Sistemas Recomendadores

Pablo Díaz

Mayo 2025

Link collaboratory

Índice

1. Introducción

En este proyecto, implementaremos diferentes sistemas recomendadores mediante diversos enfoques. El objetivo general es crear y comparar difernetes sistemas capacez de emitir recomendaciones de manera personalizada utilizando valoraciones reales de usuarios sobre peliculas.
Para ello, trabajremos con el dataset MovieLens 100k, un dataset de uso académico muy utilizado y reconocido.
Llevaremos a cabo diferentes tipos de filtrado:
Filtrado colaborativo: Basado en usuarios así como basado en items, utilizando para ello la biblioteca Surprise y comprobando su desempeño mediante métricas como RMSE y MAE.
Recomendaciones por contenido: Aprovechando la información de los géneros de las películas construiremos sistema capaz de recomendar en funcion a similitudes con los mismos.
También construiremos un modelo simple de embedding basado en Deep Learning

Por último, implementaremos un sistema híbrido que combinará tanto las valoraciones históricas de los usuarios como las características propias de las películas como el género y los metadatos de los usuarios edad y ocupación. Para ello, emplearemos un enfoque basado en Deep Learning, capaz de integrar toda esta información de forma conjunta y aprender representaciones latentes más complejas, con el objetivo de mejorar la calidad y personalización de las recomendaciones.

Además del uso habitual de métricas de error como RMSE o MAE, incluiremos también evaluación específica basada en métricas Top-K como Precisión@10, Recall@10 y Hitrate@10 con el fin de poder comparar de formas más objetiva los diferentes modelos.

2. Datos

Como acabamos de comentar, trabajaremos con el conjunto de datos MovieLens 100k. Este dataset contiene exactamente **100.000 valoraciones** con notas entre 1 y 5, realizadas por **943 usuarios** sobre un total de **1.682 películas**. El conjunto de datos está distribuido en varios ficheros, cada uno con una función específica:

1. u.data contiene las valoraciones, incluye identificador de usuario, identificador de la película, puntuación asignada y una marca temporal.
2. u.item ofrece información sobre las películas: título, año de estreno y codificacion binaria de los generos a los que pertenece cada una.
3. u.user incluye metadatos de los usuarios, como edad, sexo, ocupación y código postal

En total, cada usuario ha valorado al menos 20 películas, lo que garantiza una cierta densidad en la matriz de valoraciones y permite aplicar con éxito técnicas de filtrado colaborativo. Por otro lado, las películas están etiquetadas con hasta 19 géneros distintos, lo que proporciona una base rica para los enfoques basados en contenido.
Para comprender mejor la naturaleza de las valoraciones, generamos histograma que muestra su distribución:

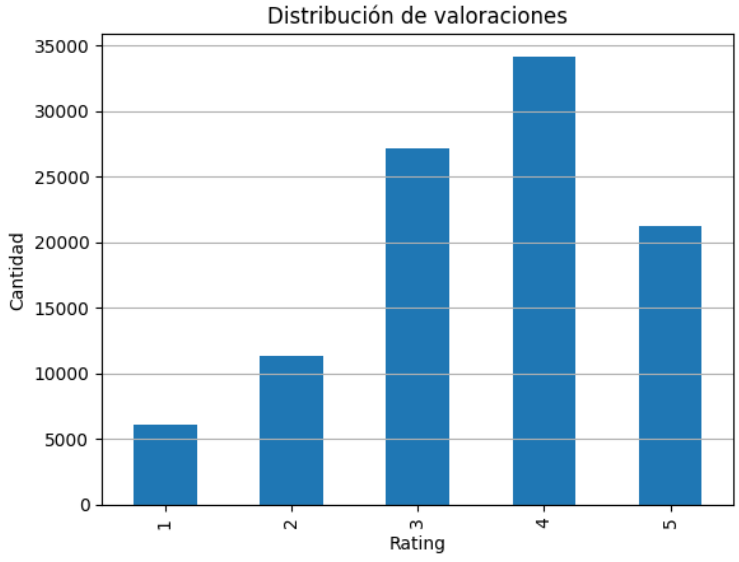

Figura 1: Distribución de valoraciones.

Como se observa en la figura, la mayoría de los usuarios tienden a valorar de manera mas o menos positiva las películas; las puntuaciones de 4 y 3 estrellas son más frecuentes que las valoraciones bajas (1 y 2 estrellas). Mientras que máxima valoración (5 estrellas) es también bastante abundante. Esta asimetría en la distribución sugiere una posible tendencia a valorar solo aquellas películas que se han disfrutado o una predisposición general hacia evaluaciones favorables.

2.1. Preprocesado

Antes de avanzar en la construcción de los sistemas de recomendación, realizamos un preprocesado inicial del conjunto de datos. Los diferentes archivos los procesaremos y uniremos para tenerlos todos juntos. Comenzamos con el archivo `u.data` con el objetivo de verificar su calidad y asegurar su consistencia.
Comprobamos si existen valores nulos en las variables principales (user_id, item_id, rating y timestamp), y confirmamos que no había registros incompletos. Además, verificamos que todas las valoraciones se encontraran dentro del rango esperado, entre 1 y 5 puntos, sin detectar puntuaciones anómalas.

En cuanto a la duplicidad de datos, comprobamos que no existen filas repetidas de forma exacta, ni tampoco valoraciones múltiples por parte de un mismo usuario sobre una misma película.

Luego, incorporamos la información complementaria del archivo `u.item`, que contiene metadatos asociados a cada película. Tras cargar este fichero, procedimos a normalizar los títulos con el objetivo de detectar posibles duplicidades. Para ello, convertimos los títulos a minúsculas, eliminamos caracteres especiales y espacios innecesarios. Este tratamiento permitió identificar varios casos en los que una misma película aparecía con títulos idénticos tras la normalización, pero con distintos identificadores (item_id). En total, se localizaron 18 títulos duplicados que presentaban más de un identificador único. Para resolver esta ambigüedad, seleccionamos, para cada título repetido, la versión con mayor número de valoraciones, asumiendo que esta representa la versión más consolidada o más utilizada por los usuarios. El duplicado correspondiente a cada una fue eliminado.
Posteriormente, realizamos una serie de comprobaciones adicionales sobre los metadatos de las películas con el fin de identificar posibles carencias o inconsistencias. En primer lugar, confirmamos que todas las películas contenían título, salvo una cuyo nombre era simplemente 'unknown'. Esta misma película tampoco contaba con fecha de estreno o género. Por otro lado, contabilizamos 135 películas que únicamente recibieron una valoración, lo cual podría tener un impacto en los sistemas de recomendación. En estos casos decidimos dejarlos como estaban sin eliminarlos de nuestro sistema

Como parte final del preprocesado, incorporamos los metadatos asociados a los usuarios a partir del archivo `u.user`, que incluye información demográfica como la edad, el género, la ocupación y el código postal.
Realizamos una serie de comprobaciones básicas para detectar posibles registros anómalos o inconsistentes. Observamos que el conjunto contiene exactamente dos géneros ('M' y 'F') y un total de 21 ocupaciones distintas, entre las que se encuentran categorías como student, programmer o retired, así como un valor 'none', asociado a perfiles sin ocupación definida. En cuanto a la variable de edad, no se detectaron valores fuera de un rango razonable (entre 5 y 100 años), lo que sugiere que no existen registros erróneos o atípicos en este campo. Sin embargo, identificamos dos usuarios con códigos postales inválidos ('00000'), lo que podría indicar valores por defecto o errores en la captura. Estos casos puntuales fueron registrados, aunque no afectan de forma significativa al análisis posterior.

Con esto tendríamos todo el preprocesado completo y estamos listos para comenzar con los sistemas recomendadores.

