

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería de
Telecomunicación

Desarrollo de una aplicación para la evaluación y comparación de algoritmos de recomendación

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA TELEMÁTICA



Autor: Belén Rosa Ruz

Director: Esteban Egea López

Cartagena, 9 de diciembre de 2020

ÍNDICE

ÍNDICE DE FIGURAS.....	4
ÍNDICES DE TABLAS.....	7
INTRODUCCIÓN.....	11
Descripción del proyecto.....	11
Objetivos.....	12
Fases del desarrollo.....	12
Estructura de la memoria.....	13
TECNOLOGÍAS USADAS.....	14
Protocolos.....	14
HTTP.....	14
Lenguajes.....	15
Python.....	15
JavaScript.....	15
HTML.....	15
CSS.....	16
Frameworks.....	16
Bootstrap 4.....	16
Django.....	16
Librerías.....	17
Scikit-learn.....	17
NumPy.....	17
SciPy.....	17
Surprise.....	17
jQuery.....	18
Font awesome.....	18
Base de datos.....	18
PostgreSQL.....	18
Control de versiones.....	19
Github.....	19
DESARROLLO DEL PROYECTO.....	20
Implementación de la Web.....	20
Preparación de Django.....	20
Preparación de la base de datos.....	20
Creación de la aplicación web.....	23

Generación de recomendaciones	39
Creación de algoritmo SVD sin bias	39
Integración de algoritmos de surprise lib	40
Métricas de rendimiento	42
Comparación de algoritmos.....	43
Visualización de recomendaciones	66
CONCLUSIONES Y LÍNEAS FUTURAS.....	76
Conclusión.....	76
Dificultades encontradas	77
Líneas futuras.....	78
Bibliografía	80
Recursos web	80
Documentación Django	80
Integración base de datos PostgreSQL	80
Bootstrap	80
Font Awesome	81
NumPy.....	81
Scikit-learn	81
Creación algoritmo SVD sin bias	81
Integrar algoritmos de surprise lib.....	81
Funcionamiento AJAX	81
Tecnologías usadas	82
ANEXOS.....	83
Estructura de la base de datos.....	83
Tabla web_genre	83
Tabla web_movie.....	83
Tabla web_moviecomments.....	83
Tabla web_moviegenre.....	84
Tabla web_recs	84
Tabla web_users	84
Tabla web_user_score	85
Media ponderada	85
SGD (Stochastic gradient descent).....	85

ÍNDICE DE FIGURAS

Figura 1: Aplicaciones de Machine Learning.....	11
Figura 2: Funcionamiento del sistema.....	14
Figura 3: Funcionamiento HTTP.....	14
Figura 4: Logotipo de Python.....	15
Figura 5: Logotipo de JavaScript.....	15
Figura 6: Logotipo de HTML5.....	15
Figura 7: Logotipo de CSS3.....	16
Figura 8: Logotipo de Bootstrap.....	16
Figura 9: Logotipo de Django.....	16
Figura 10: Logotipo de Scikit-learn.....	17
Figura 11: Logotipo de NumPy.....	17
Figura 12: Logotipo de SciPy.....	17
Figura 13: Logotipo de Surprise.....	17
Figura 14: Logotipo de jQuery.....	18
Figura 15: Logotipo de Font awesome.....	18
Figura 16: Logotipo de PostgreSQL.....	18
Figura 17: Logotipo de Github.....	19
Figura 18: Página principal al acceder a admin en Django.....	21
Figura 19: Lista de objetos dentro de un model en admin.....	22
Figura 20: Campos del objeto seleccionado en admin.....	22
Figura 21: Página principal de la web.....	24
Figura 22: Página principal con la ordenación alfabética.....	25
Figura 23: Página principal con la ordenación por puntuación.....	26
Figura 24: Página 4 del catálogo de películas con la ordenación alfabética.....	26
Figura 25: Página 4 del catálogo de películas con la ordenación por puntuación.....	27
Figura 26: Página con la información de una película sin iniciar sesión y con comentarios.....	28

Figura 27: Página con la información de una película sin iniciar sesión y sin comentarios.....	28
Figura 28: Página con la información de una película sin puntuar, logeado y con comentarios.....	29
Figura 29: Página con la información de una película sin puntuar, logeado y sin comentarios.....	30
Figura 30: Página con la información de una película puntuada.....	31
Figura 31: Página con la información de una película después de puntuarla.....	31
Figura 32: Página con la información de una película introduciendo un valor fuera del rango 1-5 en la puntuación	32
Figura 33: Página con la información de una película después de haberla comentado.....	32
Figura 34: Página de inicio de sesión	33
Figura 35: Página principal después de iniciar sesión correctamente.....	33
Figura 36: Página de inicio de sesión con credenciales incorrectos	34
Figura 37: Barra de navegación con el usuario sin identificar	34
Figura 38: Barra de navegación con el usuario identificado sin haber puntuado ninguna película ...	34
Figura 39: Barra de navegación con el usuario identificado y habiendo puntuado al menos una película.....	35
Figura 40: Barra de navegación con el usuario Admin identificado sin haber puntuado ninguna película.....	35
Figura 41: Barra de navegación con el usuario Admin identificado habiendo puntuado al menos una película.....	35
Figura 42: Página de registro	35
Figura 43: Página de registro introduciendo un nombre de usuario ya existente	36
Figura 44: Página principal después de haber realizado el registro con éxito.....	36
Figura 45: Página de perfil del usuario	37
Figura 46: Página principal después de haber editado el perfil con éxito	37
Figura 47: Página de perfil del usuario después de tratar de editar su nombre por uno ya existente	38
Figura 48: Página principal después de haber cerrado sesión.....	38
Figura 49: Página de recomendaciones	67
Figura 50: Página de recomendaciones después de hacer click en uno de los botones de algoritmos	67
Figura 51: Página de recomendaciones mostrando predicciones	68

Figura 52: Página de recomendaciones después de hacer click en uno de los botones para que se muestre otra recomendación	69
Figura 53: Página de recomendaciones mostrando más de una predicción	69
Figura 54: Página en la que se pueden modificar parámetros de los algoritmos.....	70
Figura 55: Página donde se modifican parámetros con mensaje de espera mientras se ejecuta el algoritmo.....	71
Figura 56: Página donde se modifican parámetros de algoritmos mostrando recomendaciones.....	71
Figura 57: Página donde se modifican parámetros de algoritmos de Admin para un usuario	72
Figura 58: Página donde Admin escoge el usuario al que le quiere generar recomendaciones	72
Figura 59: Página donde Admin escoge el usuario al que le quiere generar recomendaciones ordenado por número de películas puntuadas de menor a mayor.....	73
Figura 60: Página 24 donde Admin escoge el usuario al que le quiere generar recomendaciones ordenado por número de películas puntuadas de menor a mayor.....	73
Figura 61: Página de recomendaciones de Admin para otro usuario.....	74
Figura 62: Página de recomendaciones de Admin para otro usuario.....	75
Figura 63: Estructura de la tabla web_genre	83
Figura 64: Estructura de la tabla web_movie	83
Figura 65: Estructura de la tabla web_moviecomments	83
Figura 66: Estructura de la tabla web_moviegenre	84
Figura 67: Estructura de la tabla web_recgs.....	84
Figura 68: Estructura de la tabla web_users.....	84
Figura 69: Estructura de la tabla web_user_score	85

ÍNDICES DE TABLAS

Tabla 1: Recomendaciones de todos los algoritmos para Scott (2 iteraciones).	43
Tabla 2: Recomendaciones de todos los algoritmos para Adrian (2 iteraciones).	44
Tabla 3: Recomendaciones de todos los algoritmos para Jeremy (2 iteraciones).	44
Tabla 4: Recomendaciones de todos los algoritmos para Olivar (2 iteraciones).	44
Tabla 5: Recomendaciones de todos los algoritmos para Arias (2 iteraciones).	44
Tabla 6: Recomendaciones de todos los algoritmos para Pepe (2 iteraciones).	45
Tabla 7: Media de iteraciones para SVD.	45
Tabla 8: Media de iteraciones para NormalPredictor.	45
Tabla 9: Media de iteraciones para CoClustering.	46
Tabla 10: KNNBasic con $k = 40$ y $\text{min}_k = 10$.	47
Tabla 11: KNNBasic con $k = 40$ y $\text{min}_k = 20$.	47
Tabla 12: KNNBasic con $k = 40$ y $\text{min}_k = 40$.	47
Tabla 13: KNNBasic con $k = 150$ y $\text{min}_k = 100$.	47
Tabla 14: KNNBasic con $k = 150$ y $\text{min}_k = 150$.	48
Tabla 15: KNNBasic con $k = 3000$ y $\text{min}_k = 3000$.	48
Tabla 16: KNNBasic con $k = 30$ y $\text{min}_k = 1$.	48
Tabla 17: KNNBasic con $k = 20$ y $\text{min}_k = 1$.	48
Tabla 18: KNNBasic con $k = 1$ y $\text{min}_k = 1$.	49
Tabla 19: CoClustering con $n_{\text{cltr}_u} = 20$, $n_{\text{cltr}_i} = 100$ y $n_{\text{epochs}} = 20$.	50
Tabla 20: CoClustering con $n_{\text{cltr}_u} = 5$, $n_{\text{cltr}_i} = 6$ y $n_{\text{epochs}} = 20$.	50
Tabla 21: CoClustering con $n_{\text{cltr}_u} = 5$, $n_{\text{cltr}_i} = 5$ y $n_{\text{epochs}} = 20$.	50
Tabla 22: CoClustering con $n_{\text{cltr}_u} = 2$, $n_{\text{cltr}_i} = 3$ y $n_{\text{epochs}} = 20$.	50
Tabla 23: CoClustering con $n_{\text{cltr}_u} = 2$, $n_{\text{cltr}_i} = 2$ y $n_{\text{epochs}} = 20$.	51
Tabla 24: CoClustering con $n_{\text{cltr}_u} = 1$, $n_{\text{cltr}_i} = 1$ y $n_{\text{epochs}} = 20$.	51
Tabla 25: CoClustering con $n_{\text{cltr}_u} = 3$, $n_{\text{cltr}_i} = 3$ y $n_{\text{epochs}} = 15$.	51
Tabla 26: CoClustering con $n_{\text{cltr}_u} = 3$, $n_{\text{cltr}_i} = 3$ y $n_{\text{epochs}} = 30$.	51
Tabla 27: CoClustering con $n_{\text{cltr}_u} = 3$, $n_{\text{cltr}_i} = 3$ y $n_{\text{epochs}} = 40$.	52

Tabla 28: CoClustering con $n_cltr_u = 3$, $n_cltr_i = 3$ y $n_epochs = 60$.	52
Tabla 29: CoClustering con $n_cltr_u = 20$, $n_cltr_i = 100$ y $n_epochs = 40$.	52
Tabla 30: CoClustering con $n_cltr_u = 1$, $n_cltr_i = 1$ y $n_epochs = 5$.	52
Tabla 31: CoClustering con $n_cltr_u = 1$, $n_cltr_i = 1$ y $n_epochs = 1$.	53
Tabla 32: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	54
Tabla 33: SVD con $n_factors = 50$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	54
Tabla 34: SVD con $n_factors = 80$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	54
Tabla 35: : SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	54
Tabla 36: SVD con $n_factors = 800$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	55
Tabla 37: SVD con $n_factors = 2000$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	55
Tabla 38: SVD con $n_factors = 10000$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	55
Tabla 39: SVD con $n_factors = 100$, $n_epochs = 1$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	56
Tabla 40: SVD con $n_factors = 100$, $n_epochs = 10$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	56
Tabla 41: SVD con $n_factors = 100$, $n_epochs = 25$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	56
Tabla 42: SVD con $n_factors = 100$, $n_epochs = 45$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	56
Tabla 43: SVD con $n_factors = 500$, $n_epochs = 25$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	57
Tabla 44: SVD con $n_factors = 500$, $n_epochs = 10$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	57
Tabla 45: SVD con $n_factors = 1$, $n_epochs = 25$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	57
Tabla 46: SVD con $n_factors = 1$, $n_epochs = 10$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	58
Tabla 47: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0.06$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.	58

Tabla 48: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0.2$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$	58
Tabla 49: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.2$, $lr_all = 0.005$ y $reg_all = 0.02$	59
Tabla 50: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.05$, $lr_all = 0.005$ y $reg_all = 0.02$	59
Tabla 51: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0.2$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$	59
Tabla 52: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.2$, $lr_all = 0.005$ y $reg_all = 0.02$	60
Tabla 53: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.05$, $lr_all = 0.005$ y $reg_all = 0.02$	60
Tabla 54: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0.2$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$	60
Tabla 55: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.2$, $lr_all = 0.005$ y $reg_all = 0.02$	60
Tabla 56: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.05$, $lr_all = 0.005$ y $reg_all = 0.02$	61
Tabla 57: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0$ y $reg_all = 0.02$	61
Tabla 58: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.003$ y $reg_all = 0.02$	61
Tabla 59: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.007$ y $reg_all = 0.02$	62
Tabla 60: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.01$	62
Tabla 61: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.04$	62
Tabla 62: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.1$	62
Tabla 63: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.003$ y $reg_all = 0.02$	63
Tabla 64: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.007$ y $reg_all = 0.02$	63
Tabla 65: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.04$	63
Tabla 66: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.1$	64

Tabla 67: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.003$ y $reg_all = 0.02$.	64
Tabla 68: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.007$ y $reg_all = 0.02$.	64
Tabla 69: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.04$.	64
Tabla 70: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.1$.	65
Tabla 71: SVD con $n_factors = 1$, $n_epochs = 1$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0$ y $reg_all = 0.03$.	65
Tabla 72: SVD con $n_factors = 1$, $n_epochs = 1$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0$ y $reg_all = 0.1$.	65

INTRODUCCIÓN

Descripción del proyecto

El proyecto se basa en la creación de una web de películas donde se prueban diversos algoritmos de recomendación. Todo esto se lleva a cabo mediante Machine Learning, que hace referencia a la capacidad de una máquina o software para aprender mediante la adaptación de ciertos algoritmos de su programación respecto a cierta entrada de datos en su sistema. Es decir, los algoritmos que se encargan de generar las recomendaciones hacen previamente unos cálculos, sin necesidad de intervención humana, en función de los datos recopilados en el sistema y proporciona como resultado una predicción de películas, en este caso, que le pueden gustar al usuario. Cuantos más datos recopile el algoritmo, mayor será su precisión a la hora de generar predicciones.

