destination ICAO	Progress	Duration
The Witches	2023-04-01	2336	Female	1-Jan-1979 12:0	Mexico	MMQT	MMUN	3,276.40	6,000.00
The Witches	2023-04-01	3958	Female	1-Jan-1979 12:0	Mexico	MMMY	MMQT	364.878	6,000.00
The Witches	2023-04-01	1114	Female	1-Jan-1982 12:0	Mexico	MMUN	SKBO	2,779.64	6,000.00
The Witches	2023-04-01	4218	Female	1-Jan-1987 12:0	Mexico	MMMY	MMQT	159.852	6,000.00
The Witches	2023-04-01	446	Female	1-Jan-1979 12:0	Mexico	MMQT	MMUN	3,378.05	6,000.00
The Witches	2023-04-01	6632	Female	1-Jan-1979 12:0	Mexico	MMMY	MMQT	664.115	6,000.00
The Witches	2023-04-01	4005	Female	1-Jan-2002 12:0	Mexico	ERR	ERR	2,658.44	6,000.00
The Witches	2023-04-01	1955	Male	1-Jan-1992 12:0	Mexico	ERR	ERR	965.873	6,000.00
The Witches	2023-04-01	6447	Female	1-Jan-1995 12:0	Mexico	ERR	ERR	895.567	6,000.00
The Witches	2023-04-01	1306	Female	1-Jan-1982 12:0	Mexico	MMQT	MMUN	543.522	6,000.00
The Witches	2023-04-01	3214	Female	1-Jan-1987 12:0	Mexico	MMMY	KIAH	241.882	6,000.00
The Witches	2023-04-01	1814	Male	1-Jan-1965 12:0	Mexico	MMMY	KIAH	80.818	6,000.00
The Witches	2023-04-01	5265	Female	1-Jan-1975 12:0	Mexico	ERR	ERR	63.25	6,000.00
The Witches	2023-04-01	2196	Female	1-Jan-1987 12:0	Mexico	MMMY	MMQT	48.625	6,000.00
The Witches	2023-04-01	455	Female	1-Jan-1979 12:0	Mexico	MMMY	MMQT	0	6,000.00
The Witches	2023-04-01	4010	Prefer not to say	1-Jan-1972 12:0	Mexico	ERR	ERR	25	6,000.00
The Witches	2023-04-01	792	Prefer not to say	1-Jan-1999 12:0	United States of	MMTJ	MMGL	0	6,000.00
The Witches	2023-04-02	3209	Male	1-Jan-1996 12:0	Mexico	ERR	ERR	2,395.98	6,000.00
The Witches	2023-04-02	3730	Female	1-Jan-1987 12:0	United States of	ERR	ERR	2,299.24	6,000.00
The Witches	2023-04-02	3375	Female	1-Jan-1994 12:0	Mexico	ERR	ERR	2,035.68	6,000.00
The Witches	2023-04-02	1621	Prefer not to say	1-Jan-1982 12:0	Mexico	MMUN	MMMY	1,852.24	6,000.00
The Witches	2023-04-02	4994	Male	1-Jan-1969 12:0	United States of	MMMY	MMUN	478.828	6,000.00

Figura 14: Ejemplo dataset *data*

	A	B
1	**ICAO**	**IATA**
2	KAUS	AUS
3	KBNA	BNA
4	KCMH	CMH
5	KCVG	CVG
6	KDFW	DFW
7	KDTW	DTW
8	KEWR	EWR
9	KHRL	HRL
10	KIAD	IAD
11	KIAH	IAH
12	KJFK	JFK
13	KLAS	LAS
14	KLAX	LAX
15	KMCI	MCI
16	KMIA	MIA
17	KORD	ORD
18	KPHL	PHL
19	KPIT	PIT
20	KSAT	SAT

Figura 15: Ejemplo tabla *icao_iata*

4.2 Procesamiento de datos

Una vez se han descargado los datos. Procederemos a cargar el dataset en el Jupyter Notebook y a tratarlo.

Primero de todo, necesitamos añadir los códigos icao a iata. Y para lograr ese objetivo, usaremos el método merge de pandas. Como sabemos que las dos tablas comparten una columna que contiene los aeropuertos en formato ICAO, creamos un DataFrame nuevo llamado merged_df que sea el resultado del merge entre los dos DataFrames data e *icao_iata* utilizando la columna "Origin ICAO" en el primero e "ICAO" en el segundo como clave de unión. Se realiza nuevamente otro merge entre merged_df e icao_iata, pero esta vez para añadir la columna "Destination IATA" al DataFrame. El merge utilizará las columnas "Destination ICAO" de data e "ICAO" de icao_iata. Con esto lograremos añadir dos columnas nuevas al DataFrame que renombraremos "Origin IATA" y "Destination IATA".

```
def load_data():

    data = pd.read_csv('Data.csv')
    data = data.fillna(0)

    print("\nBase data")
    display(data)

    icao_iata = pd.read_csv('ICAO - IATA.csv')
    print("\nICAO - IATA")
    display(icao_iata)

    return data, icao_iata
```