3. Sistemas Recomendadores

Comenzamos ahora con los diferentes sistemas recomendadores. Dividimos los datos en un conjunto para entrenamiento y otro para test, lo que nos permitirá evaluar todos los modelos de la forma más justa posible. Esta división se ha llevado a cabo utilizando una función (*split_no_cold*) diseñada específicamente para evitar el problema del cold-start, tanto de usuarios como de ítems. El cold-start (arranque frío en español) hace referencia a situaciones en las que el sistema debe generar recomendaciones sobre usuarios o películas de los que no dispone de información previa, lo que impide realizar predicciones fiables.
En nuestro caso, seleccionamos exactamente diez valoraciones aleatorias por usuario para incluirlas en el conjunto de test. El resto de sus valoraciones permanecen en el conjunto de entramiento, garantizando así que todos los usuarios estén presentes en ambos conjuntos.
Además, nos aseguramos de que todas las películas presentes en el conjunto de test también hayan aparecido en el conjunto de entrenamiento. Esto es fundamental, ya que el sistema no puede generar predicciones sobre ítems que nunca ha visto. Para evitar este problema, en los casos en los que una película solo contaba con una única valoración —y esta caía en el test—, dicha valoración se trasladó al conjunto de entrenamiento.

Así garantizamos que el modelo tenga al menos una referencia previa sobre cada película evaluada. En nuestro caso, el conjunto de entrenamiento resultante contiene 90.189 registros, mientras que el conjunto de test incluye exactamente 9.424 valoraciones. Esta estrategia garantiza que los modelos puedan entrenarse y evaluarse de forma robusta, evitando errores comunes como la ausencia de referencias previas sobre un usuario o película en test.
Pasemos ahora a filtrado colaborativo.

3.1. Filtrado Colaborativo

Los sistemas de recomendación basados en filtrado colaborativo son un tipo de modelo que genera sugerencias analizando el historial de valoraciones realizadas por los usuarios. Este enfoque no se basa en características de los productos, sino en los patrones de comportamiento compartido entre usuarios. Es especialmente útil cuando no contamos con mucha información sobre los items, pero si se cuenta con un historial amplio de valoraciones.
Implementaremos dos tipos de filtrado colaborativo: Basado en usuarios y basado en items. Para ambos enfoques utilizaremos algoritmo `KNNBaseline`. Este algoritmo emplea técnicas de vecinos más cercanos (KNN) pero introduce una corrección que tiene en cuenta los sesgos individuales de cada usuario y cada película. En un primer modelo utilizamos un KNN básico pero debido al bajo rendimiento optamos por cambiar a KNNBaseline.

3.1.1. Basados en usuario

Este tipo de filtrado colaborativo se basa en la idea de que si dos usuarios han valorado de forma similar ciertas películas, podemos decir que ambos usuarios son similares. Y en el caso de que uno de ellos haya visto y valorado positivamente una película que el otro todavía no ha visto, es razonable suponer que también podría gustarle. A partir de este principio, el sistema genera recomendaciones para un usuario basándose en las valoraciones de sus vecinos más cercanos, es decir, aquellos con los que comparte un patrón de gustos parecido.

Para construir el modelo, utilizamos la librería `Surprise`, pensada específicamente para desarrollar sistemas de recomendación basados en valoraciones explícitas. Esta herramienta resulta especialmente útil, ya que permite preparar fácilmente los datos en el formato requerido, aplicar distintos algoritmos colaborativos y evaluar su rendimiento mediante métricas como RMSE o MAE. Además, incluye utilidades que simplifican la validación cruzada y la búsqueda automatizada de hiperparámetros.

Una vez cargados los datos en el formato requerido por `Surprise`, preparamos el proceso de entrenamiento aplicando validación cruzada para seleccionar la combinación de parámetros más adecuada. Utilizamos una validación de 5 particiones (5-fold cross-validation) y evaluamos distintos valores del número de vecinos (`k`), junto con dos tipos de medida de similitud: `coseno` y `pearson`. Para llevar a cabo esta búsqueda, empleamos la clase `GridSearchCV`, que permite comparar automáticamente todas las combinaciones posibles de parámetros, midiendo el rendimiento mediante métricas como el RMSE y el MAE. De esta forma, el modelo se ajusta no solo para minimizar el error de predicción, sino también para ofrecer resultados estables y generalizables.

Tras completar la búsqueda, el modelo que obtuvo mejores resultados fue el que utilizó `k = 35` vecinos y la métrica de similitud `pearson`, con una puntuación media de `0.933` en RMSE y `0.733` en MAE. Esta configuración ofreció el mejor equilibrio entre precisión y estabilidad en las predicciones.
Una vez seleccionados estos parámetros óptimos, entrenamos el modelo final sobre el conjunto de datos de entrenamiento, dejándolo listo para generar predicciones y recomendaciones personalizadas. El modelo final tiene un RMSE de 0.958

Con el modelo entrenado, evaluamos su capacidad de predicción comparando las valoraciones reales con las estimadas por el sistema. Para ello, generamos predicciones sobre el conjunto de test —es decir, sobre valoraciones que el modelo no ha visto durante el entrenamiento— y analizamos cómo se distribuyen las puntuaciones predichas en función de la valoración real.

En la figura 2 se muestra un diagrama de cajas que resume este comportamiento, permitiendo visualizar si el sistema tiende a sobrestimar o subestimar las valoraciones reales en función de cada nivel de puntuación.

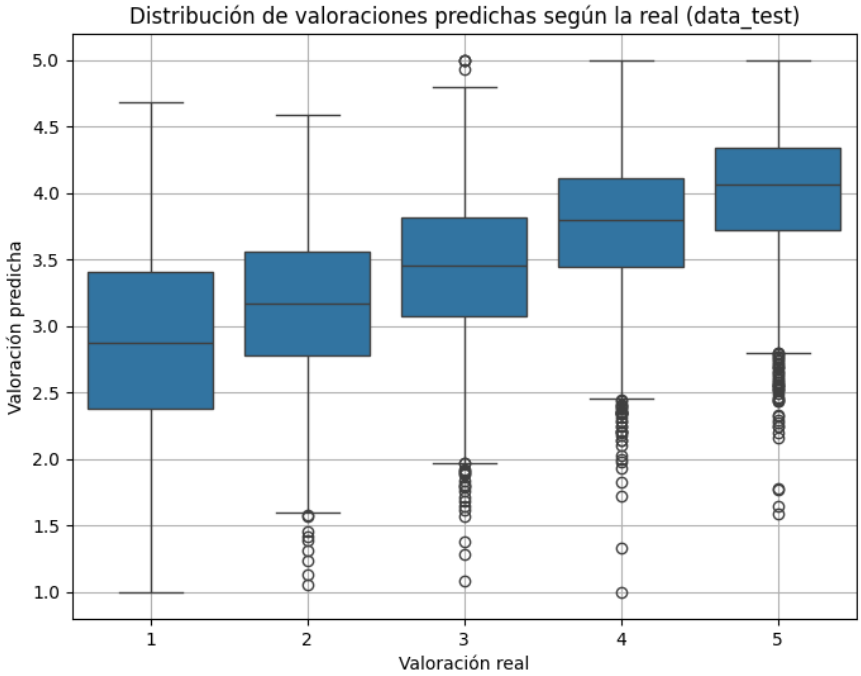

Figura 2: Valoracion real vs Predicción.