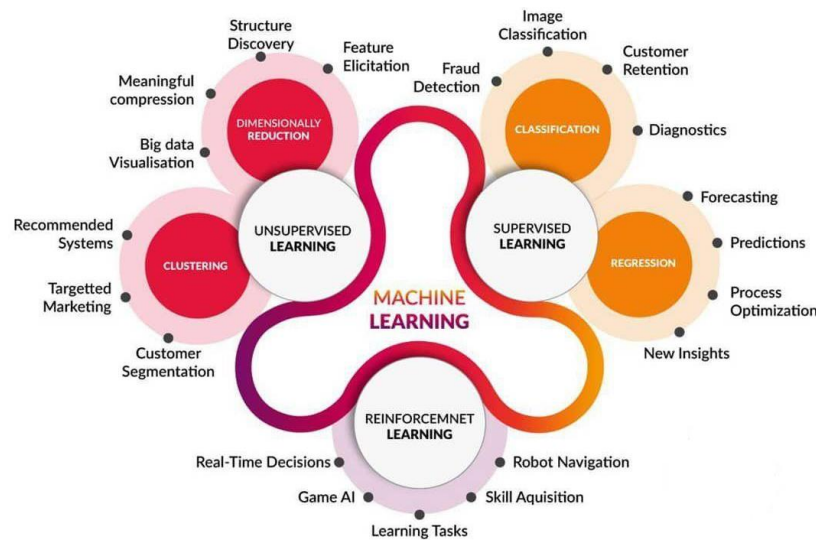


Figura 1: Aplicaciones de Machine Learning

La utilización de estos algoritmos mediante Machine Learning se han dado a conocer, sobre todo, en Netflix (una plataforma donde se pueden ver series y películas). De hecho, esta compañía hizo una competición para que las predicciones a la hora de recomendar películas o series recogiera todos los datos posibles, desde las películas que ha escogido el usuario para ver, hasta el tipo de portada que llama la atención de éste.

En esta web, cuando un usuario accede a ella, le aparece una serie de películas con unos detalles (fecha, descripción, puntuación...) y desde aquí pueden iniciar sesión, registrarse o clickar en la película que le haya llamado la atención. Para poder puntuarla o dejar un comentario debe haber iniciado sesión y, una vez que ha puntuado las películas que le haya gustado, puede acceder a recomendaciones, donde elegirá en qué algoritmos se basa dicha generación de recomendaciones. En función de las puntuaciones del usuario, el algoritmo seleccionado le mostrará las tres películas predichas que más le pueden gustar. El usuario también puede cambiar ciertos parámetros de los algoritmos si desea encontrar una recomendación de mejor calidad.

Objetivos

Los objetivos de este trabajo son:

- Saber desarrollar una web en Python con ayuda de Django.
- Incorporar a la web mi propio algoritmo de recomendación de filtrado colaborativo.
- Integrar y comparar diferentes algoritmos de recomendación de la librería surprise.
- Interpretar las métricas de rendimiento de cada algoritmo.

Fases del desarrollo

A la hora de abordar el trabajo, se organizó en varias fases:

1. Introducción en Python.

Se llevaron a cabo diversos cursos para aprender lo básico sobre Python, ya que para poder lograr el objetivo de comparar los algoritmos de la librería surprise, es necesario desarrollar la web en Python. Además, el algoritmo que se intenta implementar desde cero también se desarrolla en este lenguaje.

2. Investigación de frameworks.

Para desarrollar una web en Python, hay varios frameworks que ayudan a no empezar a elaborar una web desde cero. Los que más destacan son Flask, Django y TurboGears. Flask se usa más para aplicaciones sencillas, Django parece más completo a la hora de elaborar una web y TurboGears parece que mejora algunos detalles de Django, pero no se hallaba tanta documentación como para Django. Así que la web se realiza con Django.

3. Creación de la aplicación web.

Cuando ya está preparado el entorno de trabajo para poder desarrollar la web, se crea la aplicación en Django y, a continuación, se comienza a desarrollar la web de películas cumpliendo una serie de requisitos.

4. Investigación de la librería surprise.

Una vez que se ha terminado de desarrollar la web a excepción de la parte orientada a algoritmos, buscamos documentación de cómo funciona la librería surprise y cómo se integran a la web sus algoritmos.

5. Integración de algoritmos y métricas de rendimiento.

Se integra el algoritmo de filtrado colaborativo implementado desde cero además de los de la librería surprise en la web. Una vez que han sido integrados, se muestran las predicciones de estos al usuario. De tal manera que un usuario

logueado que ha puntuado anteriormente alguna película, pueda acceder a esta parte del proyecto.

6. Elaboración de la documentación.

Una vez que se han terminado todas las fases anteriores, se comienza a redactar la documentación del proyecto.

Estructura de la memoria

En primer lugar, se encuentra la introducción donde se explica de qué trata el proyecto, los objetivos que se quieren cumplir, las fases de desarrollo en las que se ha organizado el trabajo para su elaboración y la estructura de la memoria donde se indica qué se expone en cada apartado.

A continuación, se describen las tecnologías usadas, tales como lenguajes de programación, librerías y frameworks que se han requerido para la realización del proyecto.

En el desarrollo se explican todos los pasos que se han llevado a cabo durante el desarrollo y el funcionamiento de la web. Este apartado se divide en dos partes, en la primera se explica el desarrollo de la web para poder realizar la comparación de algoritmos, y en la segunda, lo referente a la parte de algoritmos.

En conclusiones y líneas futuras, se exponen las conclusiones finales del proyecto, las dificultades encontradas y líneas futuras que pueden mejorar el trabajo.

Después, encontramos la bibliografía y recursos web en los que se muestran las fuentes de información que han ayudado a desarrollar este proyecto.

Por último, a continuación de la bibliografía, se encuentran los anexos que pueden ayudar a comprender mejor ciertas partes y complementar con más información.

TECNOLOGÍAS USADAS

Son varias las tecnologías usadas, a continuación se muestra una figura ilustrativa del funcionamiento del sistema, en el cual cada una de las partes se explican más adelante.

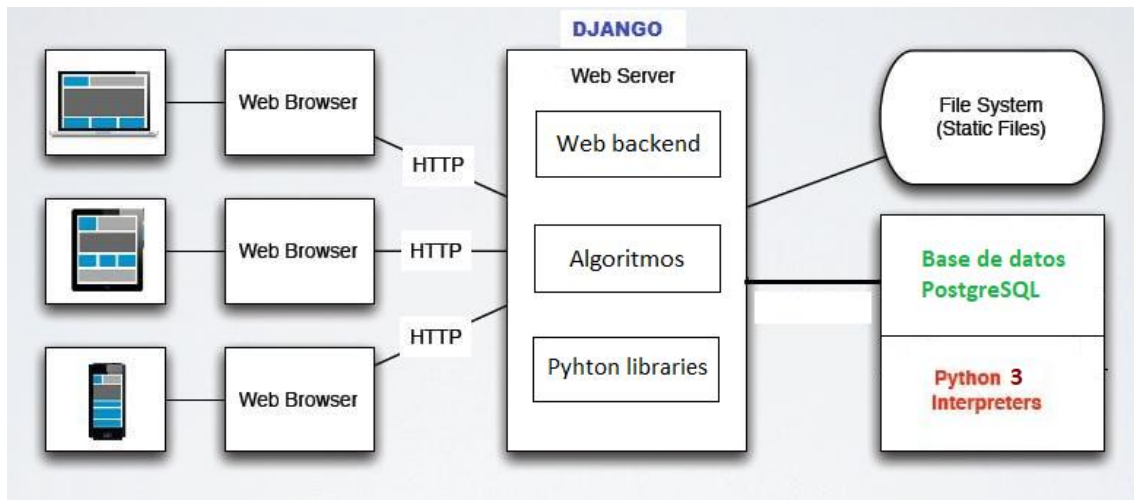


Figura 2: Funcionamiento del sistema

Protocolos

HTTP

HTTP, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML. Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador Web. Así, una página web completa resulta de la unión de distintos sub-documentos recibidos, como, por ejemplo: un documento que especifique el estilo de maquetación de la página web (CSS), el texto, las imágenes, vídeos, scripts, etc...

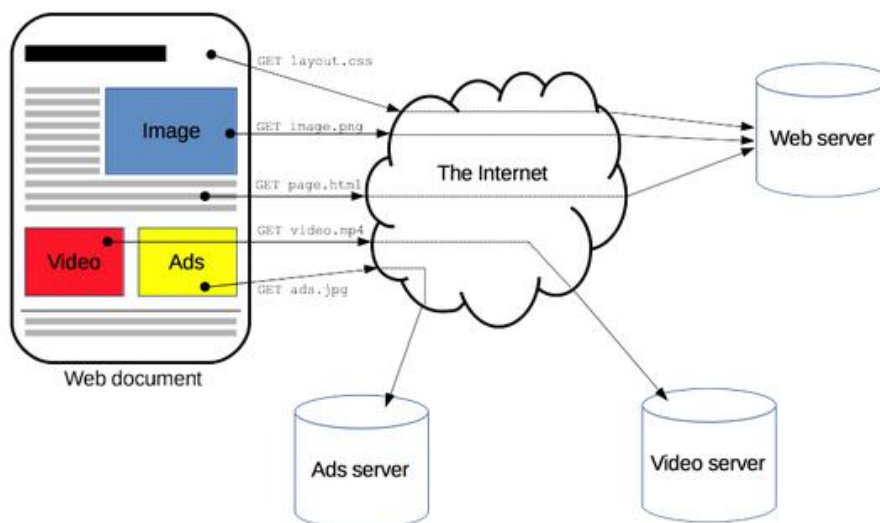


Figura 3: Funcionamiento HTTP

Lenguajes

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico, multiplataforma y, además, posee una licencia de código abierto.



Figura 4: Logotipo de Python

JavaScript

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas³ y JavaScript del lado del servidor (Server-side JavaScript o SSJS).



Figura 5: Logotipo de JavaScript

HTML

HTML, siglas en inglés de HyperText Markup Language ('lenguaje de marcas de hipertexto'), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros.



Figura 6: Logotipo de HTML5

El lenguaje HTML basa su filosofía de desarrollo en la diferenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene solamente texto mientras que recae en el navegador web (intérprete del código) la tarea de unir todos los elementos y visualizar la página final.

CSS

CSS (siglas en inglés de Cascading Style Sheets), en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML. Te puede ayudar a crear tu propio sitio web. Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web y GUIs para muchas aplicaciones móviles (como Firefox OS).



Figura 7: Logotipo de CSS3

CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes.

Frameworks

Bootstrap 4

Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.



Figura 8: Logotipo de Bootstrap

También destaca por su fácil manejo, al igual que a la hora de enlazarlo con la aplicación, que únicamente ha habido que usar la herramienta pip.

Django

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como MVC (Modelo–Vista–Controlador).



Figura 9: Logotipo de Django

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

Librerías

Scikit-learn

Scikit-learn es una biblioteca para aprendizaje automático de software libre para el lenguaje de programación Python. Incluye varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte, bosques aleatorios, Gradient boosting, K-means y DBSCAN. Está diseñada para interoperar con las bibliotecas numéricas y científicas NumPy y SciPy.



Figura 10: Logotipo de Scikit-learn

NumPy

NumPy es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas. En 2005, Travis Oliphant creó NumPy incorporando características de la competencia Numarray en Numeric, con amplias modificaciones. NumPy es un software de código abierto y cuenta con muchos colaboradores.



Figura 11: Logotipo de NumPy

SciPy

SciPy es una biblioteca libre y de código abierto para Python. Se compone de herramientas y algoritmos matemáticos.

SciPy contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales y de imagen, resolución de ODEs y otras tareas para la ciencia e ingeniería.

SciPy se basa en el objeto de matriz NumPy y es parte del conjunto NumPy, que incluye herramientas como Matplotlib, pandas y SymPy, y un conjunto en expansión de bibliotecas de computación científica.



Figura 12: Logotipo de SciPy

Surprise

Surprise es una librería de Python para analizar sistemas de recomendación. Esta librería ha sido diseñada para ofrecer a los usuarios un control sobre los experimentos que realicen, por eso tratan de redactar una documentación clara y precisa. En esta librería se proporcionan varios algoritmos ya implementados listos para usar, como por



Figura 13: Logotipo de Surprise

ejemplo vecinos cercanos y factorización de matrices. Además, se proveen herramientas para evaluar, analizar y comparar el rendimiento de los algoritmos.

jQuery

jQuery es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es software libre y de código abierto. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.



Figura 14: Logotipo de jQuery

Font awesome

Font Awesome es un framework de iconos vectoriales y estilos css. Este framework es utilizado para sustituir imágenes de iconos comunes por gráficos vectoriales convertidos en fuentes. Para ello utiliza una librería de más de 400 iconos transformadas en fuentes.



Figura 15: Logotipo de Font awesome

Tiene un fácil manejo a la hora de insertar los iconos, únicamente se añade su código en el lugar del HTML que quieras que aparezca. Y al enlazarlo con Django sólo ha sido necesario instalar font awesome con la herramienta pip.

Base de datos

PostgreSQL

PostgreSQL es un potente sistema de base de datos relacional de objetos de código abierto que utiliza y amplía el lenguaje SQL en combinación con muchas otras características. PostgreSQL se ha ganado una sólida reputación por su arquitectura probada, confiabilidad, integridad de datos, conjunto sólido de características, extensibilidad y la dedicación de la comunidad de código abierto detrás del software para ofrecer soluciones innovadoras y de alto rendimiento de manera consistente. PostgreSQL se ejecuta en todos los sistemas operativos y ha sido compatible con ACID desde 2001. Para aprender PostgreSQL no necesita mucha capacitación, ya que es fácil de usar.



Figura 16: Logotipo de PostgreSQL

Es el más recomendado a la hora de trabajar con Django ya que hay tipos de datos que solo son soportados por PostgreSQL además de otras muchas ventajas.

Control de versiones

Github

Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargar la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.



Figura 17: Logotipo de Github

Como su nombre indica, la web utiliza el sistema de control de versiones Git diseñado por Linus Torvalds. Un sistema de gestión de versiones es ese con el que los desarrolladores pueden administrar su proyecto, ordenando el código de cada una de las nuevas versiones que sacan de sus aplicaciones para evitar confusiones. Así, al tener copias de cada una de las versiones de su aplicación, no se perderán los estados anteriores cuando se va a actualizar.

