Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DE CIENCIA DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

SISTEMA DE RECOMENDACIÓN BASADO EN LAS INTERACCIONES DE LOS USUARIOS DE UN PORTAL WEB

D. REY DE LEÓN, CHRISTIAN
TUTOR: D.DÍEZ PELÁEZ, JORGE

FECHA: Julio del 2020

ÍNDICE

Índice de ilustraciones

Índice de tablas

1.- INTRODUCCIÓN

En los últimos veinte años se ha presenciado una explosión en la disponibilidad de enormes cantidades de datos correspondientes y, por consiguiente, de interés en aprendizaje automático y aplicaciones estadísticas. Este interés por la ciencia de datos y el machine learning provoca que surjan a una velocidad extraordinaria programas basados en estos campos a medida que la demanda de personas capacitadas en este campo sigue floreciendo. Esta demanda está siendo impulsada por un hecho innegable. Los enfoques provocados por el machine learning producen una nueva cantidad innumerable de percepciones significativas en numerosos campos como los negocios, las ciencias sociales, la medicina, la biología, la física, la astronomía, por nombrar solo algunas entre las muchas que existen.

Esto ha derivado en la búsqueda de algoritmos, el estudio, o el perfeccionamiento de los ya existentes con el fin de conseguir que el tratamiento de estas grandes cantidades de información existentes hoy en día se lleve hacia aplicaciones prácticas en la sociedad actual.

Una gran aplicación práctica es llevada a cabo en el campo de las recomendaciones de películas a los usuarios.

En este Trabajo de Fin de Grado vamos a abordar el problema de los Sistemas de Recomendación desde la perspectiva de los Filtros Colaborativos, es decir, vamos a tratar de predecir cuál sería la valoración que un usuario daría a un producto, pero sin conocer las características que definen al usuario (edad, género, nacionalidad, ...) en la información que tenemos de la interacción que han tenido otros usuarios con los productos.

De esta manera estaremos trabajando con las preferencias que han manifestado los usuarios con los productos y podremos realizar asociaciones como: "el usuario 1 y el usuario 2 han puntuado de manera similar todas las películas que han visto, así que es de es esperar que una película bien valorada por el usuario 1 tendrá también una buena valoración por parte del usuario 2 (aunque este usuario no la haya visto).

2.- OBJETIVO

Mediante el desarrollo de esta memoria, así como de la aplicación que se ha construido para realizar una demostración práctica sobre el IDE "PyCharm", se buscará alcanzar los objetivos establecidos durante el planteamiento de la idea de este Trabajo Final de Grado.

Son los detallados a continuación:

(Objetivo 1) : Identificación de los diferentes sistemas de recomendación.

(Objetivo 2) : Estudio en detalle del funcionamiento de los sistemas de recomendación.

(Objetivo 3) : Estudio del data-set Movie Lens y sus variables.

(Objetivo 4) : Estudio del enfoque óptimo a utilizar para la implementación con un ordenador personal.

(Objetivo 5) : Implementación y desarrollo del motor predictivo de calificaciones.

(Objetivo 6) : Pruebas y test del sistema de recomendación.

(Objetivo 7) : Desarrollo de documentación técnica, y memoria final del proyecto.

Cabe destacar, desgranando y resumiendo los objetivos marcados en el párrafo anterior, que el objeto final del trabajo, es conseguir realizar un estudio que abarque diferentes enfoques de los sistemas recomendación existentes en la sociedad actual y partiendo de la capacidad computacional que requiere cada uno de ellos, se pueda conocer cuál es posible diseñar e implementar en un ordenador personal, siempre partiendo de la capacidad computacional de un ordenador actual de sobremesa, para llevar a cabo la tarea de la predicción del rating que otorgará un usuario a una película desconocida por el mismo, así como la predicción de las similitudes por un lado, entre usuarios y por otro, entre películas.

Se dispone del dataset "*MovieLens*" proporcionado por el grupo de investigación del departamento de ciencias de la computación de la universidad de Minnesota, GroupLens Research Project.

El dataset MovieLens consta de calificaciones realizadas por usuarios sobre un conjunto de películas. Cada usuario viene descrito por por algunas características intrínsecas, como la edad, el género, la profesión. Las películas también están descritas por otro conjunto de características, como el identificador de película, el nombre, la fecha de lanzamiento, etc.

El valor a predecir, se denomina calificación o rating y puede tomar uno de los siguientes valores:

- 1 (no le gusta nada)
- 2
- 3
- 4
- 5 (le encanta)

los cuales indican lo mucho o lo poco que le gustará al usuario, siempre partiendo de que 1 es indicativo de que no le gustará, mientras que 5 es un indicativo de que le encantará.

3.- ANTECEDENTES Y ESTADO DEL ARTE

Actualmente, la cantidad de información a la que puede acceder un usuario de Internet excede sus capacidades humanas para el procesamiento de esa información. Cuando se habla de procesamiento de información, no es más que aquello que un usuario desea encontrar. Se pone como ejemplo, al redactor de dicho Trabajo de Fin de Grado a la hora de la recolección de información para la realización de este. Mediante una simple búsqueda en Internet, encontrará una gran cantidad de artículos relacionados con el tema del trabajo; los sistemas de recomendación, pero únicamente unos pocos cubrirán las necesidades de la investigación.

Otro típico caso, se observa cuando un lector se está iniciando en el mundo de la literatura, y entra en una librería a comprar un libro. Posee cierto desconocimiento de sus gustos literarios, lo cual le llevará a apoyarse sobre las recomendaciones de terceros para tomar la decisión de qué libro comprarse. Pese a que muchas veces esta información puede diferir mucho de los gustos desconocidos del lector, ya que puede ser muy subjetiva.

Debido a las circunstancias descritas, se ha vuelto necesario el desarrollo de productos software automáticos, que produzcan diferentes tipos de recomendaciones para ayudar a los usuarios a encontrar los productos deseados, mediante el filtrado de la información disponible logrando un mejor uso de esta.

Es interesante de cara a cualquier empresa que comercialice cualquier tipo de servicio o producto a usuarios, ya que podrían obtener una mayor vinculación con los usuarios aumentando sus oportunidades de venta mediante las recomendaciones de productos o servicios que se adapten al perfil de compra de los usuarios a los que vayan dirigidos.

A continuación, hablaremos de tres claros ejemplos de proveedores como son: *Netflix*, *Google y Amazon*.

El caso de *Nexflix* se remonta al año 2007 en el que dicha compañía organizó un concurso con un único objetivo; la implementación de un sistema de recomendación que mejorase en un 10% el sistema que poseía *Nexflix* en esos momentos. (*BellKor's Pragmatic Chaos*, s. f.)

Otro claro ejemplo es *Google*, el mayor SR que existe. El buscador Google es un SR, el mayor motor de búsqueda de la web. Y no sería nada, sin su algoritmo **PageRank**, encargado de ordenar páginas de mayor a menor importancia para cada pregunta que introduzca el usuario en el buscador Google.

Funcionamiento del PageRank: un usuario introduce una pregunta en el buscador Google y éste le devolverá una lista ordenada de referencias. Aquí aparece el negocio del posicionamiento, en el que las empresas intentan utilizar el *PageRank* a su favor. (Page, 2001)

El último ejemplo es *Amazon*, Amazon utiliza un sistema de recomendación construido con un enfoque de filtrado colaborativo. Funcionamiento: Depende de la condición del usuario. Si se trata de un usuario nuevo, el SR ofrecerá una recomendación utilizando la metodología de vecindad "ítem a ítem" en función de los productos vistos recientemente por el usuario. En cambio, si se trata de un usuario del que ya se conoce más información, el SR ofrecerá una recomendación utilizando la metodología de vecindad "usuario-usuario", busca usuarios que se posean gustos parecidos y ofrece recomendaciones al usuario en base a esos gustos de esos usuarios parecidos. Amazon ofrece recomendaciones

utilizando la metodología de factores latentes, primero comparando varias características de cada ítem, segundo les asigna un peso para tercero calcular la similitud que existe entre los ítems, apoyándose en aquellos factores que suponen mayor interés para el usuario.(*The History of Amazon's Recommendation Algorithm,* 2019)

4.- CONOCIMIENTOS PREVIOS

El campo de investigación de este proyecto, el aprendizaje automático, está relacionado en una buena parte con el campo de la estadística. Es por ello, necesario tener ciertos conocimientos previos acerca de los enfoques que existen en la construcción de los sistemas de recomendación para alcanzar los objetivos del proyecto.

4.1.- SISTEMAS DE RECOMENDACIÓN

Un **sistema de Recomendación** se puede definir como un algoritmo que aprende intereses y preferencias de los usuarios y ayuda a asociar de manera personalizada un producto con un usuario. Tienen como objetivo el ayudar a conectar **usuarios** e **ítems**. Facilitando a los usuarios, tareas como; encontrar productos u otros usuarios que puedan interesarles, aconsejar a un usuario a encontrar una mezcla de productos nuevos y viejos, así como también ayudar a encontrar productos según la ubicación de un usuario, etc.

Teniendo en cuenta las tareas que los usuarios del **sistema de recomendación** pretenden obtener, se presentan las siguientes funcionalidades que un usuario desea conseguir con este tipo de sistemas :

- Recomendar productos.
- Predecir valoraciones acerca de un determinado producto.
- Realizar recomendaciones a partir de un conjunto de datos conocido.

También, se ha de aclarar que a lo largo del documento trataremos como sinónimos los siguientes conceptos:

- (SR = Sistema de Recomendación)
- (Usuarios = consumidores = clientes)
- (ítem = producto)

Cabe resaltar, que los SR están muy relacionados con los sistemas de búsqueda, pero a su vez tienen ciertas diferencias. Para comenzar, la primera de las diferencias, en los sistemas de búsqueda, sólo se realiza un uso contado de ellos, mientras que los SR están enfocados a un uso repetitivo durante un largo período de tiempo. Otra diferencia fundamental es que en un SR se poseerá el llamado *perfil del usuario*, que no es más que una manera para almacenar los criterios del usuario **obtenidos de forma implícita** para posteriormente ser usados en la realización del filtrado de la información. Mientras que en un sistema de búsqueda espera que los criterios del usuario sean indicados por el usuario de manera explícita cada vez que se hace un uso de este.

4.2.- TIPOS DE SR

Un SR deberá estimar el interés que tiene un usuario sobre cada ítem recomendable, para posteriormente seleccionar aquellos ítems que sean de mayor interés en base a las estimaciones predichas. Lo que formalmente se reduciría en la siguiente función:

$$r': U \times I \rightarrow R$$

Esta función se encarga de realizar la estimación de utilidad esperada para las recomendaciones de un ítem **i** realizadas a un usuario **u**, donde R es el conjunto ordenado de recomendaciones (se trata de un ránking de ítems basado en el interés del usuario sobre los mismos). U se refiere al conjunto de todos los usuarios, mientras que I al conjunto de todos los ítems que pueden ser recomendados.

Para la implementación de esta función **r'**, existen diferentes técnicas. A continuación, se muestra una clasificación de los sistemas de recomendación en función de la técnica seleccionada.

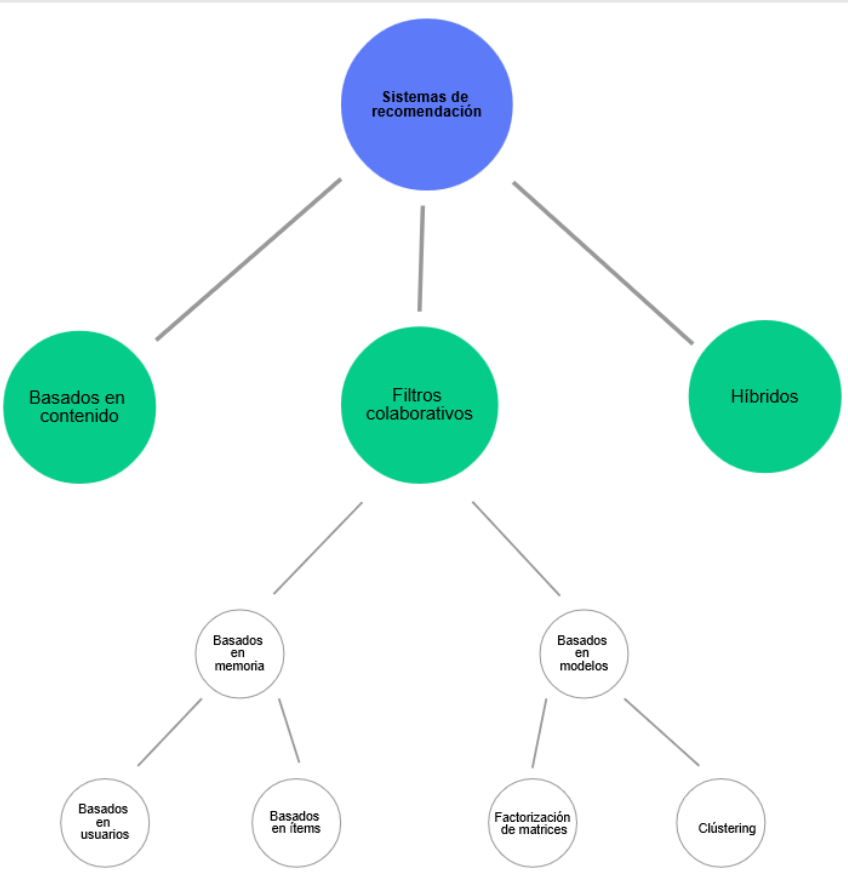

Ilustración 1. Clasificación de los Sistemas de Recomendación

4.2.1.- BASADOS EN CONTENIDO

Los SR que aplican este enfoque, se fundamentan en el tratamiento específico de las características de los ítems, también conocidas como *atributos* o *metadatos*, y no en la valoración del usuario al ítem. Para ello, este tipo de sistemas necesitan de buena descripción del producto, mediante la extracción de sus *atributos* para luego relacionarlos con los de otros productos y así medir la similitud entre cada uno de ellos y posteriormente en conjunto. Los *atributos* de cada ítem se especifican, manualmente o mediante la recolección de información de descripciones del producto, comentarios, etiquetas (tags), imágenes, audios o vídeos de este.