Lo que observamos es una clara tendencia del modelo a suavizar las valoraciones extremas. En los casos en los que la valoración real fue baja (1 o 2), el modelo tiende a sobreestimar, asignando puntuaciones más altas de las que realmente se dieron. Por el contrario, en los casos con valoraciones altas (4 o 5), tiende a subestimar, prediciendo valores algo por debajo de los reales. Esta tendencia a concentrar las predicciones en un rango medio es una limitación común en los modelos basados en vecinos, y debe tenerse en cuenta al interpretar las recomendaciones generadas.
A nivel cuantitativo, el resumen estadístico del error absoluto refuerza estas observaciones. El modelo presenta un error medio de aproximadamente 0.75, con una desviación estándar cercana a 0.59. Aunque la mayoría de los errores se concentran entre 0.3 y 1.0, se observan casos extremos con errores de hasta 3.68, lo que confirma la existencia de desviaciones importantes en determinados ítems

Además del análisis general, realizamos un estudio más detallado para identificar qué películas fueron mejor y peor estimadas por el modelo. Para ello, agrupamos las predicciones por película y calculamos, en cada caso, la media de las valoraciones reales y la media de las predichas. A partir de estas medias, obtuvimos la diferencia absoluta, lo que nos permitió ordenar las películas según su grado de acierto.

Cuadro 1: Top 5 películas mejor predichas (modelo User-Based)

Título	**Real**	**Predicha**	**Diferencia**	**Nº val**
Crows and Sparrows (1949)	1.000	1.000	0.000	1
Richie Rich (1994)	1.000	1.000	0.000	20
Amityville: A New Generation (1993)	1.000	1.000	0.000	4
Major Payne (1994)	1.000	1.000	0.000	18
Faust (1994)	5.000	5.000	0.000	4

Cuadro 2: Top 5 películas peor predichas (modelo User-Based)

Título	**Real**	**Predicha**	**Diferencia**	**Nºval**
Rough Magic (1995)	1.000	4.484	3.484	1
8 Heads in a Duffel Bag (1997)	1.000	4.173	3.173	3
Steel (1997)	5.000	1.942	3.058	17
New York Cop (1996)	4.000	1.000	3.000	1
Sliding Doors (1998)	5.000	2.294	2.706	3

Al observar las películas mejor predichas, vemos que algunas de ellas tienen un número de valoraciones relativamente bajo en el conjunto de entrenamiento. En concreto, hay casos con una sola valoración, lo que puede explicar por qué la predicción del modelo coincide exactamente con el valor real: simplemente no hay

suficiente información para ajustar una estimación más general. Sin embargo, esto no implica necesariamente que el modelo haya aprendido bien, sino que ha replicado el único dato disponible. En el caso de las películas peor predichas ocurre algo parecido: muchas de ellas también cuentan con un historial de valoraciones limitado. Esta escasez de datos hace que el modelo tenga dificultades para generar predicciones precisas, lo que se traduce en errores altos.
Por tanto, tanto los aciertos perfectos como los fallos más significativos tienden a aparecer en títulos con pocos votos previos. Esto sugiere que el volumen de valoraciones en el entrenamiento influye notablemente en la fiabilidad de la predicción

Como paso complementario a la evaluación cuantitativa general, analizamos ahora el funcionamiento práctico del sistema desde el punto de vista del usuario final. A partir del modelo entrenado, generamos recomendaciones personalizadas para un usuario concreto. Para ello, aplicamos un enfoque en dos variantes: una versión general que calcula la predicción para todas las películas no vistas, y una versión filtrada que introduce un criterio adicional basado en la actividad de los vecinos más cercanos. Durante las pruebas iniciales, observamos que sin aplicar ningún filtro el sistema tendía a recomendar títulos poco conocidos, pero con valoraciones muy altas por parte de uno o dos usuarios. Esto daba lugar a situaciones en las que se sugerían películas con una sola valoración de cinco estrellas, lo cual no resultaba fiable. Para evitar este tipo de recomendaciones sesgadas, introdujimos la condición adicional: considerar únicamente aquellas películas que hubieran sido valoradas por al menos cinco vecinos del usuario. De esta forma, se mejora la robustez de las sugerencias, priorizando títulos con un respaldo más amplio dentro del grupo de usuarios similares.

El funcionamiento del recomendador es sencillo: dado el identificador de un usuario, el sistema identifica qué películas no ha visto aún y genera una predicción de valoración para cada una de ellas. A continuación, ordena las películas según la puntuación estimada y devuelve las mejores recomendaciones. En la versión filtrada, además de la predicción del modelo, se incluye información adicional sobre cuántos vecinos del usuario han valorado cada película y cuál fue la media de esas valoraciones. Esto permite priorizar títulos que no solo tienen una alta puntuación estimada, sino que además cuentan con un respaldo significativo dentro del grupo de usuarios similares, lo que mejora la fiabilidad de las sugerencias. Como ejemplo, ponemos al usuario 123 que usaremos en todos los modelos.

Top 10 recomendaciones para el usuario 123 (según vecinos similares):

titulo	media_vecinos	prediccion_modelo	num_vecinos
Titanic (1997)	4.45	4.40	11
Good Will Hunting (1997)	4.22	4.32	9
L.A. Confidential (1997)	4.29	4.23	7
Fugitive, The (1993)	4.20	4.06	5
Apt Pupil (1998)	4.33	4.05	6
Full Monty, The (1997)	3.89	3.79	9
Boogie Nights (1997)	4.14	3.77	7
Contact (1997)	3.88	3.75	16
Ulee's Gold (1997)	4.00	3.72	6
Toy Story (1995)	4.17	3.71	6

Figura 3: Recomendaciones para usuario 123 mediante User-Based y filtro.

Las recomendaciones generadas para el usuario 123 muestran títulos populares y bien valorados por varios de sus vecinos más cercanos, lo que sugiere que el sistema está captando correctamente patrones de afinidad. Además, las puntuaciones predichas se mantienen próximas a la media de valoraciones de los vecinos, lo que refuerza la coherencia del modelo.

3.1.2. Basados en items

Continuamos con otro tipo de filtrado colaborativo, el basado en items. Se basa en la idea de que si un usuario ha valorado positivamente un determinado item, lo más probable es que también le gusten otros items que hayan recibido valoraciones similares por parte del conjunto de usuarios. En lugar de comparar usuarios entre sí, como ocurre en el filtrado colaborativo basado en usuarios, este enfoque analiza la similitud entre ítems a partir de los patrones de valoración que han recibido por parte del conjunto de usuarios.

Utilizaremos el mismo enfoque que en basado en usuarios. Buscamos los mejores parámetros como valor de k o la métrica de similitud mediante gridsearch con cross-validation en 5 particiones. De nuevo en este caso, el mejor modelo en términos de error cuadrático medio (RMSE) y error absoluto medio (MAE) fue el que utilizó k = 35 y la métrica de similitud pearson, que nos brindó un RMSE=0.944 y un MAE=0.744

Al igual que en caso basado en usuarios, una vez entrenado el modelo final con la configuración óptima cuyo RMSE final es de 0.949, procedemos a evaluar su comportamiento comparando las valoraciones reales con las predichas. Para ello, generamos predicciones sobre el conjunto de test y analizamos la distribución de los valores estimados frente a los reales.
En la figura se muestra un diagrama de cajas que representa la distribución de las valoraciones predichas en función de las valoraciones reales:

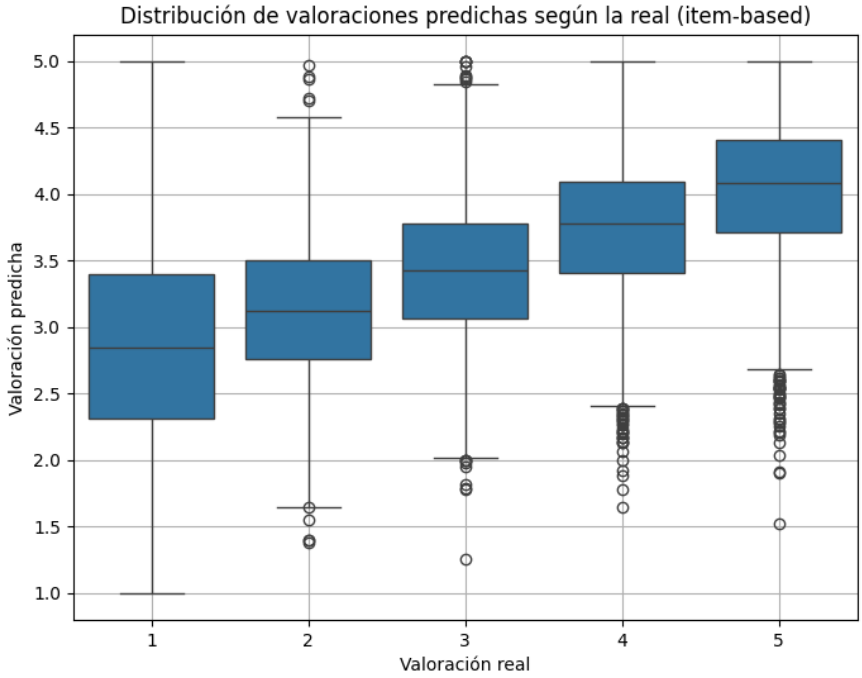

Figura 4: Distribución de valoraciones predichas según la real (item-based).

Como se observa en la gráfica, el modelo presenta una tendencia similar al caso basado en usuarios: tiende a suavizar los extremos. Las valoraciones reales bajas (1 o 2) suelen ser sobreestimadas, mientras que las altas (4 o 5) tienden a estar ligeramente subestimadas.
Esta observación también se confirma a través del análisis estadístico del error absoluto por predicción. El resumen de errores muestra un valor medio de aproximadamente 0.74, con una desviación estándar cercana a 0.59. Aunque la mayoría de los errores se concentran entre 0.3 y 1.0, se observa un error máximo de 4.0, lo que indica que en algunos casos las predicciones se alejan considerablemente del valor real.

Al igual que en modelo basado en usuarios, realizamos un estudio más detallado para evaluar qué películas fueron mejor y peor predichas por el modelo. Para ello, agrupamos las predicciones por ítem y calculamos, para cada película, la media de las valoraciones reales y la media de las predichas. A partir de estas medias, obtuvimos la diferencia absoluta, lo que nos permitió identificar los títulos con mayor y menor desviación.

Cuadro 3: Top 5 películas mejor predichas (modelo item-based)

Título	**Real**	**Predicha**	**Diferencia**	**Nº val.**
This Is Spinal Tap (1984)	3.929	3.929	0.000	177
Saint, The (1997)	3.093	3.092	0.000	262
Emma (1996)	3.731	3.731	0.000	151
Streetcar Named Desire, A (1951)	4.000	3.999	0.001	89
Dark City (1998)	3.400	3.401	0.001	11

Cuadro 4: Top 5 películas peor predichas (modelo item-based)

Título	**Real**	**Predicha**	**Diferencia**	**Nº val.**
Wes Craven's New Nightmare (1994)	5.000	2.279	2.721	34
Gabbeh (1996)	5.000	2.352	2.648	5
Rough Magic (1995)	1.000	3.537	2.537	1
8 Heads in a Duffel Bag (1997)	1.000	3.534	2.534	3
Inventing the Abbotts (1997)	1.000	3.504	2.504	22

Entre las películas mejor estimadas por el modelo item-based se encuentran títulos como This Is Spinal Tap, Emma o The Saint, todas ellas con un número moderado o alto de valoraciones. En estos casos, la predicción se ajusta con gran precisión a la media real, lo que sugiere que el modelo logra afinar mejor cuando dispone de una base suficiente de valoraciones sobre cada ítem.

En cambio, las películas peor estimadas son en su mayoría títulos con pocas valoraciones, como Rough Magic o Gabbeh . En estos casos, el sistema no cuenta con suficiente información para calcular similitudes fiables entre ítems, lo que conduce a errores importantes. A diferencia del modelo basado en usuarios —donde incluso las películas con una sola valoración pueden resultar bien predichas por coincidencia—, aquí sí observamos que disponer de más valoraciones contribuye de forma clara a mejorar la calidad de las estimaciones.

Por último, utilizamos el modelo entrenado para generar recomendaciones personalizadas para un usuario concreto, usamos de nuevo al usuario 123. Para ello, como en caso user-based, analizamos las películas que el usuario aún no ha visto y estimamos su posible valoración basándose en ítems similares que sí ha valorado previamente.
El criterio seguido para aceptar una predicción fue que el usuario hubiese valorado al menos tres películas similares a la que se quiere recomendar. Esta condición garantiza un mínimo de información útil para poder emitir una recomendación fiable. Para cada recomendación, además de la puntuación estimada, se calculó la media de las valoraciones del usuario sobre los ítems vecinos, lo que permite tener una referencia adicional sobre la coherencia del resultado.

Top 10 recomendaciones para el usuario 123 (item-based, ordenado por predicción):

titulo	media_vecinos	prediccion_modelo	num_peliculas_similares
Men With Guns (1997)	5.00	4.74	3
Last Summer in the Hamptons (1995)	5.00	4.44	3
When Night Is Falling (1995)	4.67	4.40	3
Joy Luck Club, The (1993)	4.67	4.39	6
Mina Tannenbaum (1994)	4.67	4.33	3
Of Love and Shadows (1994)	5.00	4.26	3
Last Klezmer: Leopold Kozlowski, His Life and Music, The (1995)	4.67	4.23	3
Day the Sun Turned Cold, The (Tianguo niezi) (1994)	4.67	4.20	3
My Life and Times With Antonin Artaud (En compagnie d'Antonin Artaud) (1993)	4.75	4.17	4
Rendezvous in Paris (Rendez-vous de Paris, Les) (1995)	4.50	4.14	4

Figura 5: Recomendaciones para usuario 123 mediante Item-Based y filtro.

En general, las recomendaciones obtenidas presentan predicciones elevadas, todas por encima de los 4 puntos, y están respaldadas por entre 3 y 6 películas similares previamente valoradas por el usuario. Esto indica que el modelo logra identificar ítems potencialmente relevantes a partir de las interacciones previas. Además, los títulos sugeridos no corresponden a películas especialmente populares, lo que sugiere que el sistema mantiene un equilibrio razonable entre precisión y diversidad sin limitarse únicamente a ítems de alta visibilidad.

3.2. Sistema Basado en Contenido

Es el turno ahora de construir un recomendador basado en contenido. Este tipo de recomendadores aprovechan la información de las propias películas, en nuestro caso los géneros, para sugerir otras similares. Para ello, combinamos los géneros activos de cada película en una única cadena de texto. Por ejemplo, si una película está etiquetada como “Action”, “Sci-Fi” y “Horror”, generamos el string “Action Sci-Fi Horror”. Esta transformación nos permite aplicar técnicas de análisis textual como TF-IDF, que convierten esas etiquetas en vectores numéricos según su frecuencia relativa, permitiendo calcular similitudes entre películas en función de sus géneros..

Para calcular cuan similares son dos películas, usamos la medida de similitud del coseno. Se trata de una métrica que compara dos vectores según el ángulo que forman entre ellos: cuanto más pequeño es el ángulo, más parecidos son. En nuestro caso, si dos películas comparten géneros similares, sus vectores estarán más alineados y la similitud será mayor.