DESARROLLO DEL PROYECTO

Implementación de la Web

Preparación de Django

A la hora de empezar el proyecto, se investigó qué frameworks recomendaban para desarrollar una web, como ya se ha mencionado anteriormente. Una vez que se escogió Django, con ayuda de un tutorial, el entorno de trabajo fue preparado. Se instaló Python y se preparó el directorio en el que se iba a trabajar. Cuando ya estaba todo preparado, con ayuda del símbolo del sistema (también conocido como CMD), creamos el proyecto en el que posteriormente se insertará la aplicación web. Al crear el proyecto se crean una serie de archivos y carpetas que se van a ir editando para cumplir con las necesidades de la web.

Preparación de la base de datos

Django utiliza por defecto la base de datos SQLite, pero se pueden utilizar otras que se adapten más a las necesidades de la web, tales como MySQL o PostgreSQL entre otras. Después de decidir que PostgreSQL era la mejor opción para la web que se va a desarrollar, este se instala junto con psycopg2 que es el que se encarga de la comunicación entre Django y PostgreSQL.

Uno de los archivos que se han creado en el proyecto, que se llama settings.py, es el que se edita para que, al iniciar la aplicación, Django utilice como base de datos PostgreSQL. En este archivo se deben cambiar los datos que hacen que Django use SQLite e insertar la base de datos que queremos que utilice (nombre, contraseña, usuario, host, engine y puerto) para que pueda acceder a ella.

Para poder establecer las tablas en la base de datos, se necesita crear la aplicación web en Django que generará otra serie de archivos entre los cuales uno de ellos, models.py, será clave para la creación de tablas. Antes de crear dichas tablas con ayuda de este archivo, se debe acceder a PostgreSQL (a través del comando psql) a través del símbolo del sistema y crear la base de datos a la que queremos que acceda Django.

Ahora que ya está todo preparado, abrimos el archivo models.py y creamos las tablas que se necesitan para alojar los datos necesarios, que se han obtenido del dataset de Movielens. Cada class es una tabla de la base de datos, y dentro de cada class se definen las columnas y su tipo (si es de tipo integer, text...). Además, también hay que especificar si se quiere que alguna columna sea clave primaria, que en caso contrario, se autogenera una columna llamada id de tipo integer que hace de clave primaria. También se ha dado la situación en la que se quería que dos columnas lo fueran, en este caso lo pueden ser, pero especificándolo de una manera distinta (la combinación de estas dos columnas no se puede repetir, es decir, actúan como clave primaria pero no lo son) y, además, se

autogenera la columna id de tipo integer al igual que en el caso anterior descrito. PostgreSQL no acepta tablas sin que ninguna de las columnas sea clave primaria.

Ahora que ya se han definido todas las clases necesarias para la creación de tablas en models.py, se hacen las migraciones pertinentes en el símbolo del sistema para que se registren en PostgreSQL.

A continuación, en el símbolo del sistema, se puede acceder a la base de datos de la web en psql y con ayuda del comando COPY se introducen los datos de cada tabla. Como se ha comentado anteriormente, PostgreSQL no admite tablas sin claves primarias, por lo que se ha tenido que añadir una columna que actúe de clave primaria. En las tablas de MovieLens esta columna no aparece, por lo que en el archivo .csv de cada tabla se han tenido que añadir los valores de la columna id que actúa de clave primaria y poder introducir los datos en la base de datos. Para no tener que hacerlo manualmente nos ayudamos de un script que lee el archivo .csv, lo reescribe añadiendo una columna inicial correspondiente a la id y este valor se incrementa en 1 en cada fila. Así que, una vez que ya se ha solucionado este problema, ya se pueden importar todas las tablas en la base de datos^[1].

Finalmente, en admin.py (que se encuentra entre los archivos de la aplicación) se introduce el código necesario para que aparezcan los datos de las tablas que se necesitan a la hora de desarrollar bien la web y comprobar su funcionamiento. Cada class en el código es una tabla que se quiere mostrar. A esta página se accede añadiendo a la url /admin/ y se rellenan los campos del nombre del superuser y la contraseña. Una vez logeado aparecen los campos de los models que se han añadido en admin:

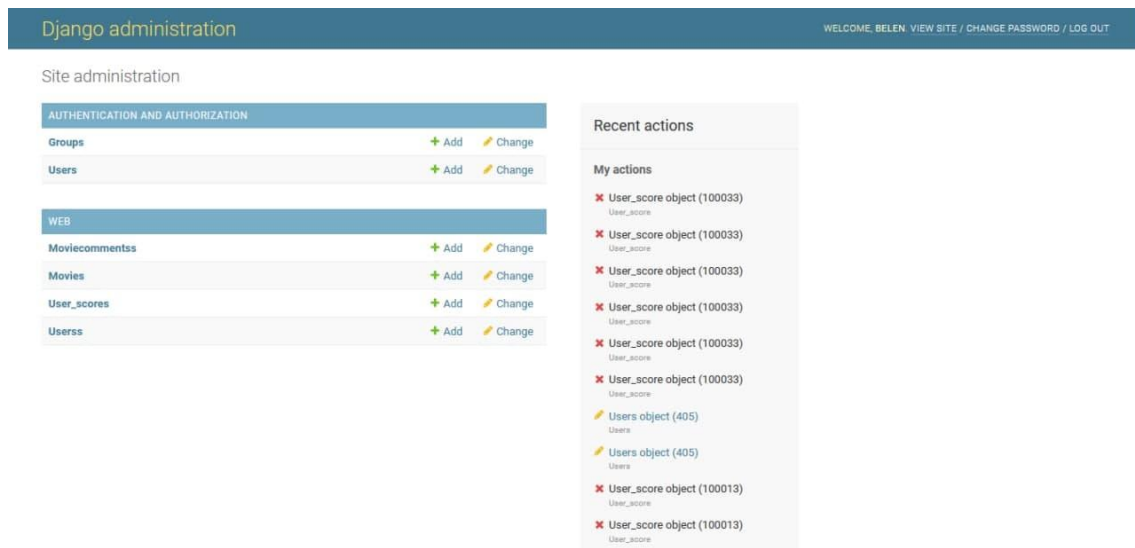


Figura 18: Página principal al acceder a admin en Django

Para ver la información de una película, por ejemplo, accedemos a Movies y aparece una lista con las películas que hay almacenadas:

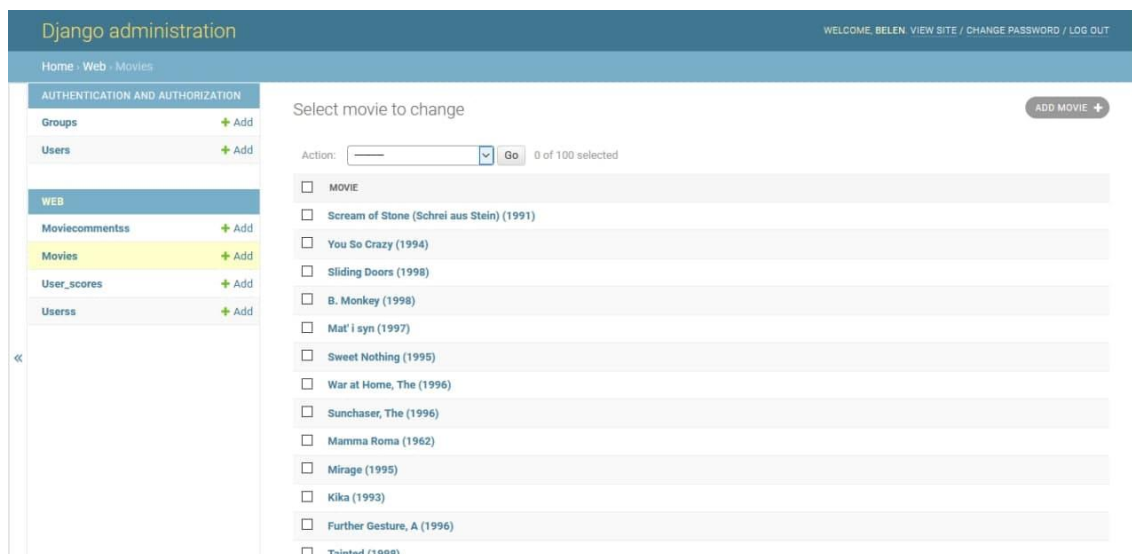


Figura 19: Lista de objetos dentro de un model en admin.

Si se hace click en uno de los objetos, se muestran lo que corresponde con las columnas de la base de datos junto con sus valores en cada uno de los campos:

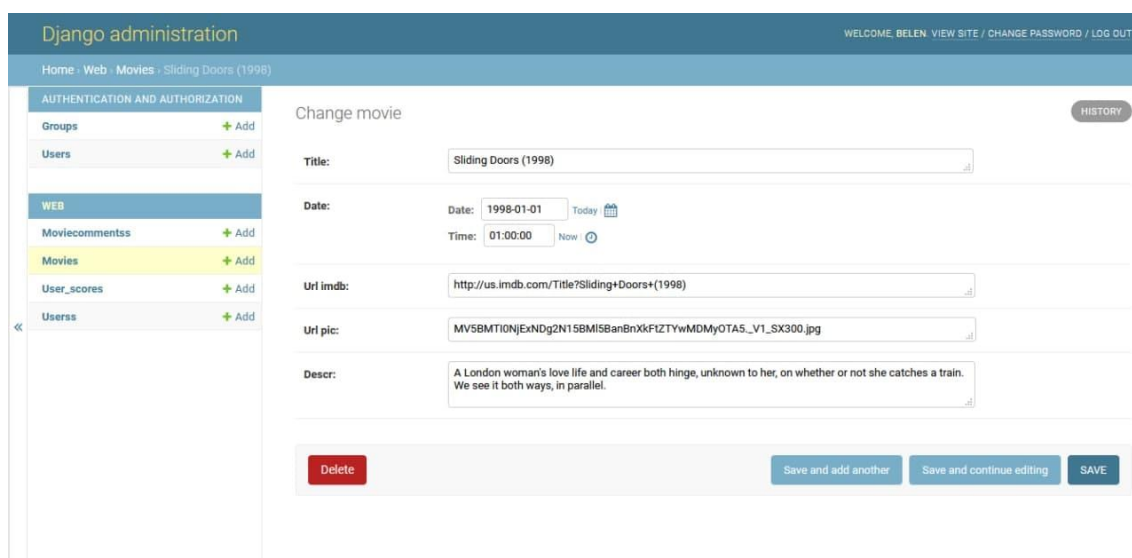


Figura 20: Campos del objeto seleccionado en admin

Creación de la aplicación web

Como ya se ha creado la aplicación en Django para poder preparar la base de datos, sólo queda editar la aplicación para que cumpla con los requisitos. Para esta parte se debe modificar `views.py`, que se encarga de pasar los valores que se quieren mostrar en el archivo HTML. Como para desarrollar la web, se van a necesitar varias vistas, necesitamos crear un archivo llamado `urls.py`, en el que se especifica con qué URL se accede a cada una de las vistas. A continuación, se explica más detalladamente cada una de las vistas.

Index

Esta es la que actúa de página principal y su objetivo es mostrar y permitir la navegación por todo el catálogo de películas. Para esto, se muestra:

- Imagen con el póster de la película.
- Botón que redirige a una nueva página que mostrará el contenido de la vista `movie` (explicado más adelante).
- Descripción de la película.
- Fecha de estreno.
- Puntuación media de la película, número de puntuaciones recibidas y puntuación ponderada^[2] de la película.
- Se debe permitir la reordenación de los elementos en base a los campos nombre y puntuación media.

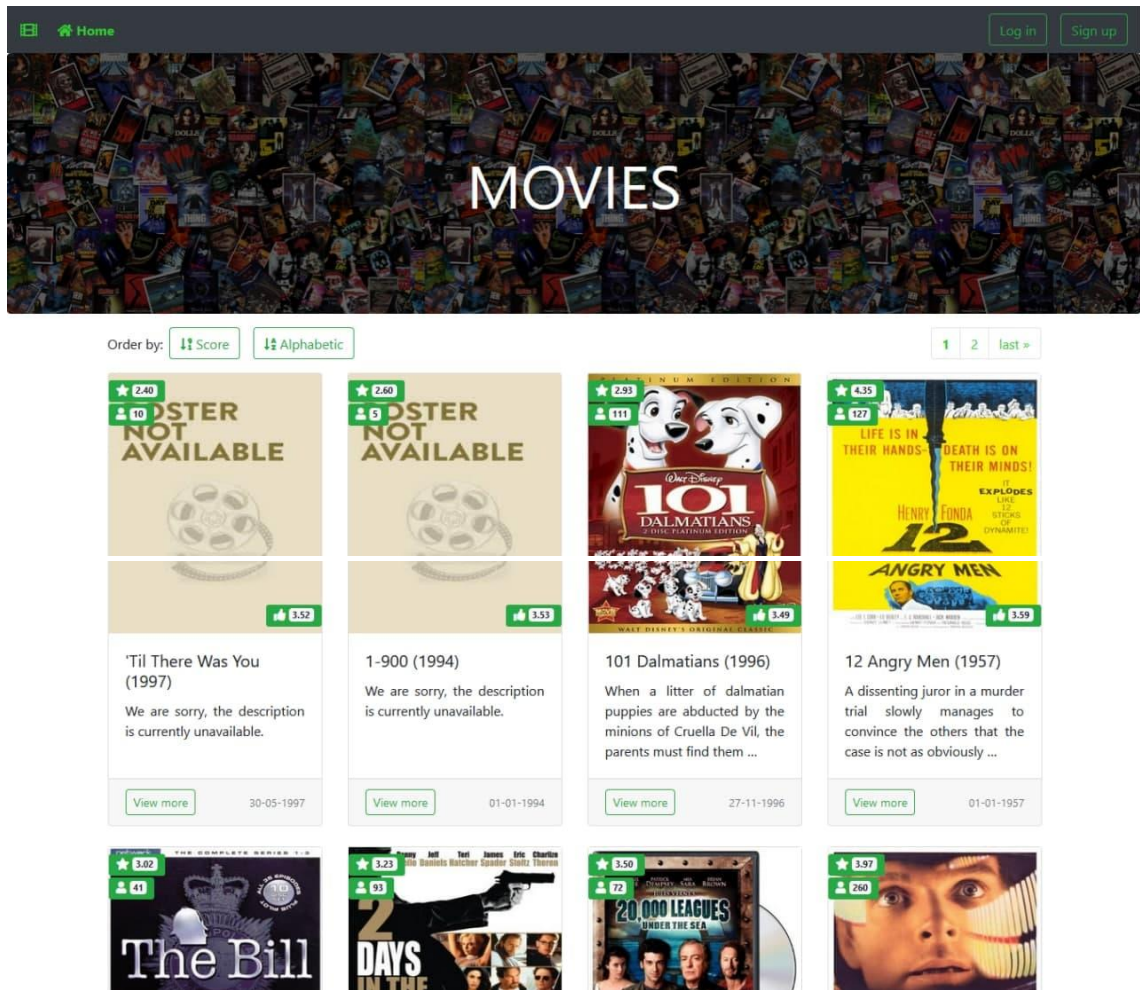


Figura 21: Página principal de la web

En views.py se recogen todos los datos de cada película que hay en la base de datos: el título, el nombre de la imagen, la fecha de estreno, la descripción, la puntuación media y el número de personas que la han votado. A continuación, en el archivo HTML (index.html) se obtienen todos estos datos y son mostrados de diversas maneras.