Una vez conocidos los *atributos* o *metadatos* de los ítems, se deberán relacionar con el término mencionado en el apartado introductorio a los SR, el *perfil del usuario*, que poseen todos los SR. El perfil de usuario a grandes rasgos se trata del comportamiento pasado del usuario. Dicho *perfil del usuario* deberá poseer un formato adecuado para poder relacionarlo con los *atributos* de los ítems y así poder obtener una estimación del interés que podría tener un usuario sobre cada uno de los ítems. Recomendando ítems que se encuentren dentro de dicho perfil de usuario.

Perfil de usuario: Se calcula en base a los metadatos, puede expresarse mediante un vector que contiene el comportamiento pasado del usuario.

Este tipo de algoritmos se utilizan métodos como los que siguen (Se detallarán más adelante, cuando se hable de los sistemas de recomendación que aplican el Filtrado Colaborativo)

Cálculo de la similitud del coseno del ángulo formado por el vector del *perfil del usuario* y el vector del *ítem*.

Algoritmo de distancia euclidiana aplica una función matemática que determina la distancia en línea recta de dos puntos en un espacio de n dimensiones. La fórmula para el cálculo de la similaridad mediante dicho algoritmo es la que sigue

Mediante el uso de la distancia euclidiana, se pueden determinar la similitud entre elementos que posean características similares, como es el caso de las calificaciones de los usuarios sobre las películas que hayan visto. Se puede utilizar en un sistema de recomendación de películas, para ver qué películas recomendar a los usuarios.

Correlación de correlación de Pearson (r), arrojará un valor entre [-1 ,1] es apropiada para los casos en los que haya que realizar la comparación entre dos variables, un ejemplo práctico sería para realizar la comparación entre el conjunto de valores reales y el conjunto de valores predichos arrojados por un motor de recomendación. A continuación, se detallan los posibles casos:

- **Correlación menor a cero,** las variables se relacionan inversamente.
- **Correlación igual a cero**, no podemos determinar la covariación. Puede existir relación NO lineal entre las variables.
- **Correlación mayor a cero**, cuanto más cercana a +1 sea, existirá una mayor relación entre las variables.

Estos tres métodos vistos, también se deberán tener en cuenta a la hora de la implementación de sistemas de recomendación basados en la técnica del filtrado colaborativo, que se verá a continuación.

4.2.2.- FILTROS COLABORATIVOS

Esta técnica de sistemas de recomendación será utilizada en el desarrollo e implementación del sistema de recomendación de este trabajo de fin de grado, ya que como se verá más adelante, se parte de un conjunto de datos que aunándolo a filtrado colaborativo, se puede realizar una implementación en un ordenador personal, ya que con otros algoritmos como se verán más adelante, pueden necesitar de una capacidad de cómputo muy superior.

Es un enfoque que se –basa en el tratamiento específico de los **ítems** o el tratamiento específico de los **usuarios**. Se fundamentan en el llamado "Wisdom of the crowd"; conocimiento de la mayoría, para realizar las recomendaciones. Se pone el escenario en el que un usuario realiza una valoración de un ítem puntuándolo de manera implícita o explícita, y el sistema de recomendación se encarga de analizar los gustos futuros de los usuarios basándose en los gustos similares mostrados por ese usuario en el pasado.

El enfoque del filtrado colaborativo se basa en la relación de las entidades: *ítems y usuarios* mediante las valoraciones. Dichas valoraciones podrán ser de dos tipos:

- **Valoración explícita** : Los usuarios puntúan a los ítems.
- **Valoración implícita** : Existen diversos ejemplos de este tipo de valoración. Desde el hecho de que un usuario se compre o no un producto, el historial de búsqueda de un usuario, el historial de navegación, el tiempo que dedica a ver un producto, los artículos que lee ese usuario, etc.

La técnica de filtrado colaborativo funciona mediante la creación de una base de datos (matriz de usuario-ítem) de valoraciones por parte de los usuarios a cada elemento, esto es conocido como **perfil de usuario**. A continuación, se describe mediante el dibujo de dicha matriz.

Perfil de usuario

En este tipo de SR, el perfil de usuario se representa mediante una matriz. Las filas representarán a los usuarios, las columnas a los ítems, y cada entrada será la calificación de un usuario sobre un ítem. Existirán ítems que no poseerán calificaciones, por lo que en esa entrada existirá un hueco.

	i1	i2	i3	i4	i5	
u1	1	1		1	1	1
u2	1			1	1	1
u3	1		1		1	

Ilustración 2 Representación matricial del perfil de un usuario

4.2.2.1.- BASADOS EN MEMORIA

Los elementos que ya fueron calificados por el usuario antes de realizar la recomendación juegan un papel muy importante en la realización de la búsqueda de un vecino que comparta sus gustos. Una vez que se encuentra un vecino de un usuario, se pueden usar diferentes algoritmos para combinar las preferencias entre los vecinos para así generar recomendaciones. Los SR de filtrado colaborativo basados en memoria se pueden lograr de dos maneras a través de dos técnicas, **basado en el usuario**, **o basado en los elementos**.

La primera variante, **vecindad** basado en (usuario – usuario), calcula la similitud entre los usuarios comparando sus calificaciones sobre el mismo **ítem**, y luego predice la calificación de un **ítem** que cree que realizaría el **usuario activo**, como un promedio ponderado de las calificaciones del **ítem** por parte de los **usuarios similares** a al **usuario activo** donde los pesos serán las similitudes entre el **usuario activo** y los **usuarios similares.**

Como caso práctico, supongamos que un **usuario U1** visualizó (la visualización se trata como una puntuación) los **ítems I1, I2, I3, I4, e I5**, y el **usuario U2** vio los **ítems I1, I3, I4, e I5**. Este enfoque considera que los usuarios poseen una gran similitud, ya que el usuario U2 vio cuatro de los cinco ítems vistos por el usuario U1 y, por lo tanto, el usuario U2 es probable que le guste el **ítem I2** . Y aquí es donde entra en juego el sistema de recomendación, recomendando el ítem I2 al usuario U2.

Existen diferentes algoritmos de filtrado colaborativo basados en memoria, pero el más reconocido se trata del algoritmo de los *k-means* (k-vecinos) (Na8, 2018),

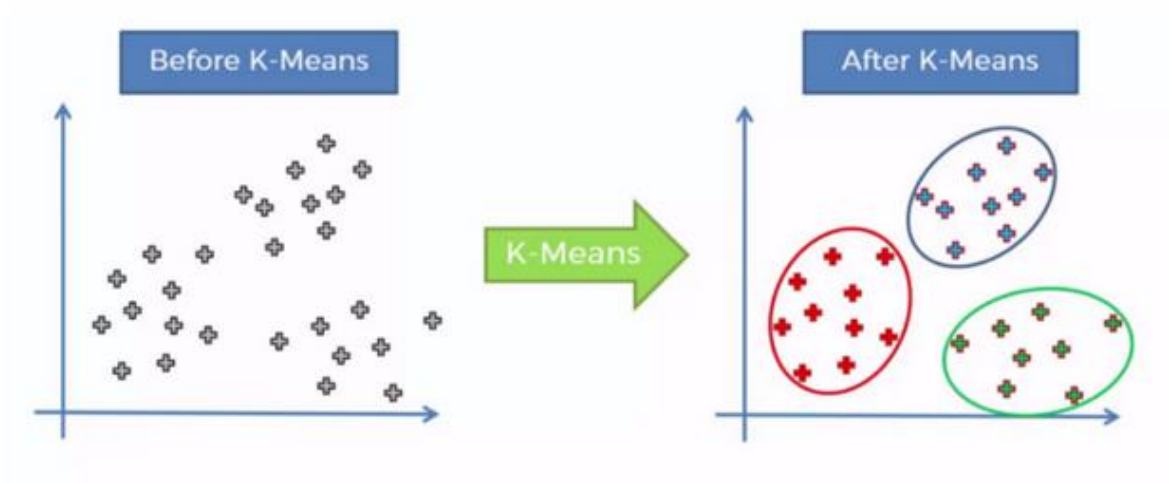

Ilustración 3 Representación gráfica de los centroides formados por el algoritmo K-means

del que a continuación se detallan los pasos de este aplicado a usuarios:

1. Calcular la SIMILITUD entre el usuario **u** al que se le realizarán las recomendaciones y cada uno de los usuarios mediante una función de similitud. Las funciones de similitud más usadas son: *coseno, error cuadrático medio (MAE), correlación de Pearson (COR)* (*Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*, s. f.).

2. Para cada ítem **i** a recomendar al usuario **u**, se buscarán y seleccionarán los usuarios más cercanos, es decir, similares, que también hayan calificado al ítem **i**. Una vez se obtiene un conjunto de usuarios similares, se realiza la estimación de la calificación del ítem **i** (r'_{ui}) mediante una función de AGREGACIÓN.

3. Se recomiendan los ítems que mejores calificaciones hayan obtenido en el paso anterior.

La segunda variante, **vecindad** basado en (ítem – ítem) fue inventada por Amazon.com, construirá una matriz de ítem – ítem que determinará las relaciones entre pares de elementos. Posteriormente se usará esa matriz junto con los datos del usuario (*user*) para intentar inferir sus gustos. Resumiendo, utilizará las calificaciones realizadas por el usuario u sobre ítems similares a i. La diferencia primordial con el anterior tipo de vecindad, (usuario-usuario), es que en este no se busca la vecindad de los usuarios y se va directamente a calcular la similaridad entre ítems, buscando aquellos ítems que sean más parecidos al ítem i que ha sido calificado por el usuario u. En resumen, se construye una matriz de similitudes entre **ítems** mediante la recuperación de todos los **ítems** calificados por el **usuario activo** de la matriz de ítems de usuario. Con esta matriz, se determinará cuán similares son los **ítems** recuperados al **ítem objetivo**. Luego se seleccionan los k **ítems** más similares.

El algoritmos de filtrado colaborativo *k-means* también podrá ser aplicado en el caso de poseer una vecindad "ítem-ítem" (Na8, 2018), que a continuación se detallan los pasos del mismo:

1. Buscar los **k** ítems vecinos de cada **ítem** del sistema.

2. Para cada **ítem *i*** que no haya sido valorado por el **usuario objetivo u**, se predecirá una valoración basándose en las valoraciones que el **usuario objetivo u** ha hecho sobre los k vecinos del **ítem *i*,** mediante una función de agregación.

3. Seleccionar las n mejores recomendaciones para el **usuario objetivo u** a partir de las predicciones realizadas en el **paso 2**.

Ambas variantes hacen uso de matrices de calificaciones, pero lo ideal para un negocio, es que esas matrices de valoraciones sean implícitas como se ha expuesto en el apartado anterior. Es decir, el rellenado de las matrices de valoraciones se basa en observaciones del comportamiento implícito del usuario, como las interacciones que tiene en una página web de e-commerce, ya sea leyendo ese artículo, el tiempo que permanece en una pantalla, qué artículos ha comprado, qué artículos tiene en la cesta esperando a ser facturados, etc. Todas estas observaciones se enfrentan a lo que todos los usuarios han hecho, utilizando estos datos para predecir el comportamiento del usuario en un futuro.

A continuación, se describirán algunas de las **funciones para medir la SIMILITUD** entre **usuarios** o **ítems**.

Índice del coseno

En el caso de querer medir la similitud entre dos **usuarios** o dos **ítems** se aplicará la medida del **índice del coseno** :

Ecuación 1

$$sim_{cos(x,y)} = \frac{\boldsymbol{r_x} \,.\, \boldsymbol{r_y}}{|\boldsymbol{r_x}| \,.\, |\boldsymbol{r_y}|} \; \boldsymbol{\epsilon} \; [\mathbf{0}, \mathbf{1}]$$

donde r_x y r_y representan dos vectores que almacenan las calificaciones realizadas por los usuarios x e y sobre cada ítem en el caso de querer medir la **similitud entre usuarios**, o las calificaciones recibidas por los ítems x e y de cada usuario, en el caso de querer medir la **similitud entre ítems**.