Una vez construida la matriz de similitud, diseñamos dos estrategias diferentes de recomendación. La primera consiste en dado el título de una película, obtenemos películas similares siempre y cuando tengan un número mínimo de 10 votos para garantizar resultados de cierta calidad. Mostramos tanto el título de las películas similares, asi como los géneros y el número de votos, pudiendo así entender por qué se ha sugerido cada recomendación.

En esta primera estrategia se utiliza el conjunto completo de datos, sin separar en entrenamiento y test, ya que el objetivo es explorar qué tipo de relaciones temáticas es capaz de capturar el modelo. Por ejemplo, para la película *Alien (1979)* nos recomienda películas como *Alien 3(1979), Body Snatchers (1993), Alien:*

Resurrection(1997), The Terminator (1984) entre otras. Como usuario, estas recomendaciones resultan coherentes y relevantes desde un punto de vista temático.

Cuadro 5: Películas similares a *Alien (1979)* según el sistema basado en contenido

Título	**Año**	**Similitud**	**Géneros**
Alien 3	1992	1.000	Action, Horror, Sci-Fi, Thriller
Body Snatchers	1993	0.929	Horror, Sci-Fi, Thriller
Alien: Resurrection	1997	0.928	Action, Horror, Sci-Fi
Hellraiser: Bloodline	1996	0.928	Action, Horror, Sci-Fi
Deep Rising	1998	0.928	Action, Horror, Sci-Fi
The Terminator	1984	0.866	Action, Sci-Fi, Thriller
Terminator 2: Judgment Day	1991	0.866	Action, Sci-Fi, Thriller
Face/Off	1997	0.866	Action, Sci-Fi, Thriller
The Arrival	1996	0.866	Action, Sci-Fi, Thriller
The Lawnmower Man	1992	0.866	Action, Sci-Fi, Thriller

La siguiente estrategia de recomendación se trata, como en otros casos, de generar recomendaciones personalizadas para un usuario en concreto. En este caso, partimos de las películas que dicho usuario ha valorado positivamente (>= 4) en el conjunto de entrenamiento y buscamos otras que aún no ha visto que guarden alta similitud con ellas. Esto nos permite ofrecer sugerencias que se alineen con sus gustos personales. De nuevo, filtramos y nos quedamos con películas que al menos tengan 10 votos para que sean algo más conocidas y relevantes esas recomendaciones.

Para cada película candidata es decir, no vista aún por el usuario, calculamos su similitud acumulada con respecto a todas las películas que el usuario ha valorado positivamente en el conjunto de entrenamiento. Esta similitud acumulada se obtiene sumando los valores de similitud coseno entre la película candidata y cada una de las favoritas. Cuanto mayor sea esta suma, mayor es la afinidad global entre la candidata y el conjunto de preferencias del usuario. Este valor no tiene un significado absoluto, dependerá del número de películas favoritas y del grado de similitud entre ellas, pero resulta muy útil para ordenar las candidatas de forma coherente.

Las recomendaciones en este caso para nuestro usuario 123 fueron las siguientes:

Cuadro 6: Películas recomendadas al usuario 123 según el sistema basado en contenido

Título	**Año**	**Similitud**	**Votos**	**Géneros**
The American President	1995	10.705	151	Comedy, Drama, Romance
Cinema Paradiso	1988	10.705	112	Comedy, Drama, Romance
Manhattan	1979	10.705	86	Comedy, Drama, Romance
Don Juan DeMarco	1995	10.705	71	Comedy, Drama, Romance
Wings of Desire	1987	10.705	51	Comedy, Drama, Romance
Corrina, Corrina	1994	10.705	38	Comedy, Drama, Romance
Brassed Off	1996	10.705	30	Comedy, Drama, Romance
Twelfth Night	1996	10.705	28	Comedy, Drama, Romance
Something to Talk About	1995	10.705	21	Comedy, Drama, Romance
Jerry Maguire	1996	10.566	330	Drama, Romance

Desde un punto de vista cualitativo, las recomendaciones obtenidas para el usuario 123 resultan coherentes y temáticamente homogéneas. Las películas recomendadas comparten un núcleo común de géneros: comedia, drama y romance. Esto confirma que el sistema es capaz de capturar patrones de afinidad temática con precisión a partir de los géneros codificados, incluso sin información explícita de valoraciones previas para esas películas en concreto.

Sin embargo, esta fortaleza también conlleva una limitación: el sistema tiende a recomendar películas que, aunque similares en etiquetas, pueden resultar redundantes o poco diversas. A diferencia de los sistemas colaborativos, el sistema basado en contenido no intenta predecir qué puntuación daría el usuario a una película determinada. Su objetivo es, más bien, identificar títulos no vistos que sean temáticamente afines a los que el usuario ha valorado positivamente, utilizando como criterio de ordenación la similitud acumulada. Esto permite generar recomendaciones interpretables y coherentes con el perfil de gustos del usuario, sin necesidad de estimar una nota exacta.

3.3. Sistema Basado en Deep Learning con Embeddings

El siguiente sistema recomendador que implementamos fue una red neuronal con embeddings básica, cuya idea principal es aprender a predecir que puntuación le daría un usuario a una película. Para ello, utilizamos el módulo Keras de TensorFlow.
En el primer paso, transformamos los identificadores originales de usuarios y películas en índices numéricos consecutivos, ya que las capas de embedding requieren como entrada enteros ordenados. A continuación, definimos dos capas de embedding independientes: una para los usuarios y otra para los ítems. Estas capas generan vectores numéricos latentes que representan las características ocultas de cada entidad. A diferencia del modelo híbrido que implementaremos más adelante, aquí no utilizamos información explícita como los géneros de las películas o la ocupación del usuario. El modelo se entrena únicamente a partir de los identificadores de usuario y película, dejando que aprenda automáticamente las relaciones latentes entre ambos a través de las valoraciones.

Una vez obtenidos los embeddings, los aplanamos y concatenamos en un único vector que captura la interacción usuario-película. Este vector pasa por una capa oculta con activación ReLU, seguida de una capa de salida lineal que produce la predicción final del rating.
Durante el entrenamiento, el modelo ajusta los pesos internos de la red para minimizar el error entre las valoraciones reales y las predichas. Una vez finalizado este proceso, generamos predicciones sobre el conjunto de test y evaluamos los resultados utilizando métricas como el RMSE que en este modelo es de RMSE=0.976.
Al igual que en los casos anteriores, representamos la distribución de errores mediante boxplot, lo que nos permite comparar visualmente las valoraciones reales y las estimadas por el modelo.

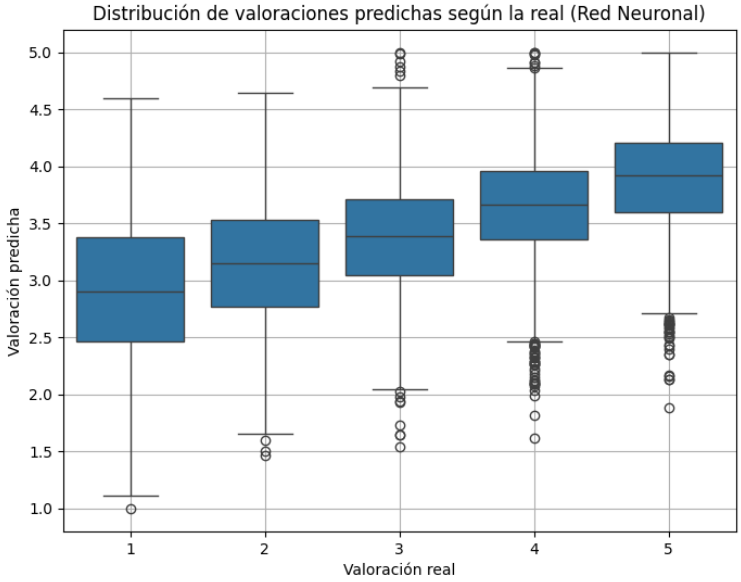

Figura 6: Distribución de valoraciones predichas según la real (Red Neuronal Simple

El análisis estadístico del error absoluto nos brinda que la media del error es de aproximadamente 0.78, con una desviación estándar de 0.59. La mayoría de las predicciones presentan un error entre 0.31 y 1.11, aunque se alcanzan diferencias máximas de hasta 3.59 puntos. Estos resultados muestran un comportamiento similar al de los sistemas colaborativos básicos logrando capturar la tendencia general de que a mayor valoración real, también mayor la predicción. Además, observamos mismo fenómeno de suavizado de valores extremos.