Como se puede ver en la figura anterior, cada película dispone de un botón en el que pone "View more". Este botón redirige a la página en la que se pueden saber todos los detalles de esa película (más adelante se explica con más precisión que se muestra en esta página).

La descripción de la película no se muestra en su totalidad en la página principal, acortándose con un tag disponible en Django de manera que cuando la descripción llega a un número "x" de palabras, aparecen puntos suspensivos. También en caso de que no haya disponible una descripción, con un tag personalizado se ha añadido una descripción por defecto que dice "We are sorry, the description is currently unavailable."

Para mostrar la imagen se obtiene su nombre y se muestra poniendo la ruta. En el caso de que ese campo esté vacío en la base de datos o no se encuentre el archivo,

con ayuda de un tag personalizado se comprueba dicha ruta y se muestra una imagen de poster no disponible por defecto.

La puntuación media, el número de personas que han votado por cada película y su media ponderada, se sitúan donde se encuentra la imagen de la portada. A la hora de mostrar la media ponderada, se utiliza un tag personalizado que pasa como argumento la id de la película y devuelve el valor calculado para la media ponderada. En el código de este tag se utiliza la id de la película que ayuda a recoger todos los datos que se necesitan de esa película para calcular la media ponderada y, una vez calculada, se devuelve el resultado para que se muestre en el HTML.

Entre el jumbotron y el catálogo de películas, a la izquierda se muestran dos botones, estos son para permitir a la persona que navegue por la web la reordenación de las películas. Por defecto vienen ordenadas alfabéticamente, pero hay un botón que te permite reordenarlas por puntuación media. También se puede volver a la ordenación por defecto clickando en el botón correspondiente. El funcionamiento de estos botones de reordenación se ha realizado pasando un parámetro de tipo GET, de esta manera en la vista se comprueba si se ha pasado un valor para dicho parámetro y recoger las películas en el orden correspondiente de la base de datos. Para no perder este orden cuando pasas de página y se cargue la ordenación que hay por defecto, hemos usado una sesión. Así, en la vista se comprueba si hay una sesión que ordene cargar los datos de la base de datos en el orden que se indica.

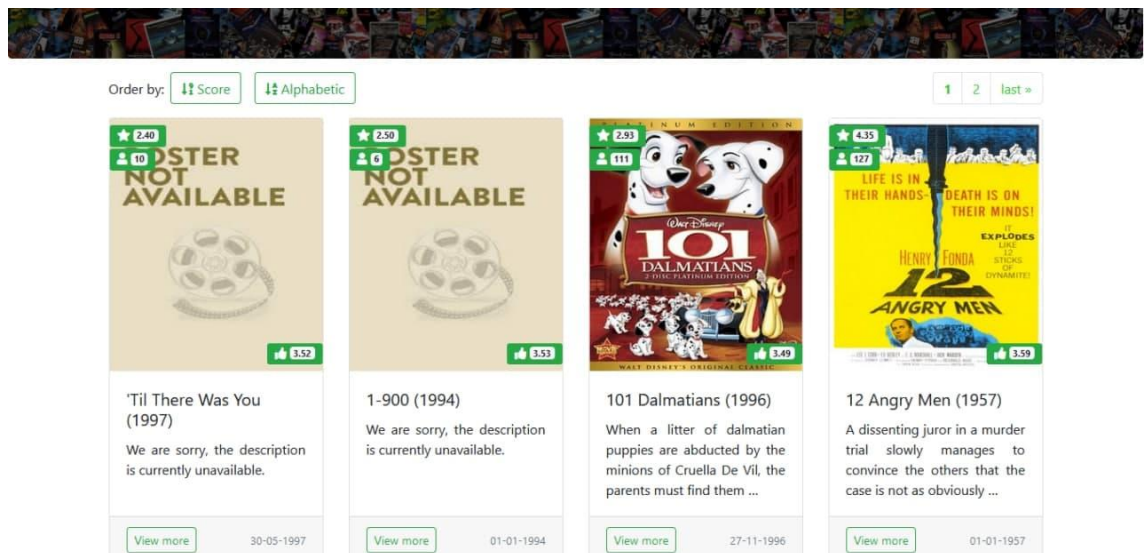


Figura 22: Página principal con la ordenación alfabética

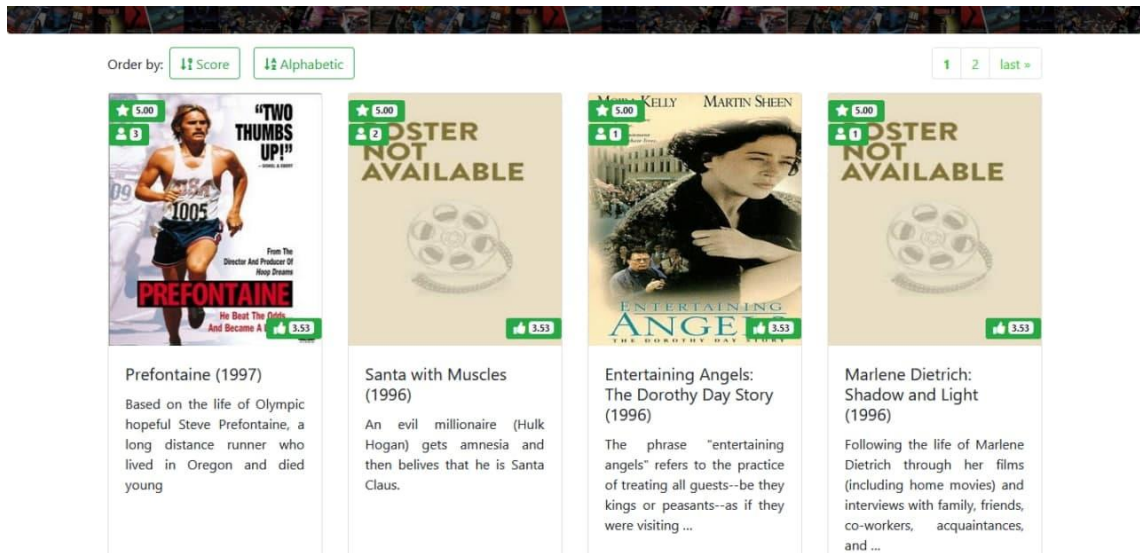


Figura 23: Página principal con la ordenación por puntuación

Para que no cargara la página todo el catálogo de películas, ya que son demasiadas y tardaría mucho, se ha decidido poner una paginación en la que se muestran 12 películas por página. Al igual que con la reordenación, también se ha usado un parámetro tipo GET que se obtiene en la vista para mostrar las películas correspondientes a esa página. La paginación permite acceder a la página siguiente, a la anterior, a la primera página y a la última.

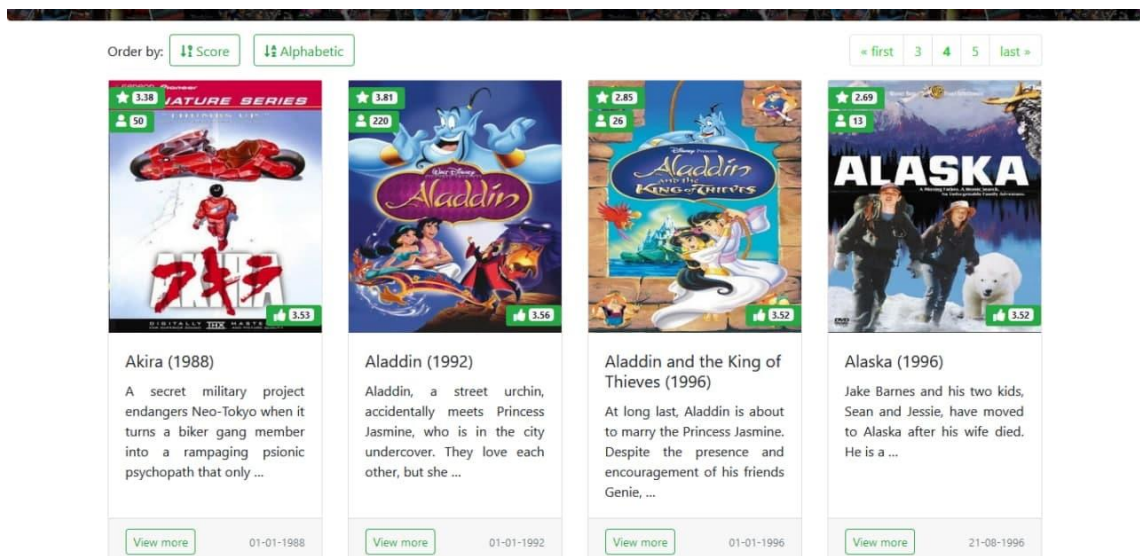


Figura 24: Página 4 del catálogo de películas con la ordenación alfabética

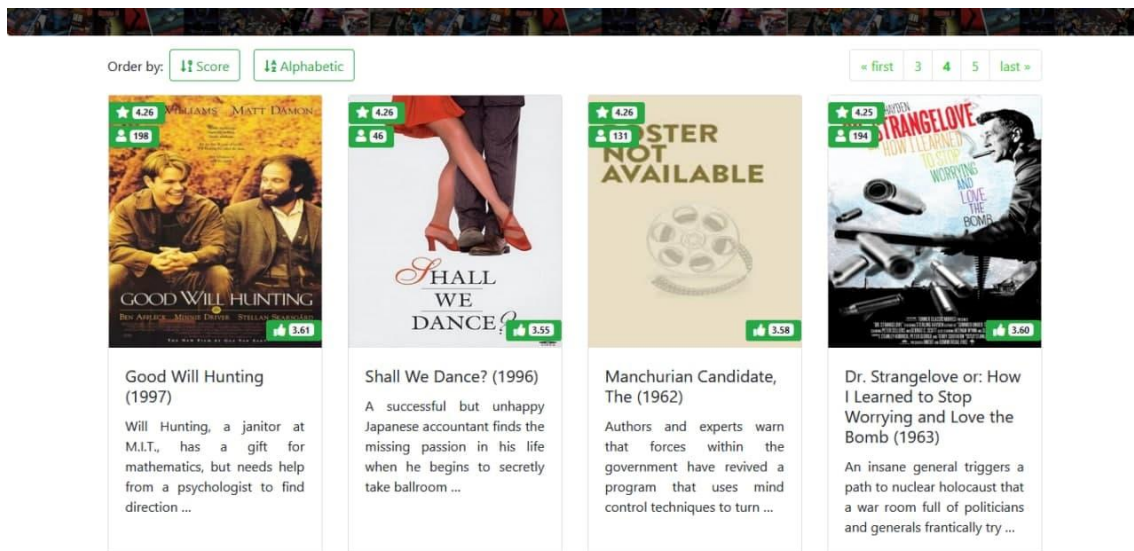


Figura 25: Página 4 del catálogo de películas con la ordenación por puntuación

Movie

A esta página se accede mediante la página principal clickando en el botón “View more” de cualquiera de las películas del catálogo. Nos redirige aquí mediante la url que busca por id de la película, y se muestra la siguiente información:

- Los campos asociados a la película que se muestran en la página principal.
- Descripción completa.
- Los géneros a los que pertenece.
- Los comentarios recibidos con el nombre del usuario que lo realizó.
- Puntuación recibida por el usuario (si la ha puntuado ya) si el usuario se ha identificado y ha iniciado una sesión.
- Posibilidad de puntuar o cambiar la puntuación si el usuario se ha identificado y ha iniciado una sesión.
- Posibilidad de añadir un comentario si el usuario se ha identificado y ha iniciado una sesión.

Cuando el usuario no se ha registrado o iniciado sesión, al navegar por esta página, aparece un jumbotron con la imagen de la película, el título, la descripción completa, los géneros a los que pertenece y el enlace a IMDB. A continuación, en la columna de la izquierda se muestra la puntuación media, el número de usuarios que han votado y la puntuación media ponderada. En la columna de la derecha le sugiere al usuario que inicie sesión o se registre para poder dejar un comentario. Y, finalmente, en la parte de abajo aparece una tabla con los comentarios que han dejado los usuarios y el nombre del usuario al que corresponde. En caso de no haber comentarios, hay un mensaje que informa de ello.

Home Log in Sign up

GoldenEye (1995)

01-01-1995

Overview
James Bond teams up with the lone survivor of a destroyed Russian research center to stop the hijacking of a nuclear space weapon by a fellow agent believed to be dead.

Genres
Adventure Animation War

[Enlace a IMDB](#)

Puntuación ponderada: 3.51
 Puntuación media: 3.23
 Personas que han votado: 132

Déjanos un comentario para saber que te ha parecido la película:

Inicia sesión o regístrate para darnos tu opinión.

Comments

User name	Comment
Eugene	h
Eugene	o
Eugene	a

Figura 26: Página con la información de una película sin iniciar sesión y con comentarios

Home Log in Sign up

12 Angry Men (1957)

01-01-1957

Overview
A dissenting juror in a murder trial slowly manages to convince the others that the case is not as obviously clear as it seemed in court.