Coeficiente de Pearson ***(para usuarios)***

Nos indica la correlación que existe entre dos ítems. Cuanto mayor sea el resultado, mayor será la similitud (*Interpretar los resultados clave para Correlación*, s. f.). A continuación, se muestra su cálculo:

Ecuación 2

$$sim_{CoefPearson(x,y)} = \frac{\sum_{i \in I}(r_{xi} - \bar{r}_x)(r_{yi} - \bar{r}_y)}{\sqrt{\sum_{i \in I}(r_{xi} - \bar{r}_x)^2}\sqrt{\sum_{i \in I}(r_{yi} - \bar{r}_y)^2}}$$

donde:

- **x** e **y** representan a los dos usuarios.
- **I** representa el conjunto de todos los ítems calificables.
- r_x representa el conjunto de las calificaciones hechas por el usuario **x**
- r_y representa el conjunto de las calificaciones hechas por el usuario **y**
- $\bar{r}_x$ representa la media del usuario **x** sobre r_x
- $\bar{r}_y$ representa la media del usuario **y** sobre r_y

Coeficiente de Pearson ***(para ítems)***
Ecuación 3

$$sim_{CoefPearson(x,y)} = \frac{\sum_{u \in U}(r_{i1\,u} - \bar{r}_{i1})(r_{i2\,u} - \bar{r}_{i2})}{\sqrt{\sum_{u \in U}(r_{i1\,u} - \bar{r}_{i1})^2}\sqrt{\sum_{u \in U}(r_{i2\,u} - \bar{r}_{i2})^2}}$$

donde:

- **i1** e **i2** representan a los dos ítems.
- **U** representa el conjunto de todos los usuarios que calificarán.
- r_{i2} representa el conjunto de las calificaciones realizadas sobre el ítem **i1**
- r_{i2} representa el conjunto de las calificaciones realizadas sobre el ítem **i2**
- $\bar{r}_{i1}$ representa la calificación media sobre el ítem **i1**
- $\bar{r}_{i2}$ representa la calificación media sobre el ítem **i2**

A continuación, se describirán algunas de las **funciones de AGREACIÓN** utilizadas para calcular las predicciones de los usuarios en función de las calificaciones de sus vecinos. Tomarán como entrada un conjunto de datos, y devolverán un único resultado. (Isinkaye et al., 2015):

Media aritmética
Ecuación 4

$$arit(\bar{x}) = \frac{1}{n}\sum_{1 \le i \le n} x_i$$

Media geométrica
Ecuación 5

$$geo(\bar{x}) = \left(\prod_{1 \leq i \leq n} x_i \right)^{\frac{1}{n}}$$

Media armónica

$$h(\bar{x}) = n \left(\sum_{1 \leq i \leq n} \frac{1}{x_i} \right)^{-1}$$

donde $\bar{x}$, se define como el conjunto de valoraciones que se aplicará en la función de agregación seleccionada.

4.2.2.2.- BASADOS EN MODELOS

Los SRs implementados bajo esta técnica, hacen uso de las calificaciones previas de los usuarios para construir un modelo bajo el cual realizar recomendaciones. El proceso de construcción del modelo se puede realizar utilizando algoritmos de ***machine learning*** o ***minería de datos*** como regresión, Clustering, árboles de decisión, factorización matricial, redes bayesianas, redes neuronales, etc. Estos algoritmos, son más eficientes en cuando a la velocidad de recomendación que poseen frente a otros algoritmos de filtrado colaborativo. Pero tienen el inconveniente de la actualización del modelo cada vez que se agreguen nuevos datos (usuarios, ítems, o calificaciones).

Las técnicas basadas en modelos analizan la matriz de elementos de usuario (*perfil de usuario*) para identificar las relaciones entre los elementos, usan estas relaciones para comparar la lista de recomendaciones de recomendaciones top-N. Además de los problemas de escasez asociados con los sistemas de recomendación. Por último, cabe destacar que este tipo de sistemas de recomendación no solo realiza recomendaciones del tipo de qué ítem consumir por los usuarios, sino también cuándo consumir realmente ese ítem. A continuación, se realiza una breve descripción de diferentes algoritmos aplicados en la construcción de los modelos de los SRs.

Regresión

El análisis de regresión se utiliza cuando se piensa que dos o más variables están sistemáticamente relacionadas linealmente. Es un proceso poderoso para analizar las relaciones asociativas entre la variable dependiente y una o más variables independientes. Los usos de la regresión van desde ajuste de curvas, predicción hasta pruebas de hipótesis sistemáticas sobre las relaciones entre variables. El ajuste de curvas puede ser útil para identificar una tendencia dentro del conjunto de datos, ya sea lineal, parabólica o de alguna otra forma.

Clustering

La principal tarea de este tipo de algoritmos trata de buscar agrupaciones de objetos dentro de un conjunto de datos, de modo que los objetos dentro de un grupo son "similares" entre sí y "diferentes a los objetos de los otros grupos. La similitud viene definida en función de cuán "cercanos" están los objetos en el espacio, en función de una función de distancia. La

(*Kużelewska: Advantages of information granulation... - Google Académico*, s. f.). Una vez definidos los grupos, las calificaciones de los usuarios dentro de esos grupos se pueden promediar y utilizar para la realización de recomendaciones hacia un usuario concreto. Este tipo de algoritmos se suele utilizar en como paso previo a otros algoritmos con el fin de reducir el conjunto de candidatos posibles de una recomendación.

K-means vuelve a ser uno de los algoritmos englobados dentro de los diferentes métodos de Clustering. Éste toma como parámetro de entrada un conjunto de n elementos dividiéndolo en K grupos.

Árbol de decisión

Se basa en la metodología de los gráficos de árbol. Se tratan de formas gráficas de representación de todos los sucesos posibles que pueden surgir en función de cada decisión puntual en cierto momento. Su función principal no es más que recomendar la decisión más acertada desde el punto de vista probabilístico.

Se construye analizando un conjunto de tuplas de entrenamiento para los que se conocen sus respectivas **etiquetas de clase**, que no son más que los nodos finales . Luego se aplican para clasificar ejemplos no vistos anteriormente. Si están capacitados en datos de muy alta calidad, tienen la capacidad de hacer predicciones muy precisas . Los árboles de decisión son más interpretables que otros clasificadores, como la máquina de vectores de soporte (SVM) y las redes neuronales porque combinan preguntas simples sobre los datos de una manera comprensible. Los árboles de decisión también son flexibles en el manejo de elementos con una combinación de características reales y categóricas, así como elementos que tienen algunas características faltantes específicas.

Un árbol de decisión poseerá tres elementos fundamentales (*Universidad politécnica de Cartagena. Introducción a los árboles de decisión (2018). Recuperado de http://www.dmae.upct.es/~mcruiz/Telem06/Teoria/arbol_decision.pdf*):

- *Nodo de decisión* : Indicador de **toma de decisión** en ese punto del proceso. Estará representado mediante un cuadrado.
- *Nodo se probabilidad* : Indicador de un **evento aleatorio** en ese punto del proceso. Estará representado mediante un círculo.
- *Rama* : Muestra los caminos que se pueden tomar cuando se produce una **toma de decisión** o un **evento aleatorio**.
- Habría que añadir un cuarto elemento en discordia, los llamados *nodos hojas*, o terminales de un árbol de decisión. Estos nodos, poseen una **etiqueta de clase.**

A continuación, se muestra un ejemplo de un árbol de decisión:

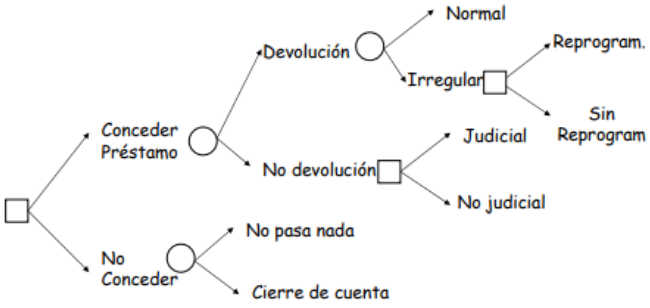

Ilustración 4 Ejemplo de construcción de árbol de decisión: Conceder un préstamo

Red neuronal artificial

Se trata de una estructura formada por varios nodos (*neuronas artificiales*) que se conectan entre sí y forman capas de manera sistemática. Cada conexión entre dos neuronas es lo que se conoce como enlace el cual posee un peso asociado dependiendo de la cantidad de influencia que posea una neurona sobre otra. El valor de salida de la neurona anterior se multiplicará en el enlace por el peso asociado. Además, en la salida de cada neurona, existe lo que comúnmente se llama *función de activación*, que no es más que una función limitadora o umbral que modifica el valor resultado o impone un límite que se debe sobrepasar antes de propagarse a otra neurona.

Una red neuronal artificial tiene la capacidad de estimar funciones no lineales, capturar relaciones complejas entre conjuntos de datos.

4.2.2.2.1- FACTORIZACIÓN MATRICIAL NO NEGATIVA(NMF)

En este apartado se va a profundizar, ya que, en un principio, este Trabajo de Finde Grado había sido orientado a realizarse con los recursos de una empresa y debido a la crisis sanitaria del Covid-19, se ha tenido que posponer para un futuro, por lo que se describirá el procedimiento a nivel teórico.

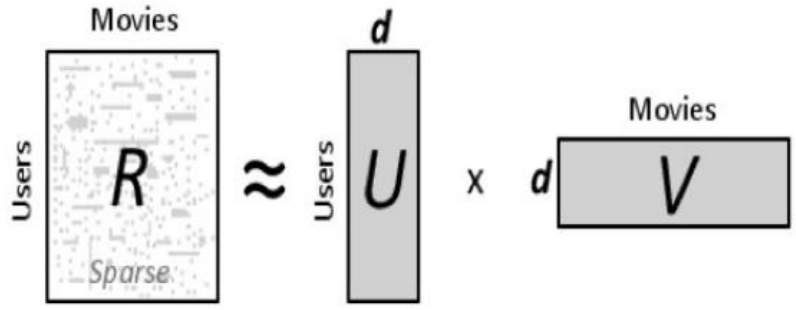

Ilustración 5 Representación esquemática de una factorización matricial

Enfocando la técnica de factorización matricial a un problema de recomendación de películas, se enfocará en la predicción de los valores desconocidos dentro de las matrices de calificaciones de los usuarios sobre los ítems.

Dependen en gran medida de los datos históricos de calificación de los usuarios sobre los ítems. En la mayoría de las ocasiones, la matriz de calificaciones es muy grande y escasa debido a que los usuarios no han calificado la mayoría de los ítems representados dentro de esa matriz. Por lo que se han utilizado diferentes algoritmos de factorización de matrices, para estimar esas entradas incompletas de dicha matriz, entre los que mejores resultados proporciona, se encuentra el algoritmo de **Factorización No-negativa de Matrices** (*Factorización de matrices no negativas (NMF) de Oracle*, 2014)

Pero no sin antes realizar unas pequeñas adaptaciones, ya que la matriz de usuarios-ítems, será una matriz incompleta, también conocida como *sparse* en el mundo de la informática.

En este subapartado, se explica en qué consiste la **Factorización No-Negativa de Matrices**, también conocida como **NMF**. Así como alguna de sus implementaciones algorítmicas. La idea fundamental que reside en este tipo de factorización es conseguir una aproximación de una matriz mediante la multiplicación de otras dos.

Se exponen diferentes medidas de similitud para comparar la matriz aproximada y la original. Así como también una serie de reglas para la actualización de los valores en el algoritmo que garantizan que la similitud entre las dos matrices vaya siempre en aumento.

Otro de los aspectos importantes que se expondrán en este subapartado, es la relación existente entre el conjunto de datos y los llamados *factores o componentes latentes* que se encuentran ocultas en dicho conjunto de datos.

La **Factorización No-Negativa de matrices**, aplicada a también conocida como **NMF**, se trata de una técnica que persigue una aproximación de una matriz **X**, de valores positivos o nulos, mediante la multiplicación de dos matrices **A** y $\mathbf{B^T}$, también de elementos positivos o nulos.

- La matriz **X** tiene una dimensión de **i** x **j**, será la matriz a aproximar.
- La matriz $\widehat{\mathbf{X}}$ tiene una dimensión de **i** x **j** será la matriz aproximada.
- La matriz **A** tiene una dimensión de **i** x **k**.
- La matriz **B** tiene una dimensión de **j** x **k**.
- La matriz $\mathbf{B^T}$ tiene una dimensión de **k** x **j**. (se transpone a B para que las dimensiones encajen).
- La matriz **E** tiene una dimensión de **i** x **j**, será la matriz de error.
- x_{ij} será el elemento **ij** de la matriz **X**
- a_{ik} será el elemento **ik** de la matriz **A**
- b_{jk} será el elemento **jk** de la matriz **B**

La igualdad matemáticamente expresada:

$$X = AB^T + E$$

siendo:

$$x_{ij}, a_{ik}, b_{jk} \geq 0$$

La aproximación de la matriz **X** mediante $\hat{\mathbf{X}}$ viene dada por la siguiente expresión:

$$\boldsymbol{X} \approx \hat{\mathbf{X}} = \boldsymbol{AB^T}$$

Cabe hablar de los llamados **factores** o **componentes latentes**, que son los encargados de determinar el rango de factorización, es decir, la dimensión **k** de las matrices **A** y **B**. Son llamados de esa manera, factores o componentes latentes, debido a que tratan de recomponer la matriz original mediante unas nuevas bases. Además, no aparecen hasta que la **NMF** los construye. Pero usualmente, el **número de factores latentes** se determina para que el número total de elementos de cada matriz de factorización, **A** y **B**, sea menor que el número total de elementos de la matriz original **X**. Lo que matemáticamente expresado sería lo siguiente:

$$\boldsymbol{K(I+J) < KJ}$$

Esta especificación, no se trata de una condición, sino meramente de una recomendación, ya que en el caso de elegir un **número de factores latentes** que no cumpla esa especificación, no incurriría más que en los tiempos de simulación del algoritmo **NMF** haciendo que fueran más pesados que en el caso de que se cumpliera la especificación expuesta.

Cabe la explicación de que la elección del **número de factores latentes** se debe a que el algoritmo **NMF** será aplicado a la matriz original **X** como una comprensión de los datos en los que existirán pérdidas, llamadas error, debido a que se trata de una **aproximación**.