Generamos la tabla con películas mejor y peor predichas. Analizando la calidad de las predicciones por ítem, el modelo híbrido identifica varias películas cuya diferencia entre la valoración media real y la predicha es prácticamente nula, con errores inferiores a 0.005 puntos. Es el caso de Brazil (1985), Flubber (1997) o Mary Reilly (1996). Algunas de ellas presentan un número alto-moderado de valoraciones, mientras que otras tienen un número más limitado (13, 15). Esto sugiere que, si bien el modelo puede realizar buenas estimaciones incluso con pocos datos, estas coincidencias deben interpretarse con cautela y no asumirse como evidencia de un rendimiento generalizado.
En el extremo opuesto, las películas peor predichas coinciden en tener generalmente pocas valoraciones (1, 5). Estos errores, ya observados en modelos anteriores, reflejan los límites de generalización del sistema cuando no hay suficiente información.

Cuadro 7: Top 5 películas mejor predichas (modelo Red Neuronal con embeddings)

Título	**Real**	**Predicha**	**Diferencia**	**Nº val.**
Brazil (1985)	3.789	3.789	0.001	189
Lost in Space (1998)	3.333	3.334	0.001	15
Air Up There, The (1994)	2.667	2.665	0.002	13
Mary Reilly (1996)	2.750	2.752	0.002	35
Flubber (1997)	2.583	2.585	0.002	41

Cuadro 8: Top 5 películas peor predichas (modelo Red Neuronal con embeddings)

Título	**Real**	**Predicha**	**Diferencia**	**Nº val.**
Rough Magic (1995)	1.000	3.651	2.651	1
Wes Craven's New Nightmare (1994)	5.000	2.506	2.494	34
Inventing the Abbotts (1997)	1.000	3.453	2.453	22
Gabbeh (1996)	5.000	2.551	2.449	5
NeverEnding Story III, The (1994)	1.000	3.426	2.426	11

Finalmente, generamos recomendaciones personalizadas para un usuario concreto. El procedimiento es el mismo: identificamos las películas no vistas, obtenemos las predicciones del modelo y ordenamos los resultados por puntuación estimada. A modo informativo, incluimos también el número de valoraciones recibidas por cada ítem, lo que ayuda a valorar el grado de respaldo que tiene cada sugerencia.

Cuadro 9: Películas recomendadas al usuario 123 según red neuronal básica

Título	**Predicción**	**Nº valoraciones**
Pather Panchali (1955)	4.33	8
Close Shave, A (1995)	4.31	97
Wallace & Gromit: The Best of Aardman Animations (1996)	4.29	57
Usual Suspects, The (1995)	4.20	240
Hard Eight (1996)	4.15	15
Paths of Glory (1957)	4.15	30
To Live (Huozhe) (1994)	4.13	11
Secrets & Lies (1996)	4.12	134
Good Will Hunting (1997)	4.09	164
Chinatown (1974)	4.08	140

En el top 10 de recomendaciones, destacan títulos como The Usual Suspects (predicho: 4.20, 240 valoraciones), Good Will Hunting (4.09, 164 valoraciones) o Chinatown (4.87, 140 valoraciones), todas ellas ampliamente valoradas por la comunidad, lo que aporta mayor respaldo a la sugerencia. También aparecen otras películas menos conocidas pero con una predicción elevada, como Pather Panchali (4.33, 8 valoraciones) o Hard Eight (4.15, 15 valoraciones), que podrían representar descubrimientos potenciales interesantes. El modelo, por tanto, no solo prioriza ítems populares, sino que también es capaz de recomendar obras menos vistas pero altamente valoradas por quienes las conocen, lo que enriquece la experiencia del usuario combinando seguridad y exploración.

3.4. Autoencoder

Un autoencoder es un tipo de red neuronal diseñada para aprender una representación comprimida de los datos y luego reconstruir los datos originales desde esa representacion. Tiene dos partes: Codificdor (encoder) que comprime los datos de entrada en un espacio de menor dimension (captura los patrones más relevantes de los datos). Decodificador (decoder): intenta reconstruir la entrada original a partir de esa representacion comprimida.

En nuestro caso, implementamos un autoencoder colaborativo, es decir, un modelo que aprende exclusivamente a partir de las valoraciones históricas de los usuarios, sin incorporar información adicional como los géneros de las películas o los metadatos de los usuarios. Este enfoque se basa en la idea de que los patrones de co-valoración entre usuarios y películas contienen suficiente información como para predecir puntuaciones no observadas. El modelo se entrena utilizando como entrada cada fila de la matriz usuario $\times$ ítem, que representa las valoraciones realizadas por un usuario, y trata de reconstruir esa misma fila en la salida. De este modo, aprende representaciones internas que capturan relaciones latentes entre ítems y usuarios, permitiendo

estimar valoraciones faltantes. Dado que la mayoría de las entradas en la matriz usuario × ítem son ceros —no porque la película haya sido mal valorada, sino porque directamente no ha sido valorada—, resulta fundamental utilizar una función de pérdida que tenga esto en cuenta. Para ello, implementamos una función de pérdida enmascarada (masked loss) que ignora los ceros durante el entrenamiento. De este modo, el modelo únicamente intenta ajustar las valoraciones que realmente existen, sin verse afectado por la ausencia de datos.

La arquitectura del autoencoder consiste en una red simétrica con capas densas: la capa de entrada recibe la vectorización completa del usuario y se reduce progresivamente mediante capas intermedias hasta alcanzar un cuello de botella (representación latente). A continuación, se reconstruye la entrada original a través de capas espejo. En nuestro caso, utilizamos una arquitectura con una capa oculta de 64 neuronas, activación ReLU y salida lineal. Para evitar sobreajuste, aplicamos Dropout tras la capa codificadora. Como función de pérdida empleamos el error cuadrático medio (MSE) y como optimizador, Adam.
El RMSE obtenido en el conjunto de test fue de 1.199, notablemente más alto que el de otros modelos probados como el filtrado colaborativo o la red neuronal simple. Este resultado sugiere que, en este caso, el autoencoder no ha logrado generalizar con la misma eficacia a las valoraciones reales no vistas durante el entrenamiento. A pesar de ello, generamos también recomendaciones personalizadas para el usuario 123 a partir de la matriz reconstruida.

En el top 10 de películas sugeridas, el modelo ofrece títulos reconocidos como The Great Escape (124 valoraciones), Dr. Strangelove (194), Vertigo (179) o The Fugitive (336), todas ellas con una alta puntuación predicha (5.0) y una sólida base de valoraciones previas, lo que aporta confianza al sistema.

Cuadro 10: Películas recomendadas al usuario 123 según el sistema autoencoder

Título	**Predicción**	**Nº valoraciones**
Great Escape, The (1963)	5.00	124
Duck Soup (1933)	5.00	93
Forbidden Planet (1956)	5.00	67
1-900 (1994)	5.00	5
Eat Drink Man Woman (1994)	5.00	80
Convent, The (Convento, O) (1995)	5.00	2
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)	5.00	194
Philadelphia Story, The (1940)	5.00	104
Vertigo (1958)	5.00	179
Fugitive, The (1993)	5.00	336

En conjunto, el autoencoder demuestra ser una herramienta potente para reconstruir patrones globales, pero también sensible a la calidad y cantidad de datos disponibles, especialmente en contextos de esparsidad elevada como el nuestro.