Genres
Fantasy

[Enlace a IMDB](#)

Puntuación ponderada: 3.59
 Puntuación media: 4.35
 Personas que han votado: 127

Déjanos un comentario para saber que te ha parecido la película:

Inicia sesión o regístrate para darnos tu opinión.

Aún no hay ningún comentario, sé el primero en darnos tu opinión.

Figura 27: Página con la información de una película sin iniciar sesión y sin comentarios

Toda esta información se recopila desde la base de datos con ayuda de consultas en views.py y, en el archivo HTML, se recoge toda la información para mostrarla como se ha explicado anteriormente.

Si el usuario ha iniciado sesión en la web, debajo del jumbotron, en la columna de la izquierda se encuentra un formulario en el que puedes puntuar la película del 1 al 5 y, a continuación, las puntuaciones que ha recibido la película (al igual que cuando no has iniciado sesión). En la columna de la derecha, hay otro formulario que te permite dejar un comentario. Finalmente, en la parte de abajo aparecen los comentarios en una tabla si los hay al igual que si no se ha iniciado sesión.

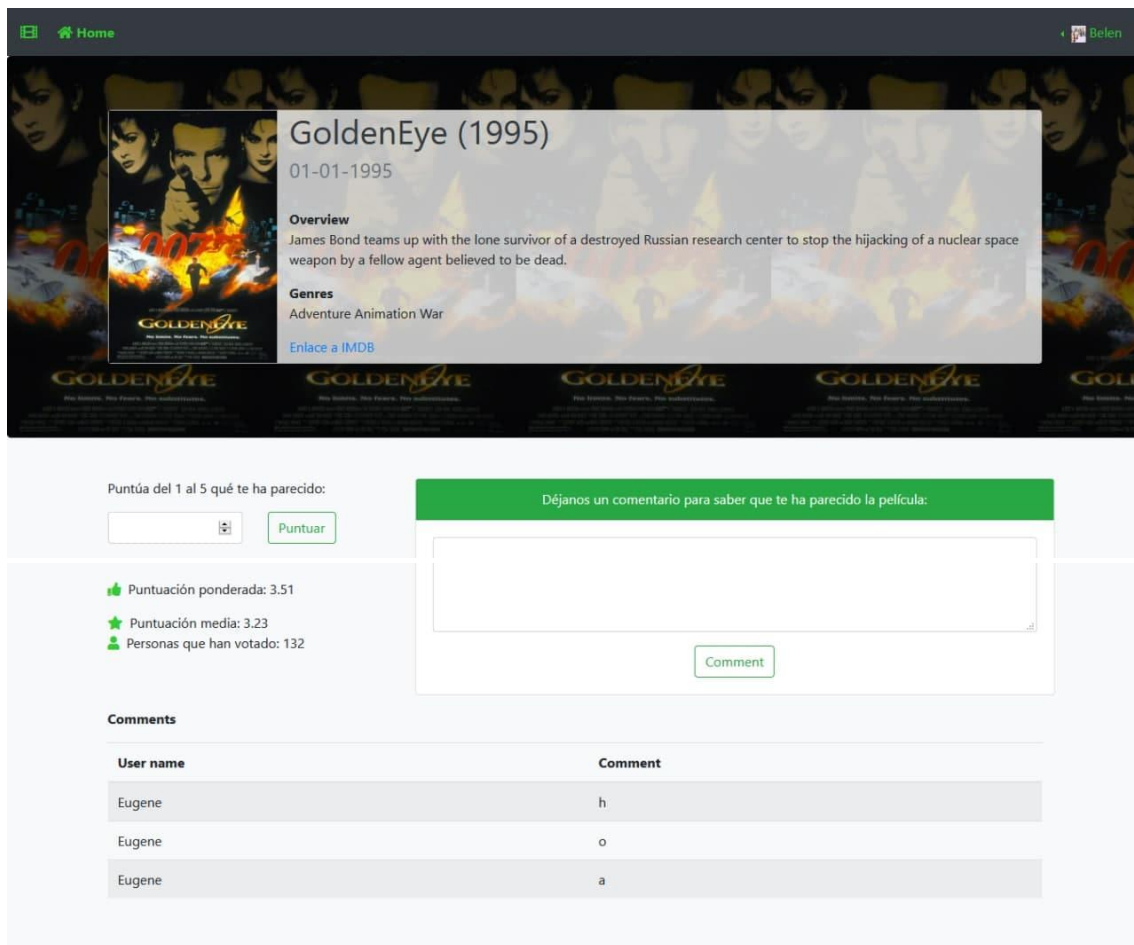


Figura 28: Página con la información de una película sin puntuar, logeado y con comentarios

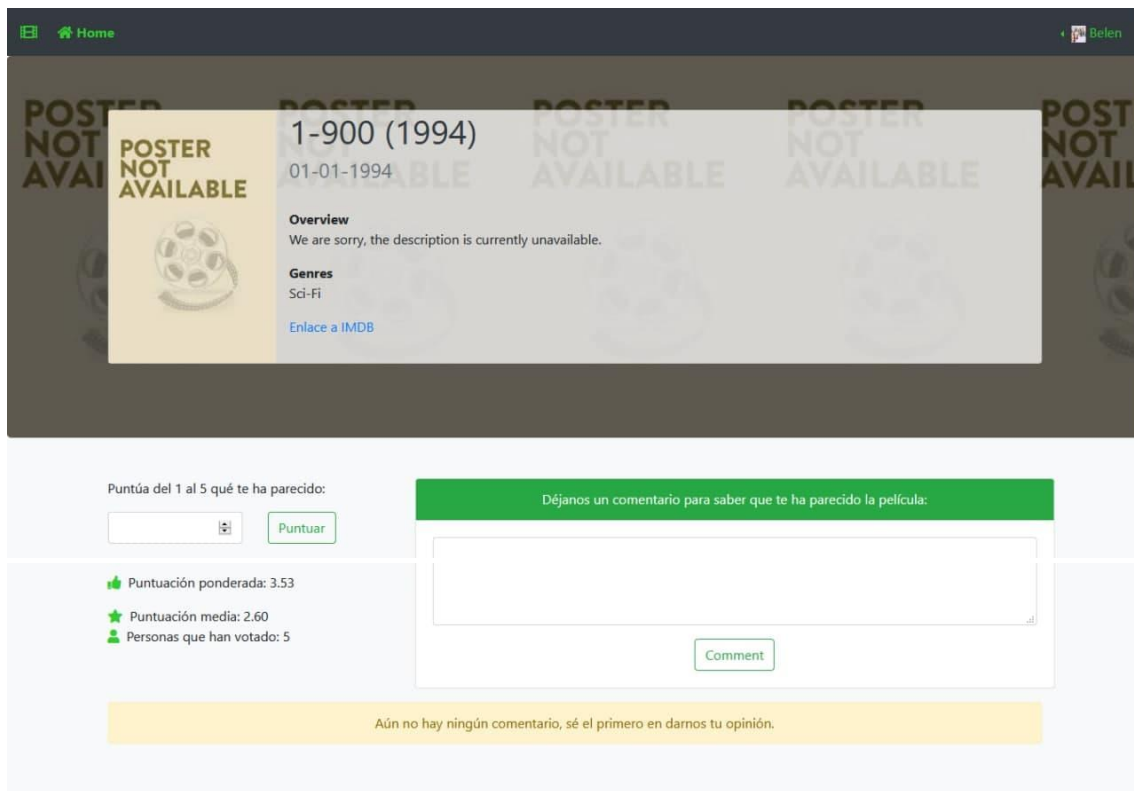


Figura 29: Página con la información de una película sin puntuar, logeado y sin comentarios

Para poder mostrar los formularios que permitan puntuar y comentar la película, se ha tenido que crear un archivo llamado `forms.py` en el que se encuentran todos los formularios. En éste se declaran los inputs del formulario con las características que se requieren y su tipo (tipo int, char...). En `views.py` se recoge el formulario y se pasa al HTML para que se muestre. Si el usuario ya ha puntuado la película, le aparece un mensaje con la puntuación que le ha dado y, además, también aparece el formulario rellenado con ese valor. Para volver a puntuar la película y darle un valor diferente, simplemente tiene que escribir el valor correspondiente y darle al botón de puntuar. Una vez que le da a puntuar, en `views.py` se comprueba si el usuario ya la había puntuado, dependiendo del resultado la consulta de la base de datos será de tipo INSERT o de tipo UPDATE, y vuelve a cargar la página de la película en la que ya le pone la puntuación actual del usuario. También se actualiza el número de personas que han puntuado la película y las diversas puntuaciones de ésta.

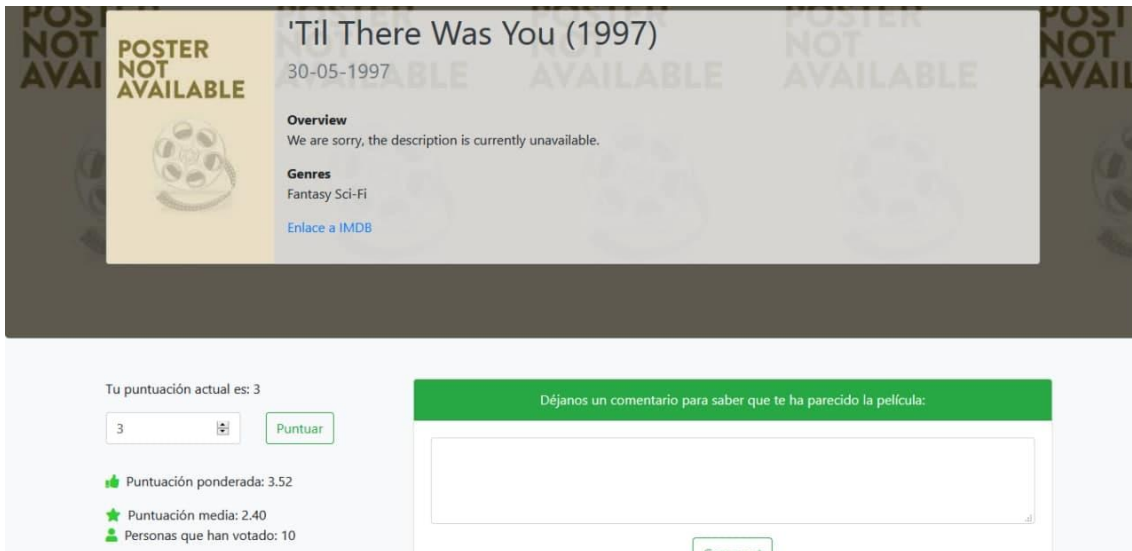


Figura 30: Página con la información de una película puntuada

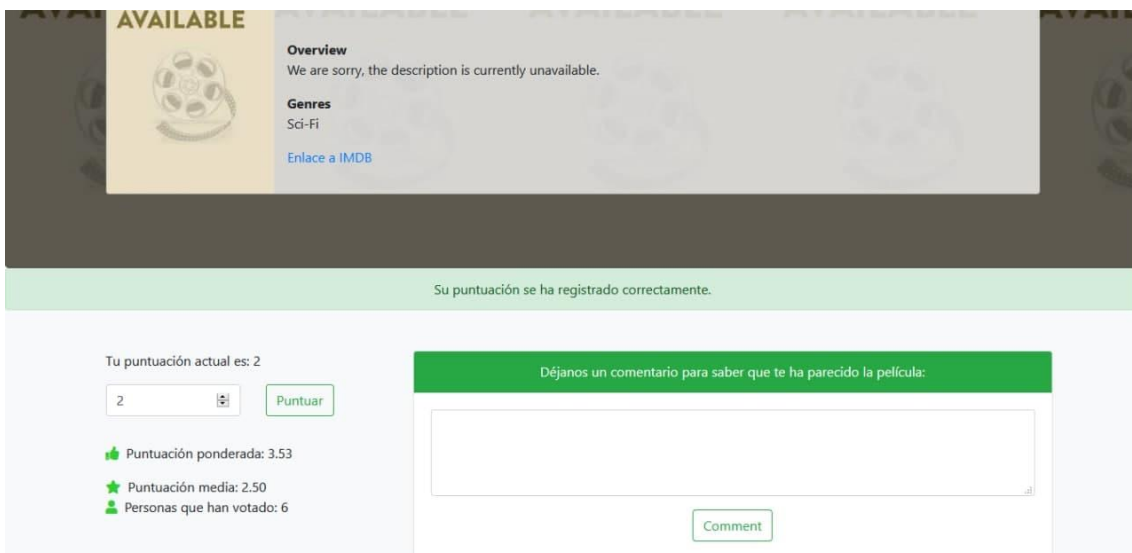


Figura 31: Página con la información de una película después de puntuarla

En el caso de que no se haya introducido un valor entre 1 y 5, vuelve a cargar la página con un mensaje indicando que no ha introducido un valor válido. Esto se detecta en la vista comprobando si el formulario ha sido válido, si no lo es, muestra el mensaje dicho anteriormente. Se sabe que no es válido ya que en forms.py se han usado unos validadores en el que se establecen que en ese input no se admite un valor que no esté entre 1 y 5 (ambos incluidos).

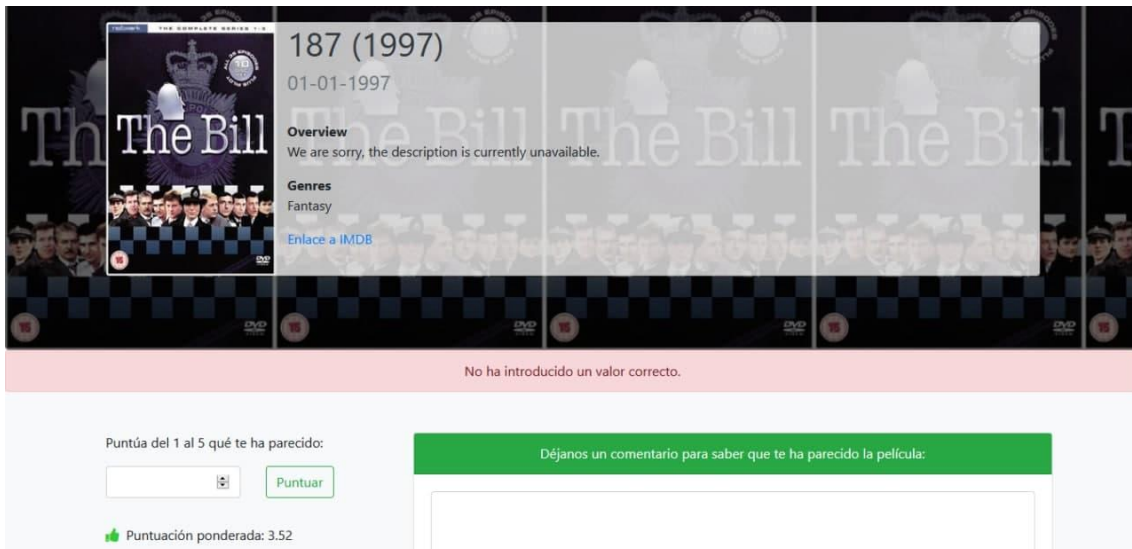


Figura 32: Página con la información de una película introduciendo un valor fuera del rango 1-5 en la puntuación

Respecto a los comentarios, cuando el usuario ha escrito su comentario y le da al botón para comentar, en la vista correspondiente de views.py, se inserta el comentario en la base de datos. Y vuelve a cargar la página con todos los comentarios de la película, incluyendo el que se acaba de añadir.