Figura 15: Datasets

Luego procederemos a crear la columna de "Route", que representa la ruta del vuelo, y está creado a partir de las columnas "Origin IATA" y "Destination IATA". Para crear la columna "Route", haremos un apply(sorted, axis=1) sobre estas dos columnas. El sorted nos garantiza que los códigos IATA estén ordenadas alfabéticamente.

df['Route'] = df[['Origin IATA', 'Destination IATA']].apply(sorted, axis=1).apply(lambda x: '({}-{}) ({}-{})'.format(x[0], x[1], x[1], x[0])).str.join('')

Esta función aplica una lambda[10] a nivel de fila para crear una cadena de texto en el formato que queremos. La función toma los códigos IATA ordenados y los utiliza para construir una cadena que representa la ruta en el formato "(Origin-Destination) (Destination-Origin)", como podemos observar en la Figura 16.

```python
def add_route(df, icao_iata):
    # Create the Route Column
    merged_df = pd.merge(df, icao_iata, left_on='Origin ICAO', right_on='ICAO')
    merged_df = merged_df.rename(columns={'IATA': 'Origin IATA'})

    merged_df = pd.merge(merged_df, icao_iata, left_on='Destination ICAO', right_on='ICAO')
    merged_df = merged_df.rename(columns={'IATA': 'Destination IATA'})

    #Drop these columns
    merged_df.drop(["ICAO_x", "ICAO_y"], axis=1, inplace=True)

    df = merged_df

    #Create the column Route from the iata codes in the format we want
    df['Route'] = df[['Origin IATA', 'Destination IATA']].apply(sorted, axis=1).apply(lambda x: '({}-{}) ({}-{})'.format(x[0], x[

    print("\nAdd route")
    display(df)

    return df
```

Figura 16: Añadiendo “Route” al dataset.

Una vez tengamos creada la columna creada, vamos a añadir otra columna más, llamada “Visits” que contendrá la cantidad de visitas que tiene cada película. Para conseguir esto es muy sencillo, tan solo hay que hacer un count de la cantidad de “User ID” por cada “Name”.

```python
# Create a column with amount of visits for each movie
def add_visits(df):
    movie_counts = df.groupby('Name')['User ID'].count().reset_index()
    movie_counts = movie_counts.rename(columns={'User ID': 'Visits'})
    df = pd.merge(df, movie_counts, on='Name', how='left')
    print("\nAdd Visits")
    display(df)
    return df
```

Figura 17: Añadiendo “Visits” al DataFrame

Como ya dijimos en un capítulo anterior, en este dataset no tenemos ninguna columna de puntuación del usuario, como ratings. Es por eso que como sustituto, he creado una nueva columna llamada “Liked”, qué tendrá de valor 0 si no le ha gustado y 1 si le ha gustado. Esta columna será calculada a partir de las columnas “Progress” y “Duration”, y la lógica detrás es que si el usuario tiene cierto porcentaje de la película vista, lo consideraremos que le ha gustado.

Entonces, primero necesitamos crear una columna llamada “% Watched”, que contendrá el porcentaje de película vista por el usuario. Esto se calculará dividiendo “Progress” con

“Duration” y multiplicando por 100, como podemos observar en la Figura 18. Pero, encontré un problema al intentar crear esta columna, y es que estas dos columnas estaban en formato *String* por lo que era imposible hacer una operación, y por eso se tuvo que convertir a *float* primero, quitando la coma y luego usando la función de *astype()* de python convertirlo en *float*.

```
# Create a column with percentage of movie watched
def add_watched(df):
    df['Progress'] = df['Progress'].str.replace(',', '').astype(float)
    df['Duration'] = df['Duration'].str.replace(',', '').astype(float)
    df['% Watched'] = df['Progress'] / df['Duration'] * 100
    print("\nAdd % Watched")
    display(df)
    return df
```

Figura 18: Añadiendo “% Watched” al DataFrame

Otra variable que creo que puede llegar a influenciar la preferencia del usuario es su edad. Es por eso que se descargó la fecha de nacimiento de cada usuario, ya que en la base de datos no teníamos disponible edad. Entonces, para calcular la edad actual de los usuarios es tan sencillo como hacer una resta del año actual con la columna “Birthyear”. Pero antes, se ha tenido que convertir la columna a *Datetime*, porque su formato era un *String*. Y para conseguir el año actual, he usado la librería *datetime* de Python.

Una vez creada la columna “Age”, he creado otra columna llamada “Age_range”. Esta decisión se ha tomado porque en la empresa preferimos trabajar con rangos de edad, ya que este resulta más conveniente que utilizar la edad exacta como variable. La utilización de rangos de edad permite una mejor segmentación de los datos y facilita el análisis al agrupar individuos en categorías más amplias, como por ejemplo, jóvenes, adultos y personas mayores. Esto permite obtener una visión más general y comprensible de la distribución de edades en el conjunto de datos.