Bien se puede tomar un elemento cualquiera de la matriz original **X** y comparar con la multiplicación del **vector fila i** por el **vector columna** j de las matrices **A** y $\mathbf{B^T}$ respectivamente la ecuación y la diferencia será el **error de la aproximación**. A continuación, se expresa matemáticamente:

$$x_{ij} = \boldsymbol{a_i b_j^T} + e_{ij}$$

- $\boldsymbol{x_{ij}}$ será un elemento **ij** de la matriz **X**
- $\boldsymbol{a_i}$ será el vector *fila* ***i*** de la matriz **A**
- $\boldsymbol{b_j^T}$ será el vector *columna* ***j*** de la matriz $\mathbf{B^T}$

Lo que más detalladamente sería:

$$\boldsymbol{x_{ij} \approx \hat{x}_{ij} = \sum_{k=1}^{K} a_{ik} b_{kj}^T}$$

- $\boldsymbol{x_{ij}}$ será un elemento **ij** de la matriz **X**
- $\boldsymbol{a_{ik}}$ será un elemento **ik** de la matriz **A**
- $\boldsymbol{b_{kj}^T}$ será un elemento **kj** de la matriz $\mathbf{B^T}$

Teniendo en cuenta, que la aproximación se trata de una suma, junto con el hecho de que los elementos son no-negativos, la **NMF** está definiendo cada elemento aproximado por la suma de sus partes, las **componentes latentes**, que en un principio estaban ocultas en el conjunto de datos y ahora se pueden visualizar. De esta suma aproximada, se puede deducir

que cuanto mayor sea el término multiplicativo $a_{ik}b_{kj}^{T}$ tendrá un mayor peso en la suma y eso se traduciría en una mayor afinidad de la componente k con el elemento x_{ij}

Ahora bien, para realizar la factorización matricial, ya sea mediante **NMF** u otro algoritmo, existen diferentes maneras, pero la que se expondrá a continuación, que será la aplicada en el desarrollo e implementación del sistema de recomendación de este Trabajo de Fin de Grado, se basa en un algoritmo iterativo en el que se utilizarán dos términos: **distancia de aproximación a la matriz original**, y **reglas de actualización**

4.2.2.2.1.1.- MÉTRICAS DE SIMILITUD

En los párrafos anteriores se ha estado hablando de una matriz original y una matriz aproximada, por lo que ahora se introducirá el concepto de **distancia**. La distancia será lo que se llama **función de coste**, que expresará la diferencia entre las dos matrices mediante un único número. Matemáticamente hablando, sería la suma de las distancias que exista entre cada elemento de la matriz original y la matriz aproximada:

Ecuación 6

$$DISTANCIA(X,\widehat{X}) = \sum_{ij} dist(x_{ij},\widehat{x}_{ij})$$

Para calcular la **distancia**, existen diversas funciones (*funciones de coste*), como : distancia Alpha-Beta , o una de las más conocidas a nivel académico, la **distancia Euclídea**. A continuación se expondrán brevemente cada una de ellas, haciendo especial hincapié el la **distancia Euclídea**.(Cichocki et al., 2011)

4.2.2.2.1.2- DISTANCIA ALPHA-BETA

Ecuación 7

$$D_{AB}^{(\alpha,\beta)}(X,\hat{X}) = -\frac{1}{\alpha\beta}\sum_{ij}\left(x_{ij}^{\alpha}\hat{x}_{ij}^{\beta} - \frac{\alpha}{\alpha+\beta}x_{ij}^{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\hat{x}_{ij}^{\alpha+\beta}\right),$$

$$\alpha,\beta,\alpha+\beta \neq 0$$

Esta **distancia** se denomina alpha-beta debido a que usa dos parámetros, α y β . Se utilizan para conseguir el cálculo preciso de la distancia, ajustándose en función de la cantidad de ruido de los datos. Es una función, de la cual se podrían deducir las funciones que realizan el cálculo de la distancia como la que sigue en el siguiente subapartado, la **distancia Euclídea**.

4.2.2.2.1.3- DISTANCIA EUCLÍDEA

También conocida como **norma-2**.

$$D_E(X,\widehat{X}) = \frac{1}{2}\sum_{ij}(x_{ij}-\widehat{x}_{ij})^2$$

A pesar de que la técnica de factorización **NMF** trabaja con matrices de términos positivos o nulos, esta función de cálculo de distancia hubiera eliminado los posibles valores

negativos, de no haber aplicado este tipo de factorización. Esto se consigue por elevar al cuadrado, la suma de cada una de las distancias que exista entre cada elemento de la matriz original y la matriz aproximada.

Pero esto también tiene sus desventajas, ya que, para distancias grandes entre elementos, el elevar al cuadrado las distancias provocará una acentuación de estas, mientras que para caso contrario causará una gran disminución de estas.

Para la atenuación y la disminución de las distancias, es recomendable la previa normalización de los datos para obtener resultados fiables.

Debido a su extensión uso académico, esta distancia ha sido escogida para el desarrollo e implementación del sistema de recomendación expuesto en este Trabajo de Fin de Grado.

4.2.2.2.1.3- REGLAS PARA LA ACTUALIZACIÓN DE VARIABLES

En el subapartado anterior se han descrito diferentes funciones para el cálculo de la distancia entre la matriz original y la matriz aproximada. El siguiente paso en la búsqueda de la matriz aproximada se trata no más, de la descripción de un algoritmo iterativo que posee unas reglas de actualización para las variables (a_{ik}, b_{kj}^T) involucradas en las matrices de la factorización ($\mathbf{A}$, $\mathbf{B}^T$).

Dicho algoritmo iterativo buscará maximizar la similitud entre la matriz original y la matriz aproximada, es decir, **minimizar la distancia** entre ambas matrices. Para ello, se tomará como métrica, la **distancia Euclídea**.

$$D_E(X, \hat{X}) = \frac{1}{2}\sum_{ij}(x_{ij} - \hat{x}_{ij})^2$$

desglosando la anterior representación, se tiene que:

$$D_E(X, \hat{X}) = \frac{1}{2}\sum_{ij}\left(x_{ij} - \sum_{k} a_{ik}\, b_{jk}\right)^2$$

Para minimizar la función anterior, se aplicará el **descenso del gradiente** (Lee & Seung, 2001). Un algoritmo iterativo de optimización que permite converger hacia el valor mínimo de una función mediante el cálculo de la derivada parcial respecto a cada parámetro en el punto de evaluación.

Antes de entrar en la parte técnica de dicho algoritmo, se explicará el funcionamiento general del mismo. Imaginemos que nos encontramos en un terreno de montañas, colinas y valles. Nos colocamos en un punto cualquiera de este terreno. A partir de ahí, no tenemos acceso al mapa del terreno, ha caído la noche y no podemos observar a nuestro alrededor. Nuestro objetivo es alcanzar al mínimo, por lo que en esta situación lo único que podríamos hacer sería:

1. Evaluar con el pie la pendiente del sitio en el que nos encontramos

2. Avanzar una distancia en la dirección en la que la pendiente descienda con mayor intensidad.
3. Paramos en una nueva posición y **repetimos**.

Realizándolo hasta que la pendiente sea próxima a nula o nula.

Vamos a ver ahora cómo trasladamos el funcionamiento general del algoritmo del descenso del gradiente a las matemáticas. Y en concreto al caso de la factorización no-negativa de matrices.

Matemáticamente toma esta forma, como se expone a continuación:

$$\mathbf{a_{ik}} \leftarrow \mathbf{a_{ik}} - \alpha_{ik} \frac{\delta \mathbf{D}}{\delta \mathbf{a_{ik}}}$$

$$\mathbf{b_{jk}} \leftarrow \mathbf{b_{jk}} - \beta_{jk} \frac{\delta \mathbf{D}}{\delta \mathbf{b_{jk}}}$$

- $\boldsymbol{\alpha_{ik}}$ será la tasa de aprendizaje de la variable $\boldsymbol{a_{ik}}$
- $\boldsymbol{\beta_{jk}}$ será la tasa de aprendizaje de la variable $\boldsymbol{b_{jk}}$

Las tasas de aprendizaje están representadas mediante números reales positivos, cuanto mayor sean, mayor pronunciado será el avance en el descenso. Esta tasa de aprendizaje define cuánto afecta el gradiente $\left(\frac{\delta D}{\delta a_{ik}} \text{ ó } \frac{\delta D}{\delta b_{jk}}\right)$ a la actualización de los parámetros $(a_{ik} \text{ ó } b_{jk})$ respectivamente en cada iteración, o lo que es lo mismo, cuándo avanzamos en cada paso. Esto es muy importante, porque va a definir el comportamiento del algoritmo.

Si el valor de dicha tasa es muy **pequeño** los parámetros $(a_{ik} \text{ ó } b_{jk})$ variarán de tal manera que la **distancia euclídea** entre la matriz original y la aproximada irá disminuyendo hasta aproximarse a un valor de mínimo coste, aunque a costa de calcular muchas iteraciones, lo que podría repercutir en un algoritmo computacionalmente ineficiente, además de quela matriz aproximada no lograse aproximarse al valor absoluto de mínimo coste.

Por el contrario, si a la tasa de aprendizaje se le asigna un valor muy **elevado**, el cálculo de la **distancia euclídea** puede que aumente o disminuya de manera aleatoria en cada iteración, haciendo imposible para el algoritmo converger al valor de mínimo coste, causando que el proceso de optimización quede en un bucle infinito.

Por lo que el correcto ajuste del parámetro **tasa de aprendizaje**, es fundamental para que el **algoritmo del descenso del gradiente** funcione bien. Existen diferentes técnicas que ajustan este parámetro de una manera dinámica. Algunos ejemplos son: SGD, Momentum, Adagrad, Adadelta, y muchas otras más.

Volviendo a la última fórmula matemática que exponía las reglas de actualización. Y una vez resueltas las derivadas parciales, las reglas de actualización quedarían de la siguiente forma:

$$\mathbf{a_{ik}} \leftarrow \mathbf{a_{ik}} + \alpha_{ik} \sum_{\mathbf{j}} \mathbf{b_{jk}}(\mathbf{x_{ij}} - \hat{\mathbf{x}}_{\mathbf{ij}})$$

$$\mathbf{b_{jk}} \leftarrow \mathbf{b_{jk}} + \beta_{jk} \sum_{i} \mathbf{a_{ik}}(\mathbf{x_{ij}} - \hat{\mathbf{x}}_{\mathbf{ij}})$$

Desglosando los paréntesis de la anterior representación, ya que $\sum_k \mathbf{a_{ik}}\, \mathbf{b_{jk}} = \hat{\mathbf{x}}_{\mathbf{ij}}$, se tiene que:

$$\mathbf{a_{ik}} \leftarrow \mathbf{a_{ik}} + \alpha_{ik} \sum_{\mathbf{j}} \mathbf{b_{jk}}\, \mathbf{x_{ij}} \; - \alpha_{ik} \sum_{\mathbf{j}} \mathbf{b_{jk}}\, \hat{\mathbf{x}}_{\mathbf{ij}}$$

$$\mathbf{b_{jk}} \leftarrow \mathbf{b_{jk}} + \beta_{jk} \sum_{\mathbf{i}} \mathbf{a_{ik}} \mathbf{x_{ij}} \; - \beta_{jk} \sum_{\mathbf{i}} \mathbf{a_{ik}}\, \hat{\mathbf{x}}_{\mathbf{ij}}$$

Para lograr mantener la no-negatividad de los elementos de las matrices involucradas en la factorización no-negativa de matrices a la hora de aplicar las reglas de actualización, se determinan las siguientes tasas de aprendizaje:

$$\boldsymbol{\alpha_{ik}} = \frac{\mathbf{a_{ik}}}{\sum_j \mathbf{b_{jk}}\, \hat{\mathbf{x}}_{\mathbf{ij}}}$$

$$\boldsymbol{\beta_{jk}} = \frac{\mathbf{b_{jk}}}{\sum_i \mathbf{a_{ik}}\, \hat{\mathbf{x}}_{\mathbf{ij}}}$$

Ahora, las reglas de actualización junto con los valores de esas tasas de aprendizaje quedarían de la siguiente forma:

$$\mathbf{a_{ik}} \leftarrow \mathbf{a_{ik}} \frac{\sum_{\mathbf{j}} \mathbf{b_{jk}}\, \mathbf{x_{ij}}}{\sum_{\mathbf{j}} \mathbf{b_{jk}}\, \hat{\mathbf{x}}_{\mathbf{ij}}}$$

$$\mathbf{b_{jk}} \leftarrow \mathbf{b_{jk}} \frac{\sum_{\mathbf{i}} \mathbf{a_{ik}}\, \mathbf{x_{ij}}}{\sum_{\mathbf{i}} \mathbf{a_{ik}}\, \hat{\mathbf{x}}_{\mathbf{ij}}}$$

Cabe destacar la ejecución de estas dos actualizaciones deberá realizarse alternando una y otra, además de actualizar los elementos de $\widehat{\boldsymbol{X}}$ tras cada una de las ejecuciones.