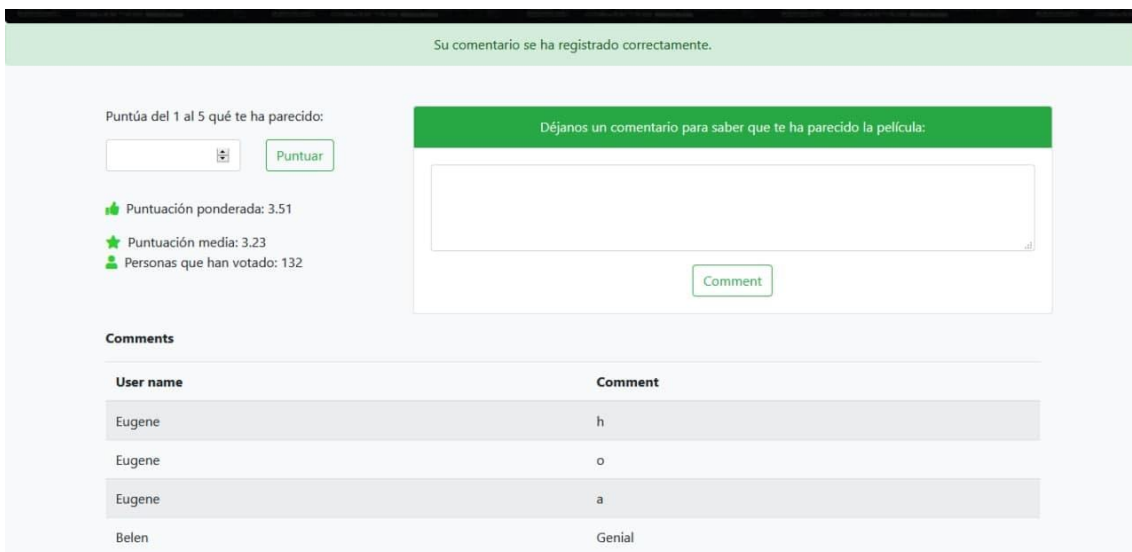


Figura 33 Página con la información de una película después de haberla comentado

Log in

En la parte superior de la página principal, hay una barra de navegación con dos botones, en uno de ellos pone Log in, éste te permite acceder a la página de inicio de sesión. En ella aparece un formulario para rellenar con un nombre de usuario y una contraseña. Cuando estás en esta página, en la barra de navegación desaparece la opción de Log in, quedándose únicamente, en la derecha, la opción de Sign up.

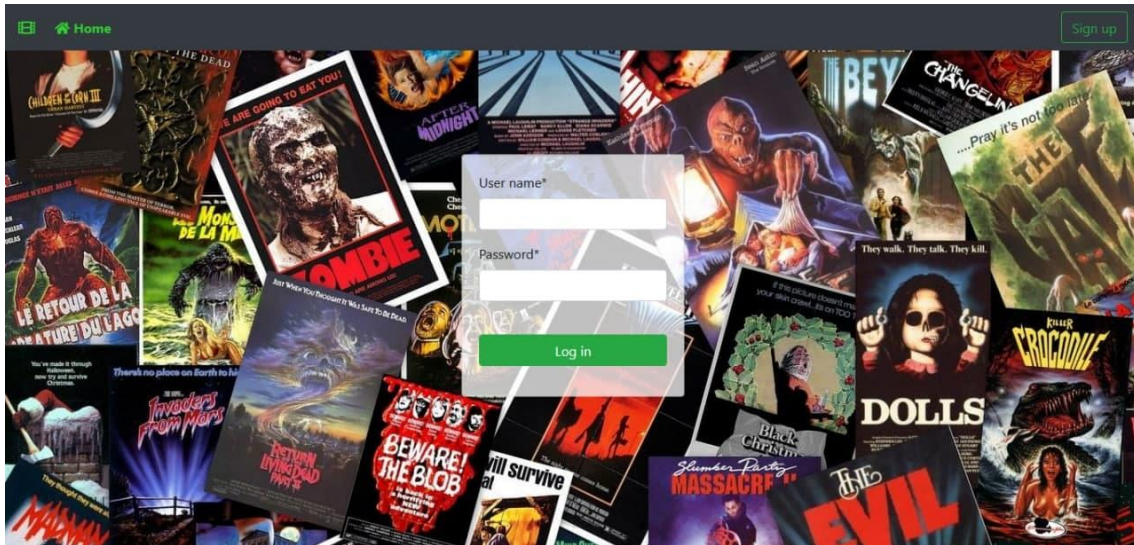


Figura 34: Página de inicio de sesión

Dicho formulario se recoge del archivo forms.py al igual que todos los demás. Cuando introduce el usuario sus datos, y son correctos, en views.py se comprueba que se ha hecho un request de tipo POST, se obtienen los datos de los campos y, si los datos son correctos, redirige a la página principal con un mensaje de que ha iniciado sesión correctamente.

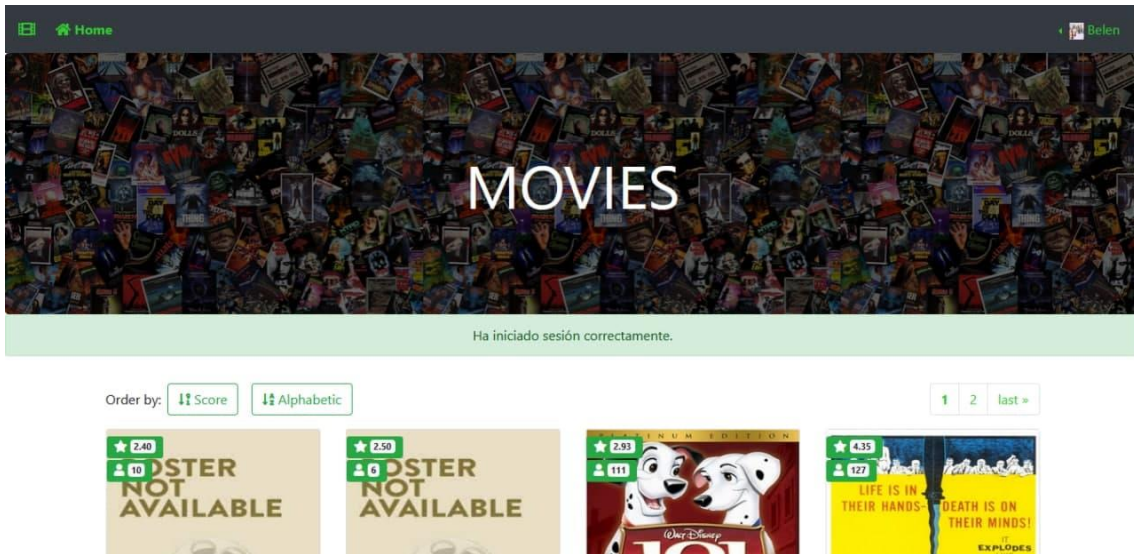


Figura 35: Página principal después de iniciar sesión correctamente

En caso de no haber introducido los datos correctamente, vuelve a cargar la página de log in con un mensaje de usuario o contraseña incorrectos.

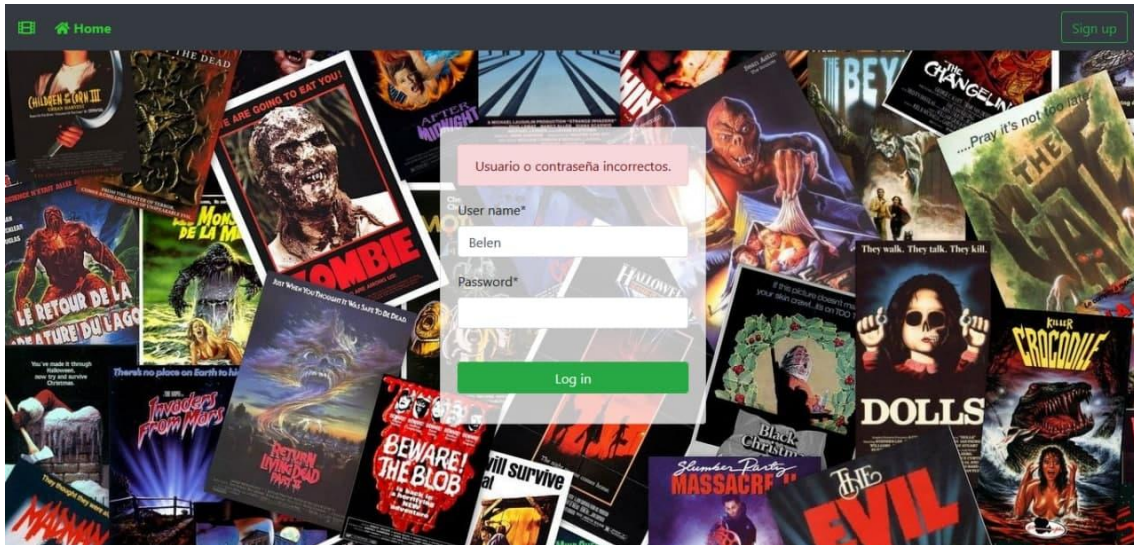


Figura 36: Página de inicio de sesión con credenciales incorrectos

En views.py, cuando se detecta un request de tipo POST, se sabe que se ha hecho un submit del formulario, por lo que lo primero que se comprueba es si el nombre de usuario introducido existe en la base de datos. Si el usuario no existe, vuelve a cargar log in con el mensaje citado anteriormente, si existe, se codifica el campo de contraseña mediante SHA1 y se compara con la de la base de datos correspondiente a ese nombre de usuario. Si los datos son correctos, se crea una sesión que permite al usuario navegar por la web como un usuario identificado.

Una vez que se ha iniciado sesión, en la barra de navegación superior (igual en todas las páginas de la web) los botones de Log in y Sign up son reemplazados por el nombre de usuario y su foto. A su vez, esto es un dropdown con las opciones de acceder a tu perfil, a la página de recomendaciones (si el usuario ha puntuado al menos una película), acceder a la página donde se muestran todos los usuarios a los que se les puede generar recomendaciones (si eres el usuario Admin) y la posibilidad de cerrar sesión. La opción de acceder a recomendaciones sólo aparece en caso de que el usuario logeado haya puntuado alguna película, ya que los algoritmos que se encargan de la generación de recomendaciones se basan en las puntuaciones de los usuarios. La opción de acceder a la página que permite generar las recomendaciones de otro usuario, que no es el identificado, solo le aparece al usuario Admin.



Figura 37: Barra de navegación con el usuario sin identificar



Figura 38: Barra de navegación con el usuario identificado sin haber puntuado ninguna película

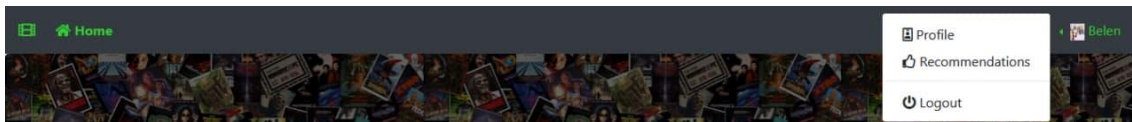


Figura 39: Barra de navegación con el usuario identificado y habiendo puntuado al menos una película

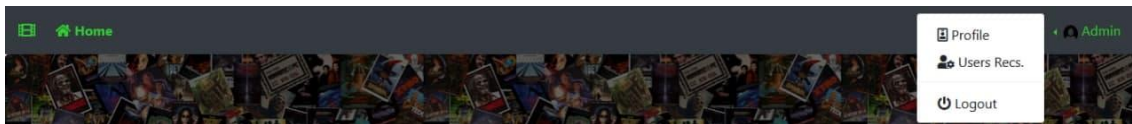


Figura 40: Barra de navegación con el usuario Admin identificado sin haber puntuado ninguna película

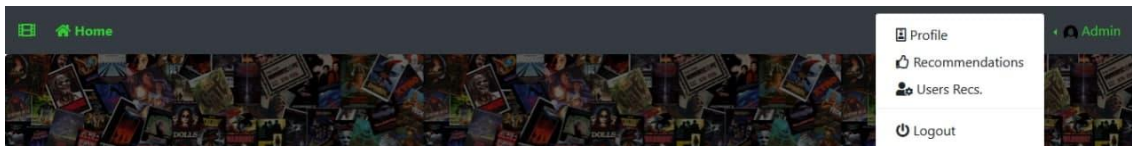


Figura 41: Barra de navegación con el usuario Admin identificado habiendo puntuado al menos una película

Sign up

Al igual que al iniciar sesión, para registrarse hay un botón que pone Sign up en la barra de navegación. Al cargar la página, desaparece dicho botón del navbar, quedando únicamente la opción de iniciar sesión. También hay un formulario en el que se deben rellenar los siguientes campos:

- Nombre.
- Edad.
- Sexo.
- Ocupación.
- Clave de acceso.
- Foto de perfil (opcional).