Para lograrlo, tan solo hemos creado dos listas con los rangos y los nombres. Estas son:

- 0-18: Los niños.
- 19-25: Los jóvenes.
- 26-35: Los adultos jóvenes.
- 36-50: Los adultos.
- 50+: Los mayores.

Y a partir de estos rangos, hemos usado la función *cut()* para dividir las variables de la columna "Age" en los intervalos y etiquetas que hemos asignado en la Figura 19.

```
# Create two columns with User Age and Age range
def add_age(df):
    # Calculate the user's age
    df['Birthyear'] = pd.to_datetime(df['Birthyear'])
    current_year = datetime.now().year
    df['Age'] = current_year - df['Birthyear'].dt.year

    # Age ranges
    age_bins = [0, 18, 25, 35, 50, float("inf")]
    age_labels = ['0-18', '19-25', '26-35', '36-50', '50+']

    # Create age range column
    df['Age_range'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)

    print("\nAdd Age")
    display(df)

    return df
```

Figura 19: Añadir "Age" y "Age_range" al DataFrame.

Siguiendo la discusión previa, hace falta añadir una columna "Liked", con el fin de identificar si al usuario le ha gustado una película. Para lograrlo, se ha establecido un criterio basado en el porcentaje de visualización de la película, la columna "% Watched". Por ejemplo, si un usuario ha visto al menos el 50% de la película, lo tomaremos como que le ha gustado. Este enfoque permitirá tener una medida objetiva para evaluar si los usuarios han mostrado interés y aprecio por una película en particular.

Para determinar el porcentaje de visualización en la columna "% Watched", es necesario explorar todos los valores presentes en la columna. Una forma de obtener un resumen estadístico de la columna y analizar su distribución es utilizando la función *describe()* de Python. Esta función proporciona una descripción estadística resumida de una serie o columna, incluyendo el recuento de valores, la media, la desviación estándar, los valores mínimo y máximo, así como los percentiles.

Nuestro objetivo aquí es estudiar los percentiles de la columna. Podemos observar como es la distribución de los porcentajes de visualización, y determinar qué valor nos servirá para determinar si a un usuario le ha gustado la película.

Al estudiar los percentiles, podemos obtener una visión más completa de cómo se distribuyen los datos y qué porcentaje de visualización representa un umbral significativo para considerar si a un usuario le ha gustado una película. Al identificar el percentil

adecuado, podremos establecer un criterio objetivo para asignar la etiqueta "Liked" a cada usuario en función de su porcentaje de visualización.

Como podemos observar en la Figura 21, en esta aerolínea y mes en particular, el porcentaje de visualización de las películas ha sido relativamente bajo, con el 50% de las visitas, habiendo visto menos del 13.3% de la duración total de las películas.
Como estos valores son demasiados bajos, decidí ver los percentiles de los porcentajes mayores que el percentil 75%, y como podéis ver, estos percentiles son mucho más razonables.

Finalmente, se decidió coger el percentil 50% del segundo *describe(),* 48.73 como valor que se usará para decidir si un usuario le ha gustado una película.

```
# Create a column with an Integer of 0 or 1, 0 means the user liked the movie 1 means they didn't like it
def add_liked(df):
    # Decide % using Percentiles
    print("\nPercentile % Watched")
    print(df['% Watched'].describe())
    print("\nPercentile % Watched > 36.21")
    print(df[df["% Watched"] > 36.21]["% Watched"].describe())

    df['Liked'] = df['% Watched'].apply(lambda x: 0 if x > 48.73 else 1) #0 Liked, 1 Dislike

    print("\nAdd Liked")
    display(df)

    return df
```

Figura 20: Añadiendo “Liked” al DataFrame

```
df['% Watched'].describe()
```

```
count    4665.000000
mean       21.845092
std        22.902491
min         0.000000
25%         2.791107
50%        13.329206
75%        36.215524
max       100.333333
Name: % Watched, dtype: float64
```

```
df[df["% Watched"] > 36.21]["% Watched"].describe()
```

```
count    1168.000000
mean       55.583217
std        15.760395
min        36.213619
25%        44.582575
50%        48.727863
75%        65.494572
max       100.333333
Name: % Watched, dtype: float64
```

Figura 21: Percentiles de “% Watched”

Como resultado, nos ha quedado que ha habido 584 Likes en total y 4081 Dislikes.

```
counts_df = pd.DataFrame(df['Liked'].value_counts()).reset_index()
counts_df.columns = ['Value', 'Count']
counts_df
```

	Value	Count
0	1	4081
1	0	584

Figura 22: Resumen “Liked”

4.3 Correlación de las variables con “Liked”

Antes de construir la matriz de similitud, es necesario identificar las variables que son más relevantes en términos de su influencia en la columna "Liked". Para ello, vamos a realizar un análisis de correlación entre cada variable y “Liked”.