3.5. Sistema Híbrido

Como último sistema recomendador, construimos una red neuronal híbrida donde aprovechamos tanto las valoraciones históricas como información adicional sobre los usuarios y las películas. En este caso incorporamos variables explícitas como la edad del usuario, su ocupación y los géneros de las películas. El objetivo es enriquecer el modelo con más contexto para que pueda realizar recomendaciones más ajustadas, especialmente en casos donde hay poca información disponible sobre un usuario o un ítem concreto.
Para ello, combinamos distintas fuentes de datos: por un lado, transformamos los identificadores de usuarios y películas en vectores de embeddings, como en la red neuronal simple; y por otro, codificamos la edad como una variable numérica normalizada, la ocupación como una variable categórica con codificación entera, y los géneros de las películas como un vector binario one-hot. Todos estos elementos se concatenan en un único vector que representa la interacción enriquecida entre usuario-película. Esta entrada se alimenta a una red densa con capas intermedias, activaciones ReLU y una capa de salida lineal, que predice la puntuación esperada. El modelo se entrena minimizando el error cuadrático medio (MSE) mediante el optimizador Adam.

Tras el entrenamiento, el modelo alcanza un RMSE en el conjunto de test de 0.981, lo que lo sitúa en un punto intermedio entre los modelos colaborativos y las redes neuronales previas. Si bien no logra superar a la red neuronal simple en este caso concreto, demuestra un rendimiento competitivo y especialmente útil cuando se desea incorporar información adicional para personalizar las recomendaciones.

Para evaluar más a fondo el comportamiento del modelo, representamos la distribución de valoraciones predichas agrupadas por la puntuación real como puede verse en el gráfico correspondiente

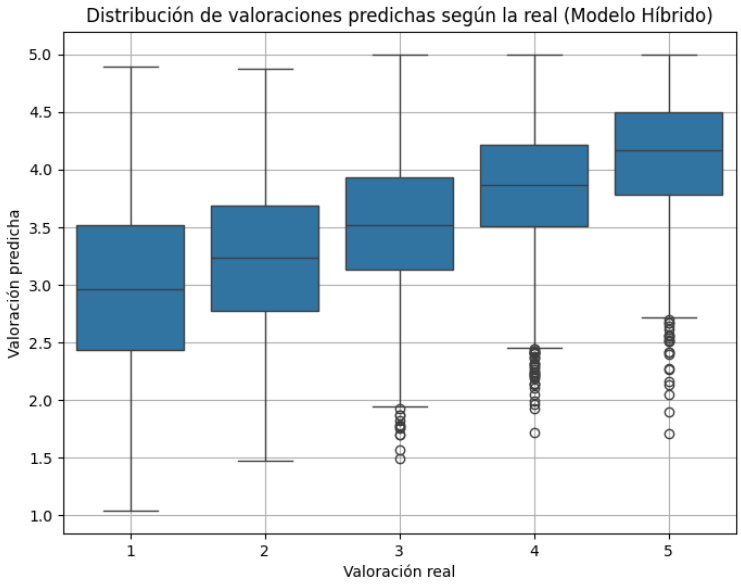

Figura 7: Distribución de valoraciones predichas según la real (Red Neuronal Simple

Como en el resto de modelos, es capaz de capturar correctamente la tendencia global: las predicciones aumentan de forma progresiva a medida que lo hace la valoración real. Destacar que el modelo híbrido, aunque también suaviza, lo hace de forma más controlada que otros modelos, y eso puede reflejar un mejor equilibrio entre generalización y precisión.
Además del análisis visual, evaluamos el error absoluto entre las valoraciones reales y las predichas. El resumen estadístico muestra una media de error en torno a 0.78, con una desviación estándar de aproximadamente 0.59. La mayoría de los errores se concentran entre 0.31 (percentil 25) y 1.11 (percentil 75), mientras que el valor mínimo es 0.0 y el máximo alcanza los 3.59 puntos, valores comparables a los observados en los otros modelos.

En cuanto a películas mejor y peor predichas:

Cuadro 11: Top 5 películas mejor predichas (modelo híbrido)

Título	**Real**	**Predicha**	**Diferencia**	**Nº val.**
My Best Friend's Wedding (1997)	3.500	3.499	0.001	150
Streetcar Named Desire, A (1951)	4.000	3.998	0.002	89
Wrong Trousers, The (1993)	4.400	4.402	0.002	108
Bullets Over Broadway (1994)	4.200	4.198	0.002	81
Four Rooms (1995)	3.000	3.003	0.003	81

Cuadro 12: Top 5 películas peor predichas (modelo híbrido)

Título	**Real**	**Predicha**	**Diferencia**	**Nº val.**
Rough Magic (1995)	1.000	4.428	3.428	1
Wes Craven's New Nightmare (1994)	5.000	2.278	2.722	34
Inventing the Abbotts (1997)	1.000	3.709	2.709	22
NeverEnding Story III, The (1994)	1.000	3.665	2.665	11
Gridlock'd (1997)	2.000	4.422	2.422	18

Al igual que en los modelos anteriores, observamos una tónica común: las películas con un mayor número de valoraciones suelen presentar una mejor concordancia entre la media real y la predicha, mientras que aquellas con pocos votos tienden a concentrar los errores más altos. Aunque el modelo híbrido logra minimizar en gran medida las desviaciones extremas, sigue siendo sensible a la falta de datos en ciertos ítems.

Finalmente, el modelo híbrido genera un conjunto de 10 recomendaciones personalizadas para el usuario 123, en las que destacan títulos como Wallace & Gromit, Pather Panchali, The Usual Suspects o Paths of Glory, todas con puntuaciones estimadas por encima de 4.4. Algunas de estas películas ya aparecían

en otros modelos, lo que sugiere una coherencia en el perfil general del usuario; sin embargo, también se incluyen títulos distintos como Safe o Thin Man, que no figuraban entre las recomendaciones anteriores. Este fenómeno pone de relieve cómo cada modelo según los datos que utiliza construye una representación distinta del usuario, y por tanto prioriza recomendaciones diferentes.

Cuadro 13: Películas recomendadas al usuario 123 según el sistema híbrido

Título	**Predicción**	**Nº valoraciones**
Wallace & Gromit: The Best of Aardman Animations (1996)	4.709	57
Pather Panchali (1955)	4.693	8
Close Shave, A (1995)	4.688	97
Usual Suspects, The (1995)	4.654	240
Hard Eight (1996)	4.638	15
Third Man, The (1949)	4.526	63
Paths of Glory (1957)	4.487	30
To Live (Huozhe) (1994)	4.471	11
Thin Man, The (1934)	4.467	49
Safe (1995)	4.409	13

4. Evaluación y Comparación de Resultados

Aparte de la métrica RMSE o MAE hay diferentes estrategias para evaluar la calidad de nuestros sistemas. Eso sí, no existe una única métrica universalmente válida que determine qué sistema recomendador es "mejor" en términos absolutos; la evaluación depende del contexto de uso, el objetivo del sistema y el tipo de interacción esperada entre usuario y contenido.
Entre las diferentes estrategias, se encuentran las ya aplicadas hasta ahora como calculo de RMSE o MAE para los modelos capaces de hacer predicción de valoración.