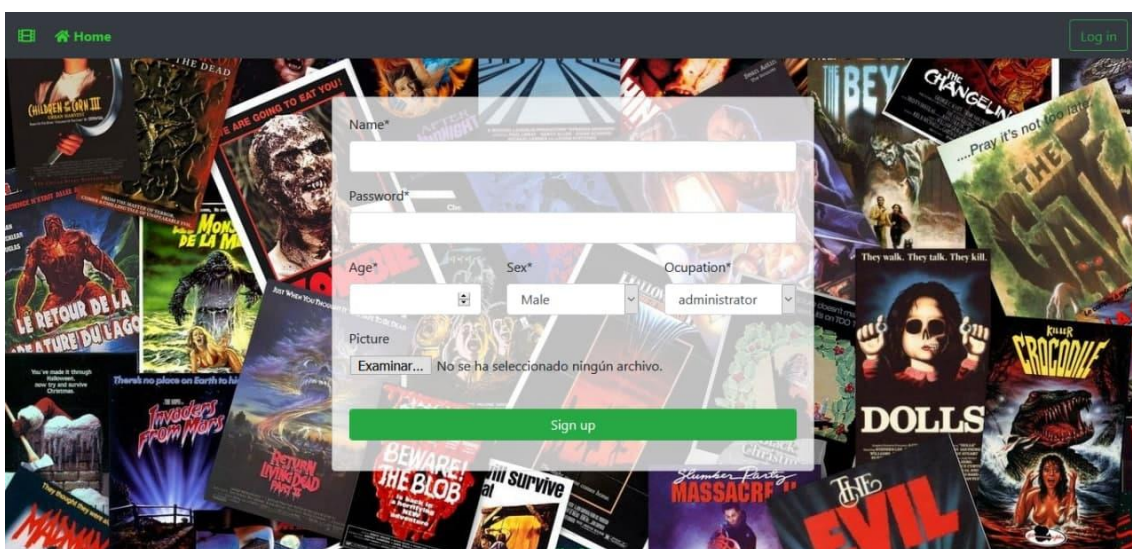


Figura 42: Página de registro

Se muestra el formulario al cargar la página y cuando el usuario ha rellenado todos los campos y le da a registrarse, en la vista de signup se observa si hay un request de tipo POST. Si lo hay, primero se comprueba que el nombre de usuario no se encuentre ya en la base de datos, si ya hay uno, se muestra un mensaje en el que se indica que ese nombre de usuario ya existe, y si no existe, se insertan en la base de datos todos los datos que ha rellenado el usuario. Una vez que el usuario se ha registrado, carga la página principal con un mensaje de que se ha realizado con éxito. Además, en la vista se crea una sesión cuando el registro ha sido exitoso para que el usuario ya aparezca logeado directamente.

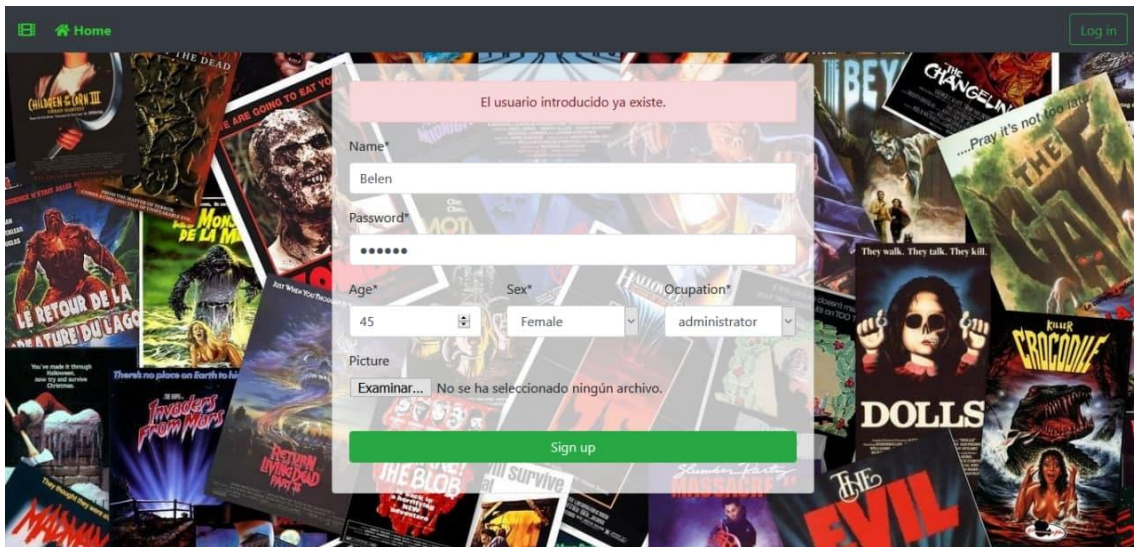


Figura 43: Página de registro introduciendo un nombre de usuario ya existente

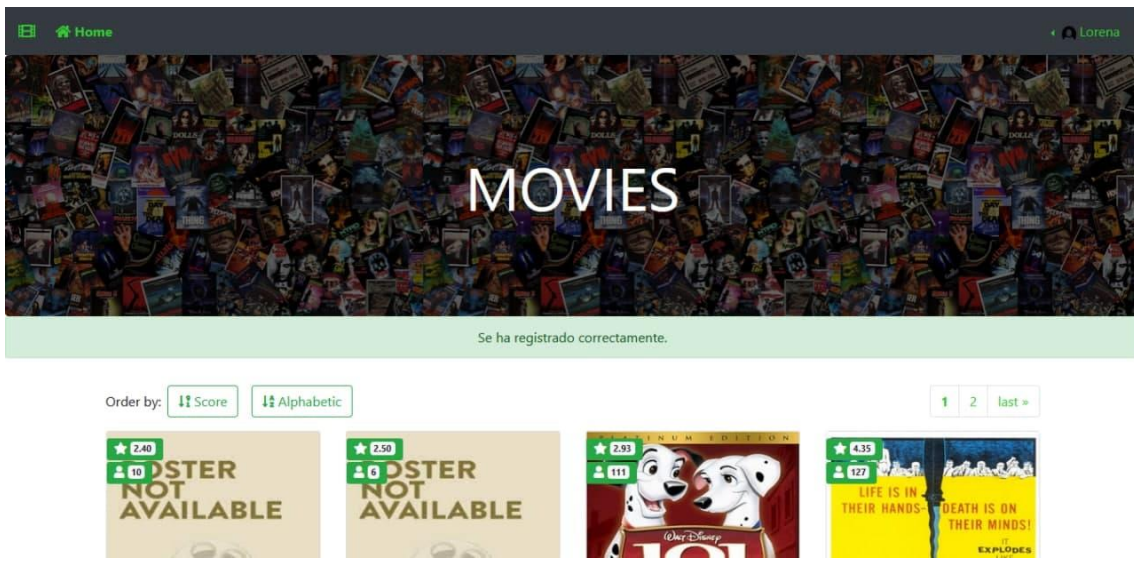


Figura 44: Página principal después de haber realizado el registro con éxito

En el caso de que no haya subido una imagen de usuario, ese campo quedará vacío en la base de datos (si el usuario que inicia la sesión no tiene una imagen, se muestra una por defecto) y si ha puesto una, se guarda el archivo en una carpeta y el nombre del archivo en la base de datos.

Profile

Se puede acceder al perfil de usuario a través de la opción Profile en el dropdown explicado anteriormente. Carga la misma página que en sign up con una pequeña diferencia, los campos de cada input ya están rellenos con sus datos (a excepción del input de la imagen).

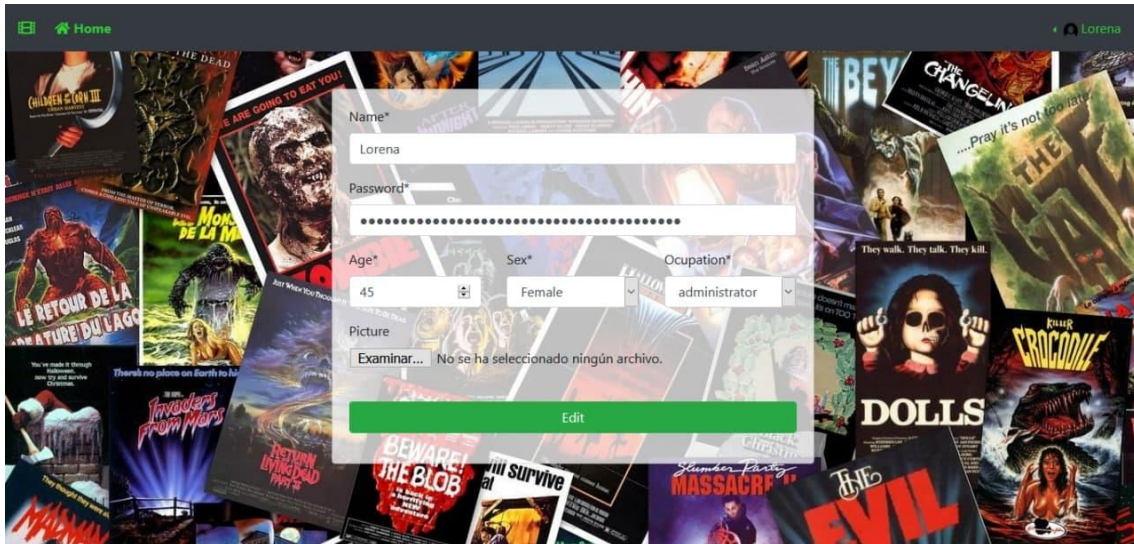


Figura 45: Página de perfil del usuario

Si el usuario quiere cambiar algo de su perfil, simplemente tiene que modificar el/los campo/campos que requiera y pulsar el botón de editar. En views.py se comprueba, cuando hay un request de tipo POST, cada campo del formulario con los datos que hay en la base de datos de dicho usuario para saber qué ha sido editado. A continuación, se actualiza la nueva información en la base de datos con lo que ha sido editado por el usuario y se carga la página principal con un mensaje de que se ha editado correctamente.

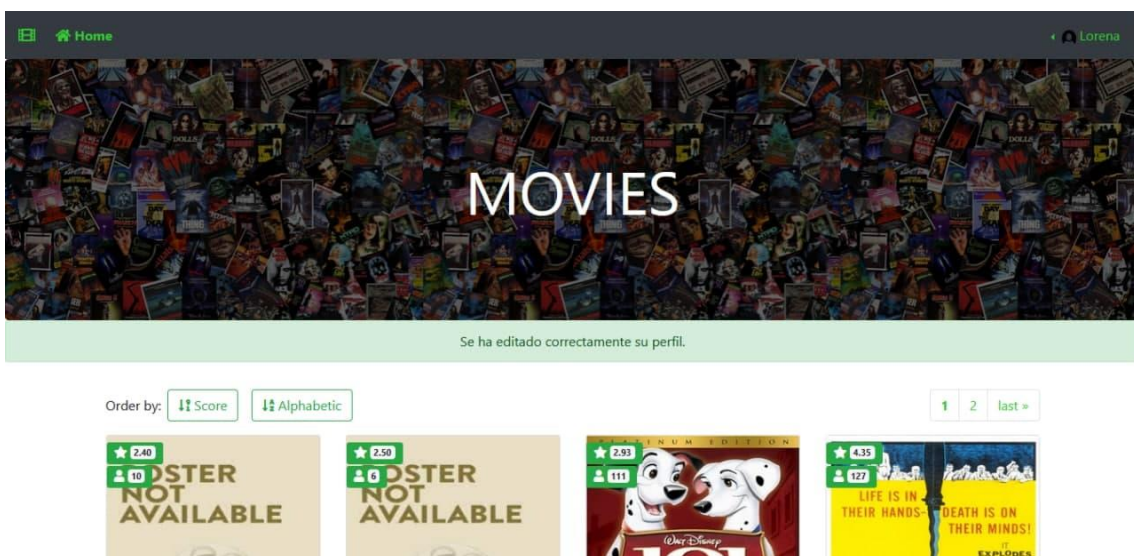


Figura 46: Página principal después de haber editado el perfil con éxito

En el caso de que el nombre de usuario sea editado, se comprueba si ese nombre ya existe para otro usuario. En caso afirmativo, vuelve a cargar profile con un mensaje advirtiéndole de ello.

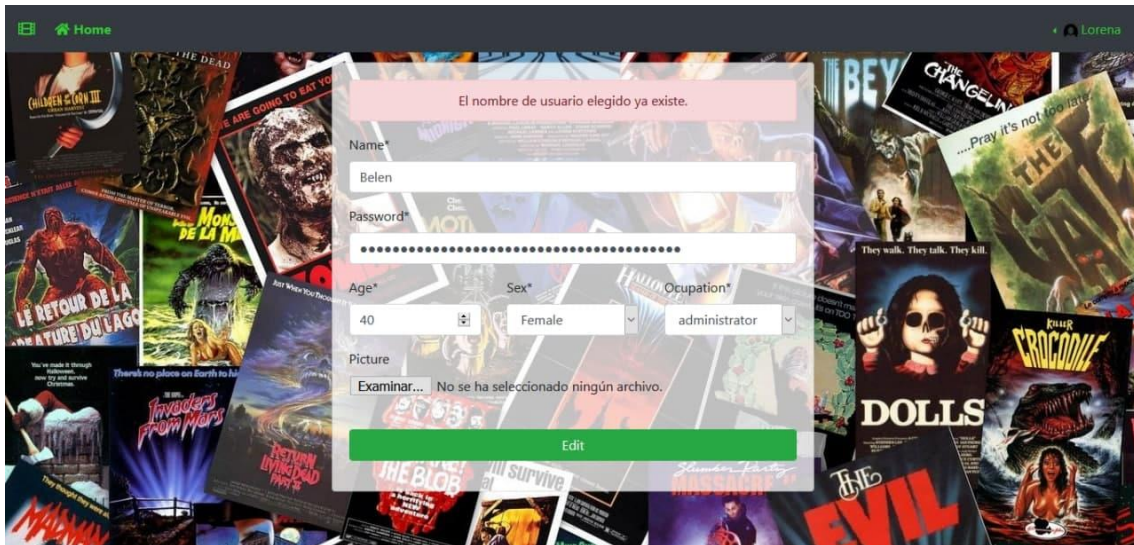


Figura 47: Página de perfil del usuario después de tratar de editar su nombre por uno ya existente

Log out

En views.py, la vista de logout comprueba si hay una sesión correspondiente al inicio de sesión de un usuario y si la hay, la elimina. El usuario cierra sesión clickando en Log out en el dropdown de la barra de navegación.

Cuando el usuario cierra sesión, la web carga la página principal con un mensaje advirtiéndole de que se ha realizado correctamente. Y la web se muestra de nuevo como cuando un usuario aún no ha iniciado sesión.