En este caso, se ha utilizado el coeficiente de correlación de Pearson. Como hemos mencionado anteriormente, esta correlación es una medida estadística que evalúa la relación lineal entre dos variables. Puede variar entre -1 y 1, donde -1 indica una correlación negativa, 0 una correlación neutra y 1 indica una correlación positiva.

Al calcular el coeficiente de correlación de Pearson para cada variable en relación con "Liked", se podrá determinar la fuerza y la dirección de la relación entre ambas. Aquellas variables que muestren un coeficiente de correlación alto indicarán una mayor influencia en "Liked".

Pero antes de calcular el coeficiente de correlación de Pearson, necesitamos convertir las variables categóricas[9], como “Country” o “Route” a variables numéricas. Y para lograr esto, podemos usar un método de codificación como el One Hot Encoding. En mi caso he usado el LabelEncoder, que codifica las variables con valores entre 0 y n_classes-1. En la Figura 24 podemos observar un ejemplo.

Una vez hemos convertido todas las variables categóricas a numéricas, procedemos a analizar las variables usando el coeficiente de correlación de Pearson. Y como podemos observar en la Figura 25, los resultados de este análisis nos indica que las variables “Age_range”, “Age” y “Route” son las más relevantes.

- Age_range: 0.049133566682167736
- Age: 0.04644408276884623

- Route: 0.0449552680784489

	Name	Day	User ID	Gender	Birthyear	Country	Origin ICAO	Destination ICAO	Progress	Duration	...	Visits	Age	Age_range	% Watched	Liked
0	The Witches	2023-04-01	2336	Female	1979-01-01	Mexico	MMQT	MMUN	3276.400	6000.0	...	74	44	36-50	54.606667	0
1	The Witches	2023-04-01	446	Female	1979-01-01	Mexico	MMQT	MMUN	3378.050	6000.0	...	74	44	36-50	56.300833	0
2	The Witches	2023-04-01	1306	Female	1982-01-01	Mexico	MMQT	MMUN	543.522	6000.0	...	74	41	36-50	9.058700	1
3	The Witches	2023-04-30	1486	Male	1992-01-01	Mexico	MMQT	MMUN	385.736	6000.0	...	74	31	26-35	6.428933	1
4	The Witches	2023-04-30	350	Female	1993-01-01	Mexico	MMQT	MMUN	1993.280	6000.0	...	74	30	26-35	33.221333	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4660	Black Adam	2023-04-16	5416	Male	2001-01-01	Mexico	MSLP	KIAD	536.152	7500.0	...	651	22	19-25	7.148693	1
4661	Black Adam	2023-04-19	3704	Male	1998-01-01	Mexico	MSLP	KIAD	26.000	7500.0	...	651	25	19-25	0.346667	1
4662	Black Adam	2023-04-23	298	Male	1984-01-01	Mexico	MSLP	KIAD	880.375	7500.0	...	651	39	36-50	11.738333	1
4663	Argo	2023-04-02	6900	Male	1981-01-01	United States of America	MSLP	KIAD	2423.830	7200.0	...	251	42	36-50	33.664306	1
4664	Argo	2023-04-23	4752	Male	1984-01-01	Mexico	MSLP	KIAD	357.750	7200.0	...	251	39	36-50	4.968750	1

Figura 23: DataFrame procesado

```python
def correlations(df):

    # Using pearson correlation, calculate the correlation between the variables and the Liked.
    correlation_gender = abs(df['Liked'].corr(df['GenderCode'], method='pearson'))
    correlation_nationality = abs(df['Liked'].corr(df['NationalityCode'], method='pearson'))
    correlation_age = abs(df['Liked'].corr(df['Age'], method='pearson'))
    correlation_age_range = abs(df['Liked'].corr(df['Age_range_code'], method='pearson'))
    correlation_route = abs(df['Liked'].corr(df['RouteCode'], method='pearson'))
    correlation_name = abs(df['Liked'].corr(df['NameCode'], method='pearson'))

    # Show all the correlations
    print("CORRELATION WITH LIKED\n")

    print("Correlation gender:", correlation_gender)
    print("Correlation nationality:", correlation_nationality)
    print("Correlation age:", correlation_age)
    print("Correlation age range:", correlation_age_range)
    print("Correlation route:", correlation_route)
    print("Correlation movie name:", correlation_name)

    print("\nTop 3 Correlations:")
    correlations = {
        "Gender": correlation_gender,
        "Country": correlation_nationality,
        "Age": correlation_age,
        "Age_range": correlation_age_range,
        "Route": correlation_route,
        "Name": correlation_name
    }

    # Get the top 3 best correlations
    top_3 = pd.Series(correlations).nlargest(3)
    print(top_3)

    return [var for var in top_3.index.tolist()]
```

Figura 24: LabelEncoder (Código)

```
CORRELATION WITH LIKED

Correlation gender: 0.01789947424155238
Correlation nationality: 0.02613015102349606
Correlation age: 0.04644408276884623
Correlation age range: 0.049133566682167736
Correlation route: 0.0449552680784489
Correlation movie name: 0.019624576029241367

Top 3 Correlations:
Age_range    0.049134
Age          0.046444
Route        0.044955
dtype: float64
```

Figura 25: Análisis de correlación de las variables

A continuación procedemos a crear un DataFrame nuevo que le llamaremos *new_df* y le pondremos las columnas relevantes, que serían:

- User ID: *String*.
- RouteCode: *Integer* variable codificada de “Route”.
- Age_range_code: *Integer* variable codificada de “Age_range”.
- Age: *Integer*.
- NationalityCode: *Integer* variable codificada de “Country”.