En el desarrollo de este tipo de algoritmos cabe hacer uso de la llamada **regularización**, con el fin de evitar el temido sobreajuste (*overfitting*) (*Overfitting in Machine Learning*, 2017). Lo que se traduciría en poca capacidad de predicción del modelo para unos datos reales, es decir, el rendimiento del modelo en cuanto a la generalización es nefasto. La regularización es una técnica para mejorar el error predictivo del modelo. Reduciendo el sobreaprendizaje evitando que la red se ajuste demasiado al conjunto de entrenamiento.

$$\boldsymbol{D_E}(\boldsymbol{X}, \widehat{\boldsymbol{X}}) = \frac{\mathbf{1}}{\mathbf{2}} \sum_{\boldsymbol{ij}} \left(\boldsymbol{x_{ij}} - \sum_{\boldsymbol{k}} \boldsymbol{a_{ik}}\, \boldsymbol{b_{jk}} \right)^{\mathbf{2}} + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \sum_{\boldsymbol{ik}} \boldsymbol{a_{ik}^2} + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \sum_{\boldsymbol{ik}} \boldsymbol{b_{jk}^2}$$

$\boldsymbol{\lambda}$, es el llamado factor de regularización, un parámetro ajustable para conseguir un buen funcionamiento del algoritmo sin que se produzca el temido sobreajuste.

Se ha incorporado una penalización en la función de coste del cálculo de la **distancia euclídea**. La penalización estándar, que se ha aplicado a la función de coste, añade un término que incluye los elementos de las matrices A y B al cuadrado, es la llamada **regularización L2**. (Roman, 2019)

Se ha elegido la **regularización L2**, debido a que es eficiente en casos de matrices dispersas y los resultados expulsados, son matrices no dispersas, lo que se traduciría en una matriz no dispersa resultante de la multiplicación matricial de dos matrices no dispersas, y consiguiendo así las tan ansiadas predicciones de las valoraciones de los usuarios sobre los ítems en los sistemas de recomendación.

Con este nuevo término introducido, las reglas de actualización expuestas anteriormente quedarían de la siguiente forma:

$$a_{ik} \leftarrow a_{ik} \frac{\sum_j b_{jk} x_{ij}}{\sum_j b_{jk} \hat{x}_{ij} + \lambda a_{ik}}$$

$$b_{jk} \leftarrow b_{jk} \frac{\sum_i a_{ik} x_{ij}}{\sum_i a_{ik} \hat{x}_{ij} + \lambda b_{jk}}$$

4.2.3.- EVALUACIÓN DE UN SR

4.2.3.1. MEDIR EL ÉXITO DE UN SR

Depende de muchos factores. Por ejemplo, depender de los objetivos que fijados en el SR: si se habla de un SR como Amazon el éxito puede ser un incremento del volumen de ventas; en el caso de un SR como Spotify el tiempo que un usuario está visitando o escuchando canciones, o para otros de otros tipos de SR puede ser el aumento de clics, aumentar el grado de satisfacción de un usuario con las recomendaciones, etc.

En cualquier caso, para todos estos objetivos, necesitamos usuarios reales, y hay ocasiones en las que no disponemos de ellos.

En el caso de encontrarnos frente a una de estas ocasiones, se deberá seleccionar el algoritmo de recomendación óptimo para poder validar los modelos desarrollados en la investigación

4.2.3.2. TAREAS EN LA RECOMENDACIÓN

La evaluación de la recomendación dependerá de las tareas de recomendación encomendadas al SR:

- **Predicción de voto/puntuación** En esta tarea, el SR se evalúa en función del grado de coincidencia entre la predicción y los gustos reales del usuario.

- **Predicción de uso** Cómo de bueno es el SR para evaluar ítems interesantes.

- **Ránking de recomendaciones** Se medirá la calidad del orden,

- **Otras tareas** Como la confianza, la diversidad, la utilidad para el usuario, etc.

4.2.3.3. TÉCNICAS DE EXPERIMENTACIÓN

4.2.3.3.1. Hold-out

Separan el conjunto de datos, por ejemplo:

- 70% entrenamiento
- 10% validación
- 20% test

Ilustración 6 Esquema Hold-out

4.2.3.3.2. Validación cruzada

Se trata de una técnica utilizada para validar modelos de aprendizaje automático. Consiste en la división de un conjunto de datos en K bloques y se realizan K iteraciones, utilizando K-1 bloques como conjunto de entrenamiento y 1 como conjunto de test, tal como se muestra en la siguiente figura:

Se realizan varias iteraciones en las que se divide el conjunto de datos en un conjunto de bloques. Unos serán de entrenamiento y validación, y otros de Test. Y se mide la calidad del SR sobre el conjunto de test. Al final lo que se devuelve, es la media de la calidad de entre todas las iteraciones.

Ilustración 7 Validación cruzada de un dataset K=5

4.2.3.3. EVALUACIÓN DE LOS MODELOS

El esquema que se empleará para estimar la calidad de los modelos consiste en una validación cruzada de 5 particiones. Por lo tanto, el procedimiento sería el siguiente:

1- Se divide el conjunto en 5 particiones.

2- Se seleccionan 4 para entrenar y 1 para evaluar.

3- Se repite el proceso para las 4 particiones restantes

4- Se obtiene la media de los resultados obtenidos en las 5 particiones

5- Se calculan los MAE y R

5. CONJUNTO DE DATOS

El conjunto de datos *MovieLens* (Harper & Konstan, 2016) se ha obtenido de la web https://grouplens.org-/datasets/movielens/100k/.

Fueron recopilados a través del sitio web MovieLens (movielens.umn.edu) por el grupo de investigación GroupLens de la universidad de Minesota durante un período de seis meses, desde el día 19 de Septiembre de 1997 al 22 de abril de 1998.

Una descripción más detallada del conjunto de los datos es como sigue:

- **u.data**:
  - Se trata del conjunto de datos completo u,
  - 100000 clasificaciones de 943 usuarios en 1682 elementos.
  - Cada usuario ha calificado al menos 20 películas.
  - Los usuarios y artículos están enumerados de manera consecutiva.
  - Los usuarios y artículos están ordenados de manera aleatoria.
  - Se trata de una lista separada por tabulaciones de:
    - ID de usuario |
    - ID del artículo |
    - calificación |
    - marca de tiempo
  - Las marcas de tiempo son de segundos desde el 1/1/1970 UTC

- **u.info**:
  - Se trata de un conjunto de datos que describe:
    - cantidad de usuarios,
    - elementos
    - clasificaciones en el conjunto de datos u.

- **u.item**:
  - Se trata de una lista separada por tabulaciones con la información de los elementos (películas) con lo que sigue:
    - ID de la película |
    - título de la película |

- fecha de lanzamiento |
- fecha de lanzamiento del video |
- URL de IMDb |
- desconocido |
- Acción |
- Aventura |
- Animación |
- Niños'

Al trabajar con los datos anteriores, se deberá obtener de los archivos (u.item) y (u.data) aquellos datos necesarios para preparar una matriz (como la de la figura inferior), que consiste en las reacciones dadas por un conjunto de usuarios a algunos elementos de un conjunto de elementos (películas). Cada fila contendría las calificaciones otorgadas por un usuario, y cada columna contendría las calificaciones recibidas por un elemento. Una matriz con cinco usuarios y cinco elementos podría verse así:

	i_1	i_2	i_3	i_4	i_5
u_1	5		4	1	
u_2		3		3	
u_3		2	4	4	1
u_4	4	4	5		
u_5	2	4		5	2

Ilustración 8 Matriz de calificaciones

En la matriz se muestran las calificaciones realizadas por un conjunto de 5 usuarios (u1, u2, u3, u4, u5) sobre un conjunto de 5 ítems (i1, i2, i3, i4, i5) , en una escala del 1 a 5. Por ejemplo, el tercer usuario ha otorgado una valoración de 1 al ítem 5.

Como los usuarios sólo califican algunos elementos, existen algunas celdas de la matriz, que se encuentran vacías. Este tipo de matrices, que se encuentra parcialmente vacía, se llama dispersa.

6.- EXPERIMENTACIÓN

Una se ha visto un extenso marco teórico sobre los diferentes enfoques y tipos de sistemas de recomendación existentes, se va a implementar un prototipo en base al material disponible. Por lo que se ha decidido realizar la implementación de los algoritmos basados en memoria (usuario-usuario) e (item-item)(Sarwar et al., 2001), los cuales aplican técnicas estadísticas sobre el conjunto de datos para calcular las predicciones de los ratings.

Se parte de la base de que se aplicará la técnica de la validación cruzada, dividiendo el conjunto de datos 5 particiones, 4 de las cuales se utilizarán para el entrenamiento (u1.data, u2.data, ...u5.data) aplicando las técnicas para calcular las predicciones de los ratings, y 1 de ellas (u1.test, u2.test, ..., um.test) se utilizarán para testear y medir la calidad del algoritmo.

6.1.- CÓMO ENCONTRAR LOS USUARIOS SIMILARES EN FUNCIÓN DE LAS CALIFICACIONES

Los datos incluyen cinco usuarios **U1, U2, U3, U4, U5**, que han calificado dos películas. Las clasificaciones se almacenan en listas, y cada lista contiene dos números que indican la clasificación de cada película:

- Las calificaciones de **A** son [1.0, 2.0].
- Las calificaciones de B son [2.0, 4.0].
- Las calificaciones de C son [2.5, 4.0].
- Las calificaciones de D son [4.5, 5.0].

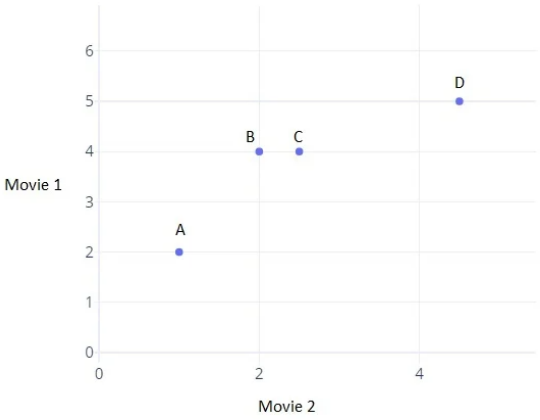

Ilustración 9 Representación sobre un plano de los vectores calificación de 4 usuarios

Si se dibujase cada uno de los vectores de calificaciones sobre un plano común, se podrían observar las distancias entre cada par de puntos.

Usar la distancia euclidiana para calcular la distancia entre las calificaciones de **A** , **B** , y **D** a la de **C** nos muestra que, en términos de distancia, las calificaciones de **C** son las más cercanos a los de **B** .

Puede ver que el usuario **C** está más cerca de **B** incluso mirando el gráfico. Pero solo de **A** y **D** , ¿a quién está más cerca **C** ?

Se podría decir que **C** está más cerca de **D** en términos de distancia. Pero mirando las clasificaciones, parecería que las elecciones de **C** se alinearían con las de **A** más que **D** porque a **A** y **C** les gusta la segunda película casi el doble de lo que les gusta la primera película, pero a **D** le gustan las dos películas Igualmente.

Entonces, ¿qué puedes usar para identificar patrones que la distancia euclidiana no pueda? ¿Se puede usar el ángulo entre las líneas que unen los puntos con el origen para tomar una decisión? Puede observar el ángulo entre las líneas que unen el origen del gráfico a los puntos respectivos como se muestra a continuación:

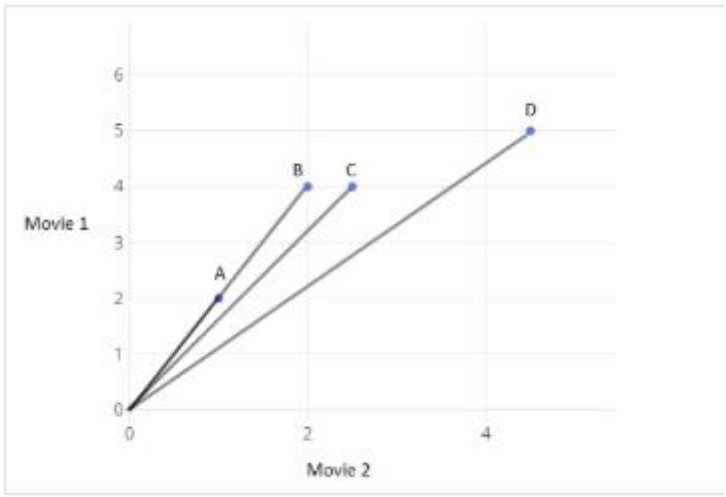

Ilustración 10 Representación de las líneas que unen cada punto al origen

El gráfico muestra cuatro líneas que unen cada punto al origen. Las líneas para **A** y **B** son coincidentes, lo que hace que el ángulo entre ellas sea cero.

Puede considerar que, si el ángulo entre las líneas aumenta, la similitud disminuye, y si el ángulo es cero, los usuarios son muy similares.

Para calcular la similitud con el ángulo, necesita una función que devuelva una **similitud mayor** o **una distancia menor** para un ángulo y una similitud menores o **una distancia mayor** para un ángulo mayor. El coseno de un ángulo es una función que disminuye de **1** a **-1** a medida que el ángulo aumenta de **0** a **180**.

Se puede usar el coseno del ángulo para encontrar la similitud entre dos usuarios. Cuanto mayor sea el ángulo, menor será el coseno y, por lo tanto, menor será la similitud de los usuarios. También puede invertir el valor del coseno del ángulo para obtener la distancia del coseno entre los usuarios restándolo de **1**.

scipy tiene una función que calcula la **distancia coseno** de los vectores. Devuelve un valor más alto para un ángulo más alto.

El ángulo inferior entre los vectores de **C** y **A** da un valor de distancia coseno menor. Si desea clasificar las similitudes de los usuarios de esta manera, use la distancia cosenoidal.