Hay otro tipo de evaluación basada en ranking o Top-N. Este tipo de enfoque se basa en recomendar al usuario un conjunto reducido de items por ejemplo 10 películas y encontrar si entre esas recomendaciones se encuentran efectivamente las que más le gustaron. Para ello se emplean métricas como Precision@k que mide cuantos de los K ítems recomendados son realmente relevantes. El Recall@k que indica la proporción de relevantes que han sido recomendados y el Hit-rate@k que evalúa si al menos un ítem relevante aparece entre las recomendaciones.

Existe una tercera vía de evaluación centrada en la experiencia del usuario final. Este enfoque busca medir no solo si el sistema acierta en sus recomendaciones, sino también si las personas realmente las encuentran útiles, relevantes o satisfactorias. En contextos reales como plataformas de streaming, tiendas online o redes sociales se utilizan indicadores como la tasa de clics, el tiempo de visualización o permanencia, la frecuencia de interacción, o incluso la tasa de conversión (porcentaje de recomendaciones que acaban en una compra, visualización o acción concreta). También es habitual realizar tests A/B, donde se comparan dos versiones distintas del sistema (por ejemplo, dos algoritmos de recomendación) y se observa cuál genera más interacción o mejores resultados.

En nuestro caso, para complementar RMSE implementamos evaluación basada en top-N. Concretamente, desarrollamos funciones específicas para calcular métricas como Precision@10, Recall@10 y HitRate@10, adaptadas a la lógica de cada tipo de modelo. Estas métricas se calculan de forma agregada sobre todos los usuarios del conjunto de test, evaluando si los ítems realmente relevantes (aquellos que el usuario valoró con una puntuación alta) aparecen entre las 10 principales recomendaciones generadas por el sistema. Este enfoque nos permite evaluar de forma equitativa todos los modelos desarrollados, aplicando las mismas métricas y condiciones de test para todos ellos. Gracias a estas medidas cuantitativas, es posible asignar a cada modelo un valor numérico que resume su rendimiento, lo cual resulta muy útil a la hora de comparar alternativas de forma objetiva. Aunque estas métricas no capturan todos los matices del comportamiento de un recomendador, sí nos ofrecen una referencia clara y matemática para entender qué sistema tiende a funcionar mejor en nuestro caso.

Cuadro 14: Comparativa de modelos de recomendación

Modelo	RMSE	Precision@10	Recall@10	Hit Rate@10	Tiempo (s)
User-Based	0.9582	0.0018	0.0022	0.0159	1.25
Item-Based	0.9496	0.0263	0.0452	0.2227	1.80
Content-Based	–	0.0106	0.0178	0.0903	–
Red Neuronal	0.9759	0.0179	0.0313	0.1569	42.10
Autoencoder	1.1992	0.0056	0.0095	0.0562	5.76
Modelo Híbrido	0.9807	0.0113	0.0203	0.1007	50.63

A la vista de los resultados obtenidos, el modelo que ha mostrado un comportamiento más equilibrado ha sido el item-based. Presenta un RMSE moderadamente bajo (0.9496), pero sobre todo destaca en las métricas top-N: logra la mayor precisión (0.0263), recall (0.0452) y tasa de aciertos (0.2227), lo que indica que sus recomendaciones resultan útiles con mayor frecuencia para los usuarios.
Por otro lado, el autoencoder ha sido el modelo con menor rendimiento general. Aunque tiene una estructura más compleja, su RMSE fue el más alto (1.1992), y las métricas top-N también fueron las más bajas. Esto sugiere que, en contextos con alta dispersión de datos y pocas valoraciones por usuario, esta arquitectura no logra generalizar adecuadamente sin un preprocesado más específico o sin un mayor volumen de datos.
Un caso especialmente interesante es el de la red neuronal simple con embeddings, que ha obtenido mejores resultados que el modelo híbrido, a pesar de contar con menos información como entrada. En este caso, el aumento de complejidad en el modelo híbrido no se traduce necesariamente en una mejora del rendimiento.
En cuanto al modelo basado en contenido, aunque no destaca en ninguna métrica, ofrece recomendaciones razonablemente consistentes y rápidas, sin requerir interacciones entre usuarios. Esto lo convierte en una opción especialmente útil en escenarios donde los datos sobre los usuarios son escasos o no están disponibles.

Cabe destacar que las métricas top-N obtenidas, como Precision@10, Recall@10 y HitRate@10, presentan en general valores bastante bajos, especialmente en los modelos más complejos. Sin embargo, consideramos que estos resultados no se deben a errores en la implementación de los sistemas, sino más bien al hecho de que los modelos no fueron específicamente optimizados para maximizar estas métricas sino para minimizar RMSE. Además, la naturaleza del dataset y las condiciones estrictas de evaluación —como el uso de test independientes y umbrales elevados para definir relevancia— también contribuyen a esta limitación.

5. Conclusión

A lo largo de este proyecto nos hemos adentrado en el mundo de los sistemas de recomendación, explorando distintas formas de abordar el problema de sugerir contenidos a los usuarios. Hemos probado enfoques clásicos como el filtrado colaborativo, métodos basados en contenido, y también modelos más avanzados como redes neuronales, autoencoders y una versión híbrida que combina distintos tipos de información.
El proceso ha sido muy enriquecedor, no solo por lo técnico, sino por todo lo que implica trabajar con datos reales como los del conjunto MovieLens 100k: preparar y limpiar los datos, dividir correctamente en entrenamiento y test, y adaptar cada modelo a las particularidades del problema. Hemos aprendido a evaluar el rendimiento no solo con métricas globales como el RMSE, sino también con métricas top-N como Precision@10, Recall@10 y Hit Rate@10, que nos han ayudado a entender qué modelos son más útiles en la práctica.
Además, a lo largo del trabajo hemos visualizado errores, analizado casos concretos y generado recomendaciones reales para usuarios, tratando de ir siempre un paso más allá de simplemente entrenar y medir. El campo de los sistemas recomendadores es muy amplio y admite múltiples estrategias; nosotros hemos seguido una aproximación inicial, combinando métodos clásicos y redes neuronales, con resultados razonablemente buenos para ser una primera implementación.

De cara a futuras versiones del sistema, podrían explorarse diferentes líneas de mejora. Por un lado, sería interesante probar modelos alternativos como SVD (Singular Value Decomposition) o SVD++, ampliamente utilizados en producción y capaces de capturar interacciones complejas entre usuarios e ítems. También se podrían probar arquitecturas de red más profundas o modelos secuenciales como RNNs o Transformers, especialmente útiles si se dispone de información temporal. Otra vía de mejora sería enriquecer el conjunto de datos incorporando información adicional como directores, actores, duración, productoras o valoraciones externas. Esta ampliación del contenido permitiría construir perfiles más ricos y mejorar la calidad de las recomendaciones personalizadas. Por último, podrían aplicarse estrategias de evaluación más sofisticadas, como tests A/B, métricas basadas en diversidad o novedad, o incluso simulaciones de comportamiento real del usuario. Estas herramientas permitirían valorar no solo la precisión del sistema, sino también su impacto práctico en la experiencia del usuario.

6. Referencias

- Shani, G., & Gunawardana, A. (2011). *Evaluating Recommendation Systems*. En F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender Systems Handbook*. Springer. doi: 10.1007/978-0-387-85820-3_8
- Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer. doi: 10.1007/978-3-319-29659-3
- Surprise Library Documentation. *Scikit-Surprise – A Python scikit for building and analyzing recommender systems*. Disponible en: https://surprise.readthedocs.io/en/stable/index.html
- Wikipedia. *Sistema de recomendación*. Disponible en: https://en.wikipedia.org/wiki/Recommender_system
- Apuntes de clase. *Sistemas recomendadores*. Universidad de La Laguna, curso 2024–2025.