Y finalmente, podemos crear la matriz de similitud. Para crearla, hemos creado una DataFrame llamada *similarity_matrix*, el cual se ha establecido con "User ID" como índice y columna. Y se ha usado la medida de similitud del coseno para calcular la similitud entre los usuarios. Como hemos explicado en un capítulo anterior, esta similitud es una medida estadística que evalúa la similitud entre dos vectores en función del ángulo entre ellos.

Al utilizar la similitud del coseno, se ha creado una matriz cuadrada que representa la similitud entre cada par de usuarios. Los valores son una representación de qué tan similares son, y cuanto más altos sean, más similares serán.

En la Figura 26 podemos ver una primera versión del código de la matriz. Esta desafortunadamente se tuvo que descartar debido a que era ineficiente, porque, debido al tamaño de nuestro dataset, este tardaba demasiado en ejecutarse. Por lo tanto, si miráis la Figura 27, podremos ver una versión más optimizada de la matriz. Las dos versiones tienen la misma función y crean la misma matriz de similitud. En la Figura 28 se puede ver la matriz de similitud resultante.

```
similarity_matrix = pd.DataFrame(index=new_df['User ID'], columns=new_df['User ID'])

for i in range(len(new_df)):
    for j in range(i, len(new_df)):
        # If same user, then similarity = 0
        if i == j:
            similarity_matrix.iloc[i, j] = 0
        # Cosinus similarity
        else:
            similarity = np.dot(new_df.iloc[i, 1:], new_df.iloc[j, 1:]) / (np.linalg.norm(new_df.iloc[i, 1:]) * np.linalg.norm(ne
            similarity_matrix.iloc[i, j] = similarity
            similarity_matrix.iloc[j, i] = similarity
```

Figura 26: Creación de una matriz de similitud usando similitud del coseno (Antiguo)

```python
def create_similarity_matrix(df):
    # Extract the features: RouteCode Age_range_code Age NationalityCode Liked
    user_features = df.iloc[:, 1:].values

    # Cosinus similarity
    coseno = np.dot(user_features, user_features.T) / np.outer(np.linalg.norm(user_features, axis=1), np.linalg.norm(user_featur

    # Create the matrix
    similarity_matrix = pd.DataFrame(coseno, index=df['User ID'], columns=df['User ID'])

    # Set diagonal values to 0. User should have 0 similarity to themselves
    np.fill_diagonal(similarity_matrix.values, 0)

    return similarity_matrix
```

Figura 27: Creación de una matriz de similitud usando similitud del coseno (Nuevo)

User ID	2336	446	1306	1486	350	2905	1180	2330	6318	1805	...	4426	1733	89	210	5594
User ID																
2336	0.000000	1.000000	0.999236	0.984943	0.982174	0.962997	0.996120	0.962997	0.934652	0.999591	...	0.938454	0.856818	0.884046	0.941741	0.941741
446	1.000000	0.000000	0.999236	0.984943	0.982174	0.962997	0.996120	0.962997	0.934652	0.999591	...	0.938454	0.856818	0.884046	0.941741	0.941741
1306	0.999236	0.999236	0.000000	0.990535	0.988309	0.971976	0.998697	0.971976	0.946728	0.999928	...	0.950078	0.874477	0.900008	0.953042	0.953042
1486	0.984943	0.984943	0.990535	0.000000	0.999880	0.995025	0.996182	0.995025	0.981970	0.988826	...	0.983779	0.932698	0.951185	0.985463	0.985463
350	0.982174	0.982174	0.988309	0.999880	0.000000	0.996415	0.994740	0.996415	0.984758	0.986416	...	0.986432	0.938152	0.955839	0.987968	0.987968
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5416	0.856818	0.856818	0.874477	0.932698	0.938152	0.963617	0.897976	0.963617	0.983945	0.868600	...	0.982105	1.000000	0.998432	0.980254	0.980254
3704	0.873385	0.873385	0.890018	0.944108	0.949080	0.971933	0.912005	0.971933	0.989305	0.884492	...	0.987762	0.999452	0.999700	0.986226	0.986226
298	0.935047	0.935047	0.946997	0.981986	0.984787	0.995501	0.962113	0.995501	0.999714	0.943071	...	0.999953	0.983880	0.992313	0.999812	0.999812
6900	0.944911	0.944911	0.955890	0.987041	0.989400	0.997747	0.969588	0.997747	0.999319	0.952295	...	0.999815	0.978330	0.988327	0.999954	0.999954
4752	0.935047	0.935047	0.946997	0.981986	0.984787	0.995501	0.962113	0.995501	0.999714	0.943071	...	0.999953	0.983880	0.992313	0.999812	0.999812

Figura 28: Matriz de similitud

Entonces, una vez llegados a este punto, podemos empezar a montar el sistema de recomendación.