Tenga en cuenta que los usuarios **A** y **B** se consideran absolutamente similares en la métrica de similitud de coseno a pesar de tener diferentes clasificaciones. Esto es realmente una ocurrencia común en el mundo real, y los usuarios como el usuario **A** son lo que se puede llamar **calificadores duros**. Un ejemplo podría ser un crítico de cine que siempre da a cabo calificaciones inferiores a la media, pero la clasificación de los artículos en su lista sería similar a los **evaluadores Promedio** como **B**.

Para tener en cuenta tales preferencias de usuario individuales, deberá llevar a todos los usuarios al mismo nivel eliminando sus **sesgos**. Puede hacer esto restando la calificación promedio dada por ese usuario a todos los elementos de cada elemento calificado por ese usuario. Así es como se vería:

- Para el usuario **A** , el vector de calificación [1, 2]tiene el promedio 1.5, por lo que si restamos 1.5 de cada calificación le daría el vector [-0.5, 0.5].

- Para el usuario **B** , el vector de calificación [2, 4]tiene el promedio 3, por lo que restar 3 de cada calificación le daría el vector [-1, 1].

Al hacer esto, ha cambiado el valor de la calificación promedio dada por cada usuario a **0**. Intente hacer lo mismo para los usuarios **C** y **D** , y se verá que las calificaciones ahora están ajustadas para dar un promedio de **0** para todos los usuarios. Lo cual los lleva a todos al mismo nivel y elimina sus prejuicios.

El coseno del ángulo entre los vectores ajustados se llama **coseno centrado** . Este enfoque se usa normalmente cuando hay muchos valores faltantes en los vectores, y necesita colocar un valor común para completar los valores faltantes.

Llenar los valores faltantes en la matriz de calificaciones con un valor aleatorio podría dar lugar a imprecisiones. Una buena elección para llenar los valores que faltan podría ser la valoración media de cada usuario, pero los promedios originales de usuario **A** y **B** son 1.5 y 3 respectivamente, y llenando todos los valores vacíos de **A** con 1.5 y los de **B** con 3 los haría usuarios diferentes .

Pero después de ajustar los valores, el promedio **centrado** de ambos usuarios es **0**, lo que le permite capturar la idea de que el elemento está por encima o por debajo del promedio con mayor precisión para ambos usuarios con todos los valores faltantes en los vectores de ambos usuarios que tienen el mismo valor **0**.

La distancia euclidiana y la similitud del coseno son algunos de los enfoques que puede utilizar para encontrar usuarios similares entre sí e incluso elementos similares entre sí. (La función utilizada anteriormente calcula la distancia del coseno. Para calcular la similitud del coseno, reste la distancia de 1.)

6.2.- CALCULAR LAS CALIFICACIONES

En el enfoque del promedio ponderado, multiplica cada calificación por un factor de similitud (que indica qué tan similares son los usuarios). Al multiplicar con el factor de similitud, agrega pesos a las calificaciones. Cuanto más pesado sea el peso, más importante será la calificación.

El factor de similitud, que actuaría como pesos, debería ser el inverso de la distancia discutida anteriormente porque una menor distancia implica una mayor similitud. Por ejemplo, puede restar la distancia del coseno de 1 para obtener la similitud del coseno.

Con el factor de similitud **S** para cada usuario similar al usuario objetivo **U** , puede calcular el promedio ponderado utilizando esta fórmula:

$$R_U = (\sum_{u=1}^{n} R_u * S_u)/(\sum_{u=1}^{n} S_u)$$

En la fórmula anterior, cada calificación se multiplica por el factor de similitud del usuario que otorgó la calificación. La calificación final pronosticada por el usuario **U** será igual a la suma de las calificaciones ponderadas dividida por la suma de las ponderaciones.

Con un promedio ponderado, tendrá más en cuenta las calificaciones de usuarios similares en orden de similitud.

6.3. EVALUACIÓN DE LOS MODELOS

El esquema que se empleará para estimar la calidad de los modelos consiste en una validación cruzada de 5 particiones. Por lo tanto, el procedimiento sería el siguiente:

6- Se divide el conjunto en 5 particiones.

7- Se seleccionan 4 para entrenar y 1 para evaluar.

8- Se repite el proceso para las 4 particiones restantes

9- Se obtiene la media de los resultados obtenidos en las 5 particiones

10- Se calculan los MAE y R medios

6.4. RESULTADOS (USER-USER) & (ITEM-ITEM)

Al ejecutar el prototipo arrojará unos resultados, los cuales serán las matrices con las predicciones de las calificaciones mediante la ejecución de los dos algoritmos (user-user) e

(item-item) A continuación se adjunta un enlace donde se pueden visualizar los resultados de las predicciones.
(*https://github.com/ChristianReyDeLeon/TFG-RecommenderSystem/tree/mast-er/ml_e-ngine*)

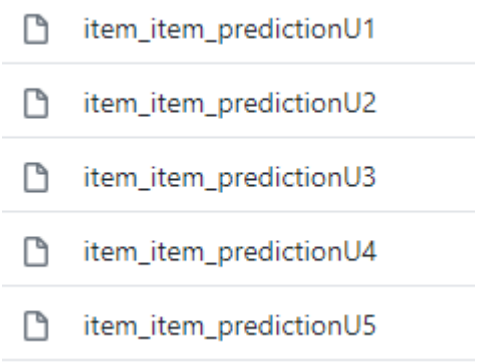

Ilustración 11 Ficheros .csv con las matrices de calificaciones predichas mediante el algoritmo (item-item)

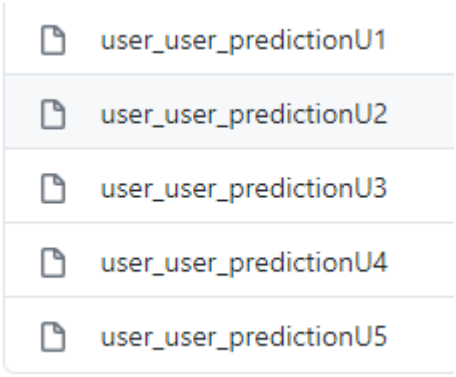

Ilustración 12 Ilustración 11 Ficheros .csv con las matrices de calificaciones predichas mediante el algoritmo (user-user)

Por otro lado, por cada experimento/Iteración de la se han obtenido los siguientes MAE y R, que determinarán la bondad de los algoritmos a la hora de realizar predicciones.

Número de experimento	**MAE (user-user)**	**R (user-user)**	**MAE(item-item)**	**R (item-item)**
Experimento 1 con u1	0.84	0.44	1.01	-0.09
Experimento 2 con u2	0.82	0.43	0.99	-0.06
Experimento 3 con u3	0.82	0.41	0.96	-0.05
Experimento 4 con u4	0.81	0.42	0.97	-0.04
Experimento 5 con u5	0.83	0.98	0.99	-0.06

Tabla 1 Resultados de los experimentos

Se obtiene la media de los 5 experimentos:

- MAE medio (user-user) = **0.824**
- R medio (user -user) = **0.424**

- MAE medio (item-item) = **0.984**
- R medio (item-item) = **-0.06**

7.- MATERIAL UTILIZADO

7.1- HARDWARE

Para la realización del trabajo, el procesado de los datos, el desarrollo del prototipo de motor de predicción de calificaciones de películas y el desarrollo de la memoria, se ha utilizado un computador portátil de las siguientes características:

- **CPU**: Intel Core i7 @2.30 GHz
- **RAM**: 8GB LPDDR3
- **Sistema operativo**: Windows 10 Education

7.2- SOFTWARE

Para los desarrollos de código se ha utilizado el lenguaje de programación *Python 3.7* junto con el entorno de desarrollo *PyCharm*. Y se ha hecho uso de las siguientes librerías:

- **NumPy**: se trata de una extensión de *Python*, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices.

- **Pandas**: Se trata de una librería de software escrita como extensión de *NumPy* para la manipulación y análisis de datos para el lenguaje de programación **Python**.

- **Sckit-learn**: se trata de una librería open-source que ofrece clases para entrenar distintos modelos de aprendizaje automático, así como funciones para el cálculo de errores y validación de modelos.

Además, todo el código, se encuentra disponible en un repositorio público en ***GitHub*** y se ha hecho uso del sistema de control de versiones de GIT, junto con su IDE GitKraken PRO.

8.- CONCLUSIONES

En el presente capítulo se realiza un análisis del trabajo llevado a cabo en la realización del "motor de recomendación" capaz de realizar predicciones de valoraciones de usuarios sobre películas.

Dicho análisis se apoya en la comparación entre el trabajo realizado y los objetivos definidos.

Por último, para cerrar este capítulo, se presentan unas líneas base de mejora del prototipo desarrollado en base a los dos algoritmos (user-user) and (item-item) de darse el caso de una continuación en su desarrollo.

8.1.- CONCLUSIONES

A continuación, se presentan las conclusiones obtenidas en cuanto al cumplimiento de los objetivos planteados al principio de este documento:

- Se ha llevado a cado un exhaustivo estudio del estado del arte de los sistemas de recomendación, así como de aquellas implementaciones que se han considerado adecuadas.

- Se han elegido los algoritmos basados en memoria (usuario-usuario) e (item-item) para la predicción de las matrices de calificaciones sobre un conjunto de usuarios y películas. En este punto, se ha podido ver, que, tras los experimentos realizados, se pueden obtener varias conclusiones:

    - En la Tabla 1, se ha podido comprobar que este tipo de algoritmos no son muy precisos a la hora de determinar una calificación a partir de conjuntos de datos muy grandes, en los que hay elementos vacíos.

    - En la Tabla 1, se ha podido comprobar, que, dentro de las capacidades de predicción de cada uno de ellos, el algoritmo (user-user) es mucho mejor algoritmo que el (item-item), ya que se ha obtenido un MAE de 0,824 frente al 0.984 del algoritmo (item-item)

- Se ha llevado a cabo la documentación de los diferentes ámbitos que el desarrollo de un prototipo de motor de recomendación con un equipo personal recoge, y a partir de los mismos, se ha desarrollado el presente documento como memoria final del Trabajo de Fin de Grado.

Con lo citado en las anteriores líneas, se da por concluido el trabajo de Fin de Grado se ha llevado a cabo con éxito cumpliendo los objetivos marcados con mi tutor de TFG.

Cabe destacar, la adquisición de nuevos conocimientos, así como dificultades que me han ido surgiendo a lo largo de la realización de este trabajo, fundamentados en parte por la inexperiencia en esta área, han logrado conseguir en mi persona, una mayor formación académica. De una manera más concreta, la mayor parte del estudio se ha centrado en la comprensión general de los motores de recomendación, y sobre todo en los que se basan en memoria y en modelo, como la Factorización matricial no negativa, que en un principio se iba a realizar recibiendo el apoyo de una empresa, pero con el tema de la Covid-19 se tenido que reformular el enfoque de este trabajo, en un tiempo récord.

Haciendo un inciso, y trasladándome a una conclusión más personal, este proyecto, sin duda alguna, ha supuesto una actividad enriquecedora, llegando a conocer la situación actual de los motores de recomendación, y nuevos campos de la estadística que en su día no hubiera tenido tan en cuenta.

8.2. LÍNEAS FUTURAS

En una posible continuación del desarrollo del prototipo utilizado para el estudio, se plantean varias de las posibles líneas a seguir:

- Implementación del modelo de motor de recomendación basado en la factorización matricial no negativa en una plataforma Cloud como **Amazon AWS**
- Comparativa con la implementación de los prototipos actuales.
- Desarrollo de la interfaz gráfica.

9.- VALORACIÓN ECONÓMICA

Se ha reservado un apartado, donde se ofrece una valoración económica evaluando el coste de la investigación realizada, para las tareas enmarcadas en la parte de desarrollo del prototipo de motor de predicciones de calificaciones de usuarios sobre los ítems que surgieron a lo largo de la investigación.

Se ha desglosado en función de las horas invertidas, el coste del software utilizado y por último el hardware.