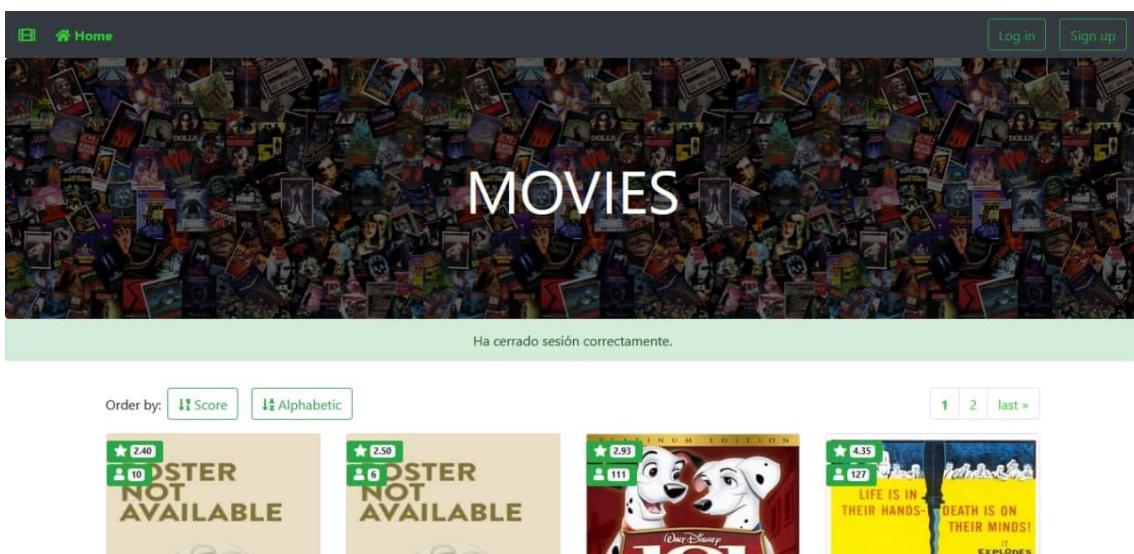


Figura 48: Página principal después de haber cerrado sesión

Generación de recomendaciones

Creación de algoritmo SVD sin bias

Para crear el algoritmo de filtrado colaborativo, el trabajo se divide en tres partes: conseguir los datos necesarios de la base de datos, crear el código de aprendizaje del algoritmo y guardar las predicciones en la base de datos.

La primera parte trata de generar las matrices R e Y y la lista movieList a partir de los datos de la base de datos. El número de filas de las matrices R e Y es el número de películas y el número de columnas es el número de usuarios. Los elementos de la matriz Y son las puntuaciones $y^{(i,j)}$ (de 1 a 5). La matriz R es una matriz indicadora binaria, donde $R(i,j) = 1$ si el usuario j puntuó la película i, y $R(i,j) = 0$ en caso contrario. El objetivo del filtrado colaborativo es predecir, para cada usuario, las puntuaciones de las películas que aún no ha puntuado, es decir, los valores de $y^{(i,j)}$, tales que $R(i,j) = 0$. Una vez realizada la predicción, le recomendamos a cada usuario las películas con mayores puntuaciones estimadas. Y movieList es una lista en la que se recogen todas las películas que hay en la base de datos.

Lo primero que se hace para el aprendizaje del algoritmo es conectar con la base de datos y recoger los parámetros que se necesitan, como número de películas y número de usuarios. También se obtienen los datos de la función getData(), que corresponde con la primera parte explicada anteriormente. A continuación, se guardan las puntuaciones realizadas por el usuario para poder predecir las puntuaciones del usuario en aquellas películas que no ha puntuado.

Para poder realizar las predicciones también se necesitan dos matrices X y Theta:

$$X = \begin{bmatrix} - & (x^{(1)})^T & - \\ \vdots & \vdots & \vdots \\ - & (x^{(n_m)})^T & - \end{bmatrix}, \quad \text{Theta} = \begin{bmatrix} - & (\theta^{(1)})^T & - \\ \vdots & \vdots & \vdots \\ - & (\theta^{(n_u)})^T & - \end{bmatrix}$$

La fila i-ésima de X contiene el vector características $x^{(i)}$ de la película i-ésima. La fila j-ésima de Theta contiene el vector de parámetros $\theta^{(j)}$ del usuario j.

Antes de poder predecir las puntuaciones, se optimiza la función de coste regularizada. Para realizar la optimización, se usa la función fmin_cg que minimiza una función usando un algoritmo de gradiente conjugado no lineal. Como argumentos se pasan la función de coste regularizado, unos parámetros que dependen de Theta y X, la función de gradiente regularizado y un número de iteraciones máximas de 100.

Para poder implementar la función de coste regularizada, primero se implementó la función de coste:

$$J(x^{(1)}, \dots, x^{n_m}, \theta^{(1)}, \dots, \theta^{n_u}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2$$

En la función sólo se acumula el coste para el usuario j y la película i si $R(i,j) = 1$. Una vez que se tenía la función de coste, había que regularizarla:

$$J(x^{(1)}, \dots, x^{n_m}, \theta^{(1)}, \dots, \theta^{n_u}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \left(\frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2 \right) + \left(\frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 \right)$$

Al igual que con la función de coste, para implementar el gradiente del filtrado colaborativo regularizado, primero se calcula sin regularizar:

$$\frac{\partial J}{\partial x_k^{(i)}} = \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)}$$

$$\frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)}$$

Ahora que ya lo tenemos, se regulariza:

$$\frac{\partial J}{\partial x_k^{(i)}} = \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)}$$

$$\frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)}$$

Finalmente, ahora que ya se tienen las predicciones, se llama a una función a la que se ha nombrado como `updateRecommendation()` y en la que se guardan o actualizan dichas predicciones de este usuario para cada una de las películas. Si es la primera vez que el usuario genera recomendaciones, estas predicciones se insertan en la base de datos, en el caso de que no sea la primera vez, simplemente se actualizan los datos.

Integración de algoritmos de surprise lib

Antes de todo, lo primero que se requiere es instalar `scikit-surprise` con la herramienta `pip`. Para que se pueda instalar con éxito, se necesita instalar Visual Studio, ya que incorpora código C++ que necesita ser compilado.

Esta librería proporciona una serie de algoritmos de predicción o recomendación, entre los cuales se ha hecho una selección para la comparación:

- **Algoritmo de descomposición en valores singulares (SVD)**

La descomposición singular de valores (Singular Value Decomposition) es una técnica de factorización de matrices que permite descomponer una matriz A en otras matrices U , S , V . Dentro del filtrado colaborativo, es necesario el manejo

de la matriz de votaciones de usuarios e ítems, por lo tanto, es posible considerar esta matriz como la base para aplicar SVD y obtener la factorización de matrices U , S y V .

- **Algoritmo Normal Predictor**

El algoritmo predice unas puntuaciones aleatorias basadas en una distribución del conjunto de formación, que se supone que es normal.

- **Algoritmo KNN Basic**

El algoritmo KNN es uno de los algoritmos de clasificación más simples y es uno de los algoritmos de aprendizaje más utilizados. Es un algoritmo de aprendizaje perezoso no paramétrico. Su propósito es utilizar una base de datos en la que los puntos de datos se separan en varias clases para predecir la clasificación de un nuevo punto de muestra.

- **Algoritmo Slope One**

El objetivo de los autores es que este algoritmo sea fácil de implementar, mantener y actualizar, eficiente al momento de recibir consultas de recomendaciones, capaz de recomendar a usuarios con pocos ratings y, lo más importante, que sea preciso en la recomendación. Consiste en calcular un promedio entre las diferencias de los ratings entre los ítems que el usuario (al cual se quiere predecir) ha calificado con el ítem cuyo rating se quiere predecir y hacer la predicción en base a esto.

- **Algoritmo CoClustering**

Consiste en agrupar una serie de vectores según un criterio en grupos o clusters. Generalmente el criterio suele ser la similitud por lo que diremos que agrupa los vectores similares en grupos. Está considerado como un aprendizaje no supervisado dentro de la minería de datos.

Una vez que ya se saben qué algoritmos comparar, se siguen una serie de pasos a la hora de integrarlos en el código. Lo primero es coger toda la tabla de puntuaciones de la base de datos y guardarla en un archivo llamado scores.csv para que de esta manera se pueda leer el archivo y poder usarlo en el algoritmo.

A continuación, se utiliza una función llamada `build_full_trainset()` en los datos recogidos para ajustar nuestro algoritmo a todo el conjunto de datos. Para entrenar el algoritmo, se utiliza la función `fit()` pasándole como argumento el resultado de la función anterior. Finalmente, para realizar la predicción, se necesita una lista de ratings que puede ser usada como testset (esta lista se consigue con ayuda de `build_anti_testset()`) para la función que se encarga de generar las predicciones (`test()`).

Una vez que ya están las predicciones hechas, se llama a una función nombrada como `get_top_n()` para que devuelva el id de las “n” películas mejor recomendadas según dichas predicciones y poder mostrar la información necesaria en la web. Este valor de “n” no varía en el proyecto, su valor siempre es 3, ya que solo se necesitan el top 3 películas.

Métricas de rendimiento

Las métricas de rendimiento que se van a mostrar en la web para cada uno de los algoritmos son:

- **Error absoluto medio o MAE (Mean absolute error)**

El error absoluto medio es la suma de los errores absolutos de todos los valores. Así que, si hay 5 valores en nuestro conjunto de datos, encontramos la diferencia entre el valor actual y el que se predijo para estos 5 valores y cogemos su valor positivo. Incluso si la diferencia entre el valor actual y el predicho es negativa, cogemos el valor positivo del cálculo. Una vez que tenemos todos estos valores positivos de todos los errores, se hace la media.

$$MAE = \sum_{(i,j):r(i,j)=1} \frac{|y^{(i,j)} - \hat{y}^{(i,j)}|}{C}$$

- **Error cuadrático medio o MSE (Mean square error)**

El error cuadrático medio siempre es positivo y es mejor que su valor sea lo más cercano a 0 posible. Se calcula de manera muy parecida al error absoluto medio, después de calcular la diferencia para cada uno de los valores, se eleva a 2. Para una mejor comprensión, muestro la ecuación:

$$MSE = \sum_{(i,j):r(i,j)=1} \frac{(y^{(i,j)} - \hat{y}^{(i,j)})^2}{C}$$

- **Raíz del error cuadrático medio o RMSE (Root mean square error)**

Este error es exactamente igual al error cuadrático medio MSE, a excepción de que habría que añadirle la de raíz cuadrada al resultado obtenido.

$$RMSE = \sqrt{\sum_{(i,j):r(i,j)=1} \frac{(y^{(i,j)} - \hat{y}^{(i,j)})^2}{C}}$$

El módulo `surprise.accuracy` proporciona las herramientas para el cómputo de las métricas de precisión en un conjunto de predicciones.

Comparación de algoritmos

A continuación, se van a realizar una serie de comparaciones de algoritmos e interpretaremos su calidad con ayuda de las métricas de rendimiento descritas anteriormente. Para poder realizar estas comparaciones de una manera más fluida, se ha usado un usuario Admin que tiene acceso a la generación de recomendaciones de cualquier usuario. Además, también puede cambiar ciertos parámetros de los algoritmos para cada uno de los usuarios.

Antes de proceder con las comparaciones, cabe destacar que tenemos que interpretar RMSE, MSE y MAE. Por lo que basándonos en lo explicado anteriormente sobre estas métricas sabemos que MAE coge directamente la diferencia entre la predicción y lo real sin considerar si la diferencia es más hacia un lado que hacia el otro.

MSE no es fácil de interpretar ya que no se encuentra en la misma unidad que nuestros datos, por lo que lo único que se sabe es que cuanto más cerca de 0 sea su valor, mejor calidad. Pero a diferencia de MAE sí que tiene en cuenta si la diferencia entre la predicción y lo real oscila más hacia un lado o hacia el otro.

Y, por último, RMSE es igual que MSE con la pequeña diferencia de que si se encuentra en la misma unidad que nuestros datos y podemos interpretarla mejor. Por lo que a la hora de poder comparar los algoritmos nos vamos a centrar más en el valor de RMSE. También hay que tener en cuenta que nuestros datos son de 1 a 5, una variación de 3 es mucho en este caso.

Recomendaciones de todos los algoritmos sin variaciones

Ahora que ya se sabe cómo interpretar la calidad de estas recomendaciones, primero vamos a comprobar cómo varía la calidad de las predicciones ejecutando el algoritmo dos veces para cada uno de los usuarios:

- Usuario Scott con 737 películas puntuadas.

Scott (737)	RMSE		MSE		MAE	
	1ª Iter	2ª Iter	1ª Iter	2ª Iter	1ª Iter	2ª Iter
SVD sin bias	1.626601	1.516448	2.645830	2.299615	1.421605	1.363153
SVD	0.606528	0.604538	0.367876	0.365466	0.476374	0.474981
NormalPredictor	1.021902	1.022195	1.044284	1.044883	0.843025	0.843173
KNNBasic	0.916161	0.916161	0.839351	0.839351	0.689743	0.689743
SlopeOne	0.940528	0.940528	0.884592	0.884592	0.741831	0.741831
CoClustering	0.993645	0.988719	0.987330	0.977566	0.773711	0.768185

Tabla 1: Recomendaciones de todos los algoritmos para Scott (2 iteraciones).

- Usuario Adrian con 405 películas puntuadas.

Adrian (405)	RMSE		MSE		MAE	
	1ª Iter	2ª Iter	1ª Iter	2ª Iter	1ª Iter	2ª Iter
SVD sin bias	2.541861	2.570588	6.461059	6.607924	2.245209	2.265107
SVD	0.607020	0.605986	0.368473	0.367219	0.476722	0.476354
NormalPredictor	1.022265	1.021737	1.045025	1.043946	0.843355	0.842851
KNNBasic	0.916161	0.916161	0.839351	0.839351	0.689743	0.689743
SlopeOne	0.940528	0.940528	0.884592	0.884592	0.741831	0.741831
CoClustering	0.986715	0.996036	0.973607	0.992088	0.765711	0.775134

Tabla 2: Recomendaciones de todos los algoritmos para Adrian (2 iteraciones).

- Usuario Jeremy con 206 películas puntuadas.

Jeremy (206)	RMSE