Primero, hemos creado un método llamado *get_similar_users()* que tiene como parámetros:

- Similarity_matrix: *DataFrame* la matriz de similitud.
- User_id: *String* id del usuario al que queremos recomendar.
- Top: *Integer* la cantidad de usuarios más similares que queremos coger. Tiene como valor por defecto 5.

Este método básicamente recibe un *user_id* y la matriz de similitud, busca dentro de la matriz los valores de similitudes del usuario al que queremos recomendar películas, y de entre todos estos usuarios, coge el top 5 usuarios más similares al usuario.

Luego tenemos el método *recommended_movies()*, que, al igual que *get_similar_users()*, también tiene como parámetros:

- Similarity_matrix: *DataFrame* la matriz de similitud.

- User_id: *String* id del usuario al que queremos recomendar.
- Top: *Integer* la cantidad de usuarios más similares que queremos coger. Tiene como valor por defecto 5.

Este método recibe un user_id y una matriz de similitud, y busca los usuarios más similares al usuario objetivo dentro de la matriz llamando al método *get_similar_users()*.

Una vez se han obtenido el top 5 usuarios más parecidos, cogemos todas las películas que han visto estos usuarios y los guardamos dentro de una lista. Para lograr esto, iteramos sobre la lista de usuarios y usando la función extend() de Python, que nos permite guardar todos los elementos de un iterable a la lista [19].

Pero, como no queremos recomendarle al usuario películas que ya ha visto, borramos estas películas de la lista. Para lograrlo, tan solo hemos de coger las películas que ha visto el usuario y borrarlas de la lista de películas que han visto los otros usuarios.

Como cogemos todas las películas de los usuarios similares, se incluyen también películas que consideramos que no les han gustado a los usuarios, es decir, las que tienen como valor 1 en "Liked". Como queremos recomendarle al usuario películas que pensamos que sí que le podrían llegar a gustar, vamos a priorizar las películas que les hayan gustado, usando la función *sorted()*, que ordena la lista según la condición que se le ponga. Hemos ordenado las películas en función de dos críticas, que son:

1. Las películas con 0 en "Liked".
2. Frecuencia de cada película en la lista de películas que han visto los usuarios similares.

De esta manera, nos aseguramos de recomendar siempre las películas que les han gustado a los usuarios similares y las que sean más frecuentes. Finalmente, devolvemos el top 3 películas al usuario.

```python
def get_similar_users(similarity_matrix, user_id, top=5):
    # Get similarity score of the user_id from the matrix
    users_similarity = similarity_matrix.loc[user_id]
    # Get the top x most similar users
    users = users_similarity.drop(user_id).sort_values(ascending=False).index[:top]
    # Print those users to see who they are
    print("\n\nSimilar users: ", users.values)
    # Return those users
    return users


def recommended_movies(df, similarity_matrix, user_id, top=3):
    # Get the most similar users
    most_similar_users = get_similar_users(similarity_matrix, user_id)

    # Collect all the movies those users have seen
    movies = []
    df_users = pd.DataFrame()
    for similar_user in most_similar_users:
        movies.extend(df[df['User ID'] == similar_user]['Name'])
        df_users = df_users.append(df[df['User ID'] == similar_user])

    # Sort the movies by Liked, so the Liked movies are above the disliked ones,
    # also sort by the amount of occurrence of each movie and then finally by % Watched.
    movies_sorted = sorted(movies, key=lambda movie: (df_users[df_users['Name'] == movie]['Liked'].values[0],
                                                      -movies.count(movie),
                                                      -df_users[df_users['Name'] == movie]['% Watched'].values[0]))

    # Remove the movies the user has already seen
    user_rated_movies = df[df['User ID'] == user_id]['Name']

    movies_recommended = []
    for movie in movies_sorted:
        if movie not in movies_recommended and movie not in user_rated_movies.values:
            movies_recommended.append(movie)

    # Get the top x movies
    return movies_recommended[:top]
```

Figura 29: Recomendador

```python
def recommend_movies_for_user(user_id, df, similarity_matrix):
    recommendations = recommended_movies(df, similarity_matrix, user_id)
    print("\nRecommended movies for user ", user_id, ":")
    for i in recommendations:
        print(i)
```

```python
user_id = 2336
recommend_movies_for_user(user_id, df, similarity_matrix)
```

```
Similar users:  [ 446 5063 3154  230 5697]

Recommended movies for user  2336 :
Black Adam
The Fallout
```

Figura 30: Ejemplo de una recomendación

4.4 Tests

Se han realizado pruebas para evaluar la precisión del modelo de recomendación. Durante estas pruebas, se introdujeron usuarios y se intentó recomendar películas a base de los usuarios similares.

Después de generar las recomendaciones, se analizaron las películas recomendadas y se examinaron los perfiles de los usuarios más similares. El objetivo era comprobar si las películas recomendadas eran buenas recomendaciones o no.

Durante el análisis, se evaluaron varios aspectos como el número de películas recomendadas que resultaron ser del agrado de los usuarios similares, la similitud entre los perfiles de los usuarios y las películas que se consideraron para las recomendaciones.

En la Figura 31 vemos los datos sobre el usuario objetivo y en la Figura 32 vemos los datos de los usuarios similares. Podemos ver que de entre todas las películas que han visto los usuarios similares obtenidos al hacer una recomendación, las películas *The Witches, Black Adam y Elvis* han obtenido un me gusta. Por lo tanto, estas películas tendrán más importancia que las demás. Las películas restantes han recibido todas un no me gusta.

Así que, como hay múltiples películas con me gustas, se recomendarán primero las películas que más veces han salido en la lista y si han salido la misma cantidad de veces, se recomendarán las que tengan mayor porcentaje de "% Watched". En este caso, la película *Black Adam* se pondrá en primer lugar porque ha aparecido 2 veces y será seguido por *The Fallout*. Solo se recomendarán estas dos películas debido a que todas las otras ya han sido puntuadas por el usuario.

```
df[df['User ID'] == 2336]
```

	Name	Day	User ID	Gender	Birthyear	Country	Origin ICAO	Destination ICAO	Progress	Duration	...	Visits	Age	Age_range	% Watched	Liked
0	The Witches	2023-04-01	2336	Female	1979-01-01	Mexico	MMQT	MMUN	3276.4	6000.0	...	74	44	36-50	54.606667	0
8	The Batman	2023-04-01	2336	Female	1979-01-01	Mexico	MMQT	MMUN	8480.0	10500.0	...	498	44	36-50	80.761905	0
17	Fantastic Beasts: The Secrets of Dumbledore	2023-04-18	2336	Female	1979-01-01	Mexico	MMQT	MMUN	3276.4	8520.0	...	301	44	36-50	38.455399	1
21	Elvis	2023-04-15	2336	Female	1979-01-01	Mexico	MMQT	MMUN	2276.4	9540.0	...	507	44	36-50	23.861635	1

Figura 31: Usuario objetivo

```
df[(df['User ID'] == 446) | (df['User ID'] == 5063) | (df['User ID'] == 3154) | (df['User ID'] == 230) | (df['User ID
```

	Name	Day	User ID	Gender	Birthyear	Country	Origin ICAO	Destination ICAO	Progress	Duration	...	Visits	Age	Age_range	% Watched	Liked
1	The Witches	2023-04-01	446	Female	1979-01-01	Mexico	MMQT	MMUN	3378.05	6000.0	...	74	44	36-50	56.300833	0
9	The Batman	2023-04-01	446	Female	1979-01-01	Mexico	MMQT	MMUN	3592.00	10500.0	...	498	44	36-50	34.209524	1
52	The Fallout	2023-04-15	230	Female	1981-01-01	Mexico	MMMY	MMUN	2480.50	5520.0	...	269	42	36-50	44.936594	1
864	Elvis	2023-04-18	5697	Male	1981-01-01	Mexico	MMUN	MMMY	4929.58	9540.0	...	507	42	36-50	51.672746	0
4208	Black Adam	2023-04-14	5063	Male	1978-01-01	Mexico	MMUN	MMRX	2026.48	7500.0	...	662	45	36-50	27.019733	1
4209	Black Adam	2023-04-15	3154	Male	1978-01-01	Mexico	MMUN	MMRX	5065.11	7500.0	...	662	45	36-50	67.534800	0

Figura 32: Usuarios similares

5. Conclusión

En esta sección hablaremos sobre las conclusiones finales del proyecto y se evaluará si se han llegado a completar los objetivos propuestos.

El objetivo que se propuso al principio del trabajo, era desarrollar un sistema de recomendación para una empresa de servicios online, cuya plataforma tenía falta de uno.

El trabajo se puede considerar parcialmente exitoso, porque hace su propósito, que es recomendar películas a los usuarios. Pero, al trabajar con un dataset estático, nos encontramos con el problema de los usuarios nuevos. Estos, al no tener ningún tipo de historial y el hecho de que no poseemos información sobre él/ella en nuestro dataset, se hace muy difícil recomendarle películas. Se podría conseguir si se simulasen sus datos, es decir, darle datos auxiliares para intentar recomendarle algo. O tan solo buscar usuarios que hayan visto las mismas películas que él/ella y hacerle recomendaciones, pero este caso no sería muy eficiente.

Para poder solucionar de manera eficiente este problema, sería si tuviésemos conexión directa a la base de datos de la empresa, de esta manera, cada vez que aparezca un usuario nuevo, tan solo debemos de coger sus datos de la base de datos y actualizar nuestro dataset.

Bibliografía

[1] Janos. (2022). *History of recommender systems: overview of information filtering solutions*. Onespire - SAP and IT services. https://onespire.net/news/history-of-recommender-systems/#

[2] *Aprendeia.* https://aprendeia.com/sistema-de-recomendaciones-inteligencia-artificial/

[3] *Intef.* https://descargas.intef.es/recursos_educativos/ODES_SGOA/Bachillerato/TeI/7B6_SA_Recomendacion_a_tu_servicio/sistema_basado_en_filtrado_colaborativo.html#:~:text=Un%20filtrado%20colaborativo%2C%20es%20el,%2C%20fuentes%20de%20datos%2C%20etc

[4] Mendoza Olguín, Gustavo Emilio; Laureano de Jesús, Yadira; Pérez de Celis Herrero, María de la Concepción. (2019). *Métricas de similaridad y evaluación para sistemas de recomendación de filtrado colaborativo*. Logroño, La Rioja, España: Universidad de la Rioja.

[5] Fernando Torralba Barrabés. (2016). *MEJORA DE LAS RECOMENDACIONES EN ALGORITMOS CONVERSACIONALES BASADOS EN EXPERIENCIAS*. Barcelona: Universidad de Barcelona.

[6] Paula A. Rodríguez, Néstor D. Duque y Demetrio A. Ovalle. (2016). *Método Híbrido de Recomendación Adaptativa de Objetos de Aprendizaje basado en Perfiles de Usuario*. Colombia: Universidad Nacional de Colombia.

[7] Malawi Med J. (2012). *A guide to appropriate use of Correlation coefficient in medical research.* Blantyre, Malaui: Malawi-Liverpool-Wellcome Trust Clinical Research Programme.https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3576830/#:~:text=There%20are%20two%20 main%20 types,and%20Spearman's%20rank%20correlation%20coefficient.

[8] Wikipedia. *Pearson correlation coefficient*. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

[9] Wikipedia. *Spearman's rank correlation coefficient.* https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

[9] Wikipedia. Similitud coseno. https://es.wikipedia.org/wiki/Similitud_coseno

[11] Challenger-Pérez, Ivet; Díaz-Ricardo, Yanet; Becerra-García, Roberto Antonio. (2014). *El lenguaje de programación Python*. Holguín, Cuba: Centro de Información y Gestión Tecnológica de Santiago de Cuba

[12] Kuchling, Andrew. (1988). *Interview with Guido van Rossum*. Linux Journal. http://www.linuxjournal.com/article/2959

[13] Python, R. (2023). *Jupyter Notebook: An Introduction*. realpython.com. https://realpython.com/jupyter-notebook-introduction/

[14] Wikipedia. *Pandas (software)*. https://en.wikipedia.org/wiki/Pandas_(software)

[15] Wikipedia. *Numpy*. https://es.wikipedia.org/wiki/NumPy

[16] Oliphant, T. E. (2006). *A guide to NumPy (Vol. 1, p. 85)*. USA: Trelgol Publishing.

[17] InteractiveChaos. *LabelEncoder*. https://interactivechaos.com/es/python/function/labelencoder

[18] Python. *datetime — Basic date and time types*. https://docs.python.org/3/library/datetime.html

[19] Python. *5. Data Structures*. https://docs.python.org/3/tutorial/datastructures.html

Definiciones

1. Nearest Neighbor Search: Es un algoritmo usado para clasificar muestras o para predecir.
2. Apache Lucene: Una API de código abierto para recuperación de información.
3. Dataset: Conjunto de datos
4. Variables cuantitativas: Variable estadística que puede expresarse a través de cifras.
5. Lenguajes multiparadigmas: Es aquel lenguaje que permite utilizar diferentes enfoques para resolver problemas
6. ndarray: Representa una matriz multidimensional y homogénea de elementos de tamaño fijo.
7. Ufunc: Una función universal (o ufunc) es una función que opera en ndarrays de elemento por elemento, admitiendo la difusión de matrices, la conversión de tipos y varias otras características estándar. Por decirlo más fácilmente, es una versión vectorizada de una función que toma una cantidad fija de entradas y produce una cantidad fija de salidas.
8. DataFrame: Paneles bidimensionales compuestos por filas y columnas, que permiten destacar las relaciones entre las distintas variables de la serie de datos
9. Variables categóricas: Las variables categóricas son aquellas que no presentan un valor de un número real, sino una categoría; es decir, pueden tomar un valor dentro de un conjunto fijo y limitado de posibles valores, con o sin orden.
10. Lambda: Las expresiones lambda en Python son una forma corta de declarar funciones pequeñas y anónimas