Valoración del hardware utilizado

Hardware	Coste (€)
Ordenador portátil: ASUS N55V	**1000**

Tabla 2 Componentes Hardware

Valoración del software utilizado

Software	Coste (€)
GitHub	**0**
GitKraken PRO	**0**
PyCharm IDE	**0**
Librerías adicionales	**0**
Licencias correspondientes a la versión PRO de GitKraken	**0**

Tabla 3 Componentes software

Referencias

[1] *Arbol_decision.pdf*. (s. f.). Recuperado 12 de octubre de 2019, de http://www.dmae.upct.es/~mcruiz/Telem06/Teoria/arbol_decision.pdf

[2] *BellKor's Pragmatic Chaos*. (s. f.). Recuperado 17 de julio de 2020, de http://stats.research.att.com/volinsky/bpc.html

[3] Cichocki, A., Cruces, S., & Amari, S. (2011). Generalized Alpha-Beta Divergences and Their Application to Robust Nonnegative Matrix Factorization. *Entropy*, *13*(1), 134-170. https://doi.org/10.3390/e13010134

[4] *Factorización de matrices no negativas (NMF) de Oracle*. (2014, octubre 24). www.ibm.com/support/knowledgecenter/es/ss3ra7_sub/modeler_mainhelp_client_ddita/clementine/oracle_nmf.html

[5] Harper, F. M., & Konstan, J. A. (2016). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, *5*(4), 1-19. https://doi.org/10.1145/2827872

[6] *Interpretar los resultados clave para Correlación*. (s. f.). [Mtbconcept]. Recuperado 8 de octubre de 2019, de https://support.minitab.com/es-mx/minitab/18/help-and-how-to/statistics/basic-statistics/how-to/correlation/interpret-the-results/key-results/

[7] Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261-273. https://doi.org/10.1016/j.eij.2015.06.005

[8] *Kużelewska: Advantages of information granulation... - Google Académico*. (s. f.). Recuperado 12 de octubre de 2019, de

https://scholar.google.com/scholar_lookup?title=Advantages%20of%20information%20granulation%20in%20clustering%20algorithms&publication_year=2013&author=U.%20Ku%C5%BCelewska

[8] Lee, D. D., & Seung, H. S. (2001). Algorithms for Non-negative Matrix Factorization. En T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13* (pp. 556–562). MIT Press. http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[9] Na8. (2018, julio 10). *Clasificar con K-Nearest-Neighbor ejemplo en Python*. Aprende Machine Learning. https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/

[10] *Overfitting in Machine Learning: What It Is and How to Prevent It*. (2017, septiembre 7). EliteDataScience. https://elitedatascience.com/overfitting-in-machine-learning

[11] Page, L. (2001). *United States Patent: 6285999 - Method for node ranking in a linked database* (Patent N.º 6285999). http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=6285999.PN.&OS=PN/6285999&RS=PN/6285999

[12] Roman, V. (2019, marzo 10). *Machine Learning Supervisado: Fundamentos de la Regresión Lineal*. Medium. https://medium.com/datos-y-ciencia/machine-learning-supervisado-fundamentos-de-la-regresi%C3%B3n-lineal-bbcb07fe7fd

[13] Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the Tenth International Conference on World Wide Web - WWW '01*, 285-295. https://doi.org/10.1145/371920.372071

[14] *The history of Amazon's recommendation algorithm*. (2019, noviembre 22). Amazon Science. https://www.amazon.science/the-history-of-amazons-recommendation-algorithm

[15] *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*. (s. f.). Recuperado 6 de octubre de 2019, de https://www.computer.org/csdl/journal/tk/2005/06/k0734/13rRUxDIthy

A. ANEXO I: CÓDIGO

```
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.metrics import mean_absolute_error


'''
Prepara los arrays con los ratings reales y los ratings predichos por el modelo
'''
def getYPredTrue(y_pred_ux,y_true_ux,y_pred,y_true,ratings_pred,ratings_true):

#Serecorrenlosúnicoselementosdematriz_calificacionesdeTESTcalculandoladiferenciaentreelvalorreal
yelvalorpredicho(ERRORABSOLUTO)
    for i in range(len(ratings_true)):
        for j in range(len(ratings_true[i])):
            if ratings_true[i][j] != 0:
                y_true_ux.append(ratings_true[i][j])
                y_pred_ux.append(ratings_pred[i][j])
                y_true.append(ratings_true[i][j])
                y_pred.append(ratings_pred[i][j])
    return y_pred_ux, y_true_ux, y_pred, y_true

'''
Formato inicial de los elementos de ratings  [-2, -1, 0, 1, 2] a los valores iniciales [1,
'''
def formatoInicial(ratings_prediction):
    for i in range(len(ratings_prediction)):
        for j in range(len(ratings_prediction[i])):
            ratings_prediction[i][j] = ratings_prediction[i][j] + 3
    return ratings_prediction

def calculaPromedioValoracionesUsuario(valoraciones_train):
    # calcular la media únicamente de las películas que hayan sido calificadas
    media_valoraciones_usuario = []
    for i in range(len(valoraciones_train)):
        contador = 0
        suma = 0
        for j in range(len(valoraciones_train[i])):
            if valoraciones_train[i][j] != 0:
                suma = suma + valoraciones_train[i][j]
                contador = contador + 1
        media = suma / contador
        media_valoraciones_usuario.append(media)
    media_valoraciones_usuario = np.array(media_valoraciones_usuario)
    return media_valoraciones_usuario

def getMatriz(calificaciones):
    matriz_calificaciones = np.zeros((943, 1682))
    for fila in calificaciones.itertuples():
        matriz_calificaciones[fila[1] - 1, fila[2] - 1] = fila[3]
    return matriz_calificaciones

'''
```

```
(Usuario-Usuario)
Devuelve las predicciones de los ratings de los ususarios sobre las películas
'''
def prediceUU(ratings, similarity):
   calificacion_media_usuario = calculaPromedioValoracionesUsuario((ratings))
   # eliminamos SESGOS de los usuarios, para llevar a todos los usuarios al mismo nivel
   rattings_NoSesgados = (ratings - calificacion_media_usuario[:, np.newaxis])
   pedicciones = calificacion_media_usuario[:, np.newaxis] + similarity.dot(rattings_NoSesgados) /
np.array([np.abs(similarity).sum(axis=1)]).T
   return pedicciones

'''
(Item-Item)
Devuelve las predicciones de los ratings de los ususarios sobre las películas
'''
def prediceII(ratings, similarity):
   calificacion_media_usuario = calculaPromedioValoracionesUsuario((ratings))
   # eliminamos SESGOS de los usuarios, para llevar a todos los usuarios al mismo nivel
   rattings_NoSesgados = (ratings - calificacion_media_usuario[:, np.newaxis])
   pedicciones = calificacion_media_usuario[:, np.newaxis] + rattings_NoSesgados.dot(similarity) /
np.array([np.abs(similarity).sum(axis=1)])
   return pedicciones

# Datos de los usuarios (USUARIOS)
columnas_usuarios = ['user_id', 'age', 'sex', 'occupation', 'zip_code']
usuarios = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-100k/u.user',
sep='|', names=columnas_usuarios)


# Datos de las películas (ITEMS)
columnas_items = ['movie id', 'movie title' , 'release date', 'video release date',
            'IMDb URL', 'unknown', 'Action', 'Adventure','Animation',
            'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
            'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi',
            'Thriller', 'War', 'Western']
items = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-100k/u.item',
sep='|', names=columnas_items,
              encoding='latin-1')

# Datos de las calificaciones otorgadas por los usuarios a las películas (CALIFICACIONES)
columnas_calificaciones = ['user_id', 'movie_id', 'rating', 'unix_timestamp']
calificaciones = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-
100k/u.data', sep='\t', names=columnas_calificaciones)
'''
Partiendo de la información proporcionada por el fichero u.info:
943 users
1682 items
100000 ratings
'''
# Se obtiene la matriz con todas las CALIFICACIONES otorgadas por los usuarios a las películas
matriz_calificaciones = getMatriz(calificaciones)

# Se vuelca la matriz de CALIFICACIONES sobre un fichero .csv
# D = pd.DataFrame(data=matriz_calificaciones)
# D.to_csv('matriz_calificacionescsv', sep=';', header=False, float_format='%.2f', index=False)


# ////////////////////////////////////////////// EVALUACIÓN (Offline)
```

```
//////////////////////////////////////////////////////////////
# Este sistema de recomendación posee como tarea de recomendación, la predicción de voto / valoración de los usuarios sobre las películas que no hayan votado / valorado / calificado
# Se utiliza la técnica cross-validation para la validación del modelo. En este SR, se parte de un conjunto de datos u.data que se ha dividido en K=5 particiones para realizar K iteraciones, utilizando K-1 particiones para el conjunto de TRAIN y 1 para el conjunto TEST

######################################## u1  (Primer Experimento / Iteración)
##########################################
# TRAIN
# Datos con la información de las calificaciones_u1base (CALIFICACIONES)
columnas_calificaciones_u1base = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u1base = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-100k/u1.base', sep='\t', names=columnas_calificaciones_u1base)

# TEST
# Datos con la información de las calificaciones_u1test (CALIFICACIONES)
columnas_calificaciones_u1test = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u1test = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-100k/u1.test', sep='\t', names=columnas_calificaciones_u1test)

# Se obtiene la matriz con las CALIFICACIONES del TRAIN para la primera iteración otorgadas por los usuarios a las películas
matriz_calificaciones_u1base = getMatriz(calificaciones_u1base)

# Se obtienen las matrices con las similitudes entre ususarios y entre items respectivamente
user_similarity_u1base = pairwise_distances(matriz_calificaciones_u1base, metric='cosine') #(user-user)
item_similarity_u1base = pairwise_distances(matriz_calificaciones_u1base.T, metric='cosine') #(item-item)

# Se obtienen las predicciones de los ratings de los usuarios sobre las películas
user_prediction_u1base = prediceUU(matriz_calificaciones_u1base, user_similarity_u1base)#(user-user)
item_prediction_u1base = prediceII(matriz_calificaciones_u1base, item_similarity_u1base)#(item-item)
user_prediction_u1base = formatoInicial(user_prediction_u1base)
item_prediction_u1base = formatoInicial(item_prediction_u1base)

#Se vuelcan las matrices de CALIFICACIONES con las predicciones sobre dos ficheros en formato .csv
DU1 = pd.DataFrame(data=user_prediction_u1base)
DU1.to_csv('user_user_predictionU1', sep=';', header=False, float_format='%.2f', index=False)
DI1 = pd.DataFrame(data=item_prediction_u1base)
DI1.to_csv('item_item_predictionU1', sep=';', header=False, float_format='%.2f', index=False)

# Se obtiene la matriz con las CALIFICACIONES del TEST para la primera iteración otorgadas por los usuarios a las películas
matriz_calificaciones_u1test = getMatriz(calificaciones_u1test)

############################################## Cálculo de MAE y R2 para u1
#####################################################################
# Estos primeros arrays se volcarán sbre un fichero .csv para posteriormente mostrar las gráficas
y_usertrue = []
y_userpred = []
y_itemtrue = []
y_itempred = []

y_usertrue_u1 = []
```

```
y_userpred_u1 = []
y_itemtrue_u1 = []
y_itempred_u1 = []

y_userpred_u1, y_usertrue_u1, y_userpred, y_usertrue = getYPredTrue(y_userpred_u1, y_usertrue_u1,
y_userpred, y_usertrue, user_prediction_u1base, matriz_calificaciones_u1test)
mae_useru1 = mean_absolute_error(y_usertrue_u1, y_userpred_u1)
y_itempred_u1, y_itemtrue_u1, y_itempred, y_itemtrue = getYPredTrue(y_itempred_u1, y_itemtrue_u1,
y_itempred, y_itemtrue, item_prediction_u1base, matriz_calificaciones_u1test)
mae_itemu1 = mean_absolute_error(y_itemtrue_u1, y_itempred_u1)
print("********************* u1 *************************")
print("mae_u1 (user-user)")
print(mae_useru1)
print()
print("u1 - Coeficiente de correlación de PEARSON (user-user)")
print(np.corrcoef(y_usertrue_u1, y_userpred_u1)[0, 1])
print()
print("mae_u1 (item-item)")
print(mae_itemu1)
print()
print("u1 - Coeficiente de correlación de PEARSON (item-item)")
print(np.corrcoef(y_itemtrue_u1, y_itempred_u1)[0, 1])
print("************************************************")
print()
##########################################################################
##########################################################################
####

######################################## u2  (Segundo Experimento / Iteración)
##########################################
# TRAIN
# Datos con la información de las calificaciones_u2base (CALIFICACIONES)
columnas_calificaciones_u2base = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u2base = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-
100k/u2.base', sep='\t', names=columnas_calificaciones_u2base)

# TEST
# Datos con la información de las calificaciones_u2test (CALIFICACIONES)
columnas_calificaciones_u2test = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u2test = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-
100k/u2.test', sep='\t', names=columnas_calificaciones_u2test)

# Se obtiene la matriz con las CALIFICACIONES del TRAIN para la primera iteración otorgadas por los
usuarios a las películas
matriz_calificaciones_u2base = getMatriz(calificaciones_u2base)

# Se obtienen las matrices con las similitudes entre ususarios y entre items respectivamente
user_similarity_u2base = pairwise_distances(matriz_calificaciones_u2base, metric='cosine') #(user-
user)
item_similarity_u2base = pairwise_distances(matriz_calificaciones_u2base.T, metric='cosine') #(item-
item)

# Se obtienen las predicciones de los ratings de los usuarios sobre las películas
user_prediction_u2base = prediceUU(matriz_calificaciones_u2base, user_similarity_u2base)#(user-
user)
item_prediction_u2base = prediceII(matriz_calificaciones_u2base, item_similarity_u2base)#(item-
item)
user_prediction_u2base = formatoInicial(user_prediction_u2base)
```

```
item_prediction_u2base = formatoInicial(item_prediction_u2base)

#Se vuelcan las matrices de CALIFICACIONES con las predicciones sobre dos ficheros en formato .csv
DU2 = pd.DataFrame(data=user_prediction_u2base)
DU2.to_csv('user_user_predictionU2', sep=';', header=False, float_format='%.2f', index=False)
DI2 = pd.DataFrame(data=item_prediction_u2base)
DI2.to_csv('item_item_predictionU2', sep=';', header=False, float_format='%.2f', index=False)

# Se obtiene la matriz con las CALIFICACIONES del TEST para la primera iteración otorgadas por los
usuarios a las películas
matriz_calificaciones_u2test = getMatriz(calificaciones_u2test)

############################################### Cálculo de MAE y R2 para u2
#################################################################
y_usertrue_u2 = []
y_userpred_u2 = []
y_itemtrue_u2 = []
y_itempred_u2 = []

y_userpred_u2, y_usertrue_u2, y_userpred, y_usertrue = getYPredTrue(y_userpred_u2, y_usertrue_u2,
y_userpred, y_usertrue, user_prediction_u2base, matriz_calificaciones_u2test)
mae_useru2 = mean_absolute_error(y_usertrue_u2, y_userpred_u2)
y_itempred_u2, y_itemtrue_u2, y_itempred, y_itemtrue = getYPredTrue(y_itempred_u2, y_itemtrue_u2,
y_itempred, y_itemtrue, item_prediction_u2base, matriz_calificaciones_u2test)
mae_itemu2 = mean_absolute_error(y_itemtrue_u2, y_itempred_u2)
print("********************* u2 ************************")
print("mae_u2 (user-user)")
print(mae_useru2)
print()
print("u2 - Coeficiente de correlación de PEARSON (user-user)")
print(np.corrcoef(y_usertrue_u2, y_userpred_u2)[0, 1])
print()
print("mae_u2 (item-item)")
print(mae_itemu2)
print()
print("u2 - Coeficiente de correlación de PEARSON (item-item)")
print(np.corrcoef(y_itemtrue_u2, y_itempred_u2)[0, 1])
print("************************************************")
print()
#################################################################################
#################################################################################
####


######################################### u3  (Tercer Experimento / Iteración)
#########################################
# TRAIN
# Datos con la información de las calificaciones_u3base (CALIFICACIONES)
columnas_calificaciones_u3base = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u3base = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-
100k/u3.base', sep='\t', names=columnas_calificaciones_u3base)

# TEST
# Datos con la información de las calificaciones_u3test (CALIFICACIONES)
columnas_calificaciones_u3test = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u3test = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-
100k/u3.test', sep='\t', names=columnas_calificaciones_u3test)
```

```
# Se obtiene la matriz con las CALIFICACIONES del TRAIN para la primera iteración otorgadas por los usuarios a las películas
matriz_calificaciones_u3base = getMatriz(calificaciones_u3base)

# Se obtienen las matrices con las similitudes entre ususarios y entre items respectivamente
user_similarity_u3base = pairwise_distances(matriz_calificaciones_u3base, metric='cosine') #(user-user)
item_similarity_u3base = pairwise_distances(matriz_calificaciones_u3base.T, metric='cosine') #(item-item)

# Se obtienen las predicciones de los ratings de los usuarios sobre las películas
user_prediction_u3base = prediceUU(matriz_calificaciones_u3base, user_similarity_u3base)#(user-user)
item_prediction_u3base = prediceII(matriz_calificaciones_u3base, item_similarity_u3base)#(item-item)
user_prediction_u3base = formatoInicial(user_prediction_u3base)
item_prediction_u3base = formatoInicial(item_prediction_u3base)

#Se vuelcan las matrices de CALIFICACIONES con las predicciones sobre dos ficheros en formato .csv
DU3 = pd.DataFrame(data=user_prediction_u3base)
DU3.to_csv('user_user_predictionU3', sep=';', header=False, float_format='%.2f', index=False)
DI3 = pd.DataFrame(data=item_prediction_u3base)
DI3.to_csv('item_item_predictionU3', sep=';', header=False, float_format='%.2f', index=False)

# Se obtiene la matriz con las CALIFICACIONES del TEST para la primera iteración otorgadas por los usuarios a las películas
matriz_calificaciones_u3test = getMatriz(calificaciones_u3test)

############################################### Cálculo de MAE y R2 para u3
#####################################################################
y_usertrue_u3 = []
y_userpred_u3 = []
y_itemtrue_u3 = []
y_itempred_u3 = []

y_userpred_u3, y_usertrue_u3, y_userpred, y_usertrue = getYPredTrue(y_userpred_u3, y_usertrue_u3, y_userpred, y_usertrue, user_prediction_u3base, matriz_calificaciones_u3test)
mae_useru3 = mean_absolute_error(y_usertrue_u3, y_userpred_u3)
y_itempred_u3, y_itemtrue_u3, y_itempred, y_itemtrue = getYPredTrue(y_itempred_u3, y_itemtrue_u3, y_itempred, y_itemtrue, item_prediction_u3base, matriz_calificaciones_u3test)
mae_itemu3 = mean_absolute_error(y_itemtrue_u3, y_itempred_u3)
print("******************** u3 ************************")
print("mae_u3 (user-user)")
print(mae_useru3)
print()
print("u3 - Coeficiente de correlación de PEARSON (user-user)")
print(np.corrcoef(y_usertrue_u3, y_userpred_u3)[0, 1])
print()
print("mae_u3 (item-item)")
print(mae_itemu3)
print()
print("u3 - Coeficiente de correlación de PEARSON (item-item)")
print(np.corrcoef(y_itemtrue_u3, y_itempred_u3)[0, 1])
print("************************************************")
print()
#########################################################################
#########################################################################
########################################
```

```
######################################## u4  (Cuarto Experimento / Iteración)
##########################################
# TRAIN
# Datos con la información de las calificaciones_u4base (CALIFICACIONES)
columnas_calificaciones_u4base = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u4base = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-100k/u4.base', sep='\t', names=columnas_calificaciones_u4base)

# TEST
# Datos con la información de las calificaciones_u4test (CALIFICACIONES)
columnas_calificaciones_u4test = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u4test = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-100k/u4.test', sep='\t', names=columnas_calificaciones_u4test)

# Se obtiene la matriz con las CALIFICACIONES del TRAIN para la primera iteración otorgadas por los usuarios a las películas
matriz_calificaciones_u4base = getMatriz(calificaciones_u4base)

# Se obtienen las matrices con las similitudes entre ususarios y entre items respectivamente
user_similarity_u4base = pairwise_distances(matriz_calificaciones_u4base, metric='cosine') #(user-user)
item_similarity_u4base = pairwise_distances(matriz_calificaciones_u4base.T, metric='cosine') #(item-item)

# Se obtienen las predicciones de los ratings de los usuarios sobre las películas
user_prediction_u4base = prediceUU(matriz_calificaciones_u4base, user_similarity_u4base)#(user-user)
item_prediction_u4base = prediceII(matriz_calificaciones_u4base, item_similarity_u4base)#(item-item)
user_prediction_u4base = formatoInicial(user_prediction_u4base)
item_prediction_u4base = formatoInicial(item_prediction_u4base)

#Se vuelcan las matrices de CALIFICACIONES con las predicciones sobre dos ficheros en formato .csv
DU4 = pd.DataFrame(data=user_prediction_u4base)
DU4.to_csv('user_user_predictionU4', sep=';', header=False, float_format='%.2f', index=False)
DI4 = pd.DataFrame(data=item_prediction_u4base)
DI4.to_csv('item_item_predictionU4', sep=';', header=False, float_format='%.2f', index=False)

# Se obtiene la matriz con las CALIFICACIONES del TEST para la primera iteración otorgadas por los usuarios a las películas
matriz_calificaciones_u4test = getMatriz(calificaciones_u4test)

################################################ Cálculo de MAE y R2 para u4
#####################################################################
y_usertrue_u4 = []
y_userpred_u4 = []
y_itemtrue_u4 = []
y_itempred_u4 = []

y_userpred_u4, y_usertrue_u4, y_userpred, y_usertrue = getYPredTrue(y_userpred_u4, y_usertrue_u4, y_userpred, y_usertrue, user_prediction_u4base, matriz_calificaciones_u4test)
mae_useru4 = mean_absolute_error(y_usertrue_u4, y_userpred_u4)
y_itempred_u4, y_itemtrue_u4, y_itempred, y_itemtrue = getYPredTrue(y_itempred_u4, y_itemtrue_u4, y_itempred, y_itemtrue, item_prediction_u4base, matriz_calificaciones_u4test)
mae_itemu4 = mean_absolute_error(y_itemtrue_u4, y_itempred_u4)
print("********************* u4 ************************")
print("mae_u4 (user-user)")
```

```
print(mae_useru4)
print()
print("u4 - Coeficiente de correlación de PEARSON (user-user)")
print(np.corrcoef(y_usertrue_u4, y_userpred_u4)[0, 1])
print()
print("mae_u4 (item-item)")
print(mae_itemu4)
print()
print("u4 - Coeficiente de correlación de PEARSON (item-item)")
print(np.corrcoef(y_itemtrue_u4, y_itempred_u4)[0, 1])
print("************************************************")
print()
##########################################################################
##########################################################################
########################################

######################################## u5  (Quinto Experimento / Iteración)
##########################################
# TRAIN
# Datos con la información de las calificaciones_u5base (CALIFICACIONES)
columnas_calificaciones_u5base = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u5base = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-100k/u5.base', sep='\t', names=columnas_calificaciones_u5base)

# TEST
# Datos con la información de las calificaciones_u5test (CALIFICACIONES)
columnas_calificaciones_u5test = ['user_id2', 'movie_id2', 'rating2', 'unix_timestamp2']
calificaciones_u5test = pd.read_csv('file://localhost/E:/TFG/TFG-RecommenderSystem/data/ml-100k/u5.test', sep='\t', names=columnas_calificaciones_u5test)

# Se obtiene la matriz con las CALIFICACIONES del TRAIN para la primera iteración otorgadas por los usuarios a las películas
matriz_calificaciones_u5base = getMatriz(calificaciones_u5base)

# Se obtienen las matrices con las similitudes entre ususarios y entre items respectivamente
user_similarity_u5base = pairwise_distances(matriz_calificaciones_u5base, metric='cosine') #(user-user)
item_similarity_u5base = pairwise_distances(matriz_calificaciones_u5base.T, metric='cosine') #(item-item)

# Se obtienen las predicciones de los ratings de los usuarios sobre las películas
user_prediction_u5base = prediceUU(matriz_calificaciones_u5base, user_similarity_u5base)#(user-user)
item_prediction_u5base = prediceII(matriz_calificaciones_u5base, item_similarity_u5base)#(item-item)
user_prediction_u5base = formatoInicial(user_prediction_u5base)
item_prediction_u5base = formatoInicial(item_prediction_u5base)

#Se vuelcan las matrices de CALIFICACIONES con las predicciones sobre dos ficheros en formato .csv
DU5 = pd.DataFrame(data=user_prediction_u5base)
DU5.to_csv('user_user_predictionU5', sep=';', header=False, float_format='%.2f', index=False)
DI5 = pd.DataFrame(data=item_prediction_u5base)
DI5.to_csv('item_item_predictionU5', sep=';', header=False, float_format='%.2f', index=False)

# Se obtiene la matriz con las CALIFICACIONES del TEST para la primera iteración otorgadas por los usuarios a las películas
matriz_calificaciones_u5test = getMatriz(calificaciones_u5test)
```

```
############################################### Cálculo de MAE y R2 para u5
#####################################################################
y_usertrue_u5 = []
y_userpred_u5 = []
y_itemtrue_u5 = []
y_itempred_u5 = []

y_userpred_u5, y_usertrue_u5, y_userpred, y_usertrue = getYPredTrue(y_userpred_u5, y_usertrue_u5,
y_userpred, y_usertrue, user_prediction_u5base, matriz_calificaciones_u5test)
mae_useru5 = mean_absolute_error(y_usertrue_u5, y_userpred_u5)
y_itempred_u5, y_itemtrue_u5, y_itempred, y_itemtrue = getYPredTrue(y_itempred_u5, y_itemtrue_u5,
y_itempred, y_itemtrue, item_prediction_u5base, matriz_calificaciones_u5test)
mae_itemu5 = mean_absolute_error(y_itemtrue_u5, y_itempred_u5)
print("********************* u5 ************************")
print("mae_u5 (user-user)")
print(mae_useru5)
print()
print("u5 - Coeficiente de correlación de PEARSON (user-user)")
print(np.corrcoef(y_usertrue_u5, y_userpred_u5)[0, 1])
print()
print("mae_u5 (item-item)")
print(mae_itemu5)
print()
print("u5 - Coeficiente de correlación de PEARSON (item-item)")
print(np.corrcoef(y_itemtrue_u5, y_itempred_u5)[0, 1])
print("************************************************")
print()
#####################################################################################
#####################################################################################
########################################

# Media de los MAE de los 5 experimentos
# Se calcula la media de los EAM de los 5 Experimentos / Iteraciones
mae_usermedio = mean_absolute_error(y_usertrue, y_userpred)
mae_itemmedio = mean_absolute_error(y_itemtrue, y_itempred)
print("************************** MAE medio de los 5 Experimentos realizados
************************")
print("mae_medio (user-user)")
print(mae_usermedio)
print()
print("Coeficiente de correlación de PEARSON (user-user)")
print(np.corrcoef(y_usertrue, y_userpred)[0, 1])
print()
print("mae_medio (item-item)")
print(mae_itemmedio)
print()
print("Coeficiente de correlación de PEARSON (item-item)")
print(np.corrcoef(y_itemtrue, y_itempred)[0, 1])
print("*************************************************************************************************"
)
print()

df_user = pd.DataFrame({'y_usertrue': y_usertrue, 'y_userpred': y_userpred},
            columns=['y_usertrue', 'y_userpred'])
df_item = pd.DataFrame({'y_itemtrue': y_itemtrue, 'y_itempred': y_itempred},
            columns=['y_itemtrue', 'y_itempred'])

# se vuelcan los resultados de las predicciones obtenidos mediante los dos algoritmos (user-user) and
```

```
(item_item)
# Se vuelcan sobre dos ficheros .csv
LUS = pd.DataFrame(data=df_user)
LUS.to_csv('df_user.csv', sep=';', header=False, float_format='%.2f', index=False)
LIS = pd.DataFrame(data=df_item)
LIS.to_csv('df_item.csv', sep=';', header=False, float_format='%.2f', index=False)

##########################################################################
#################################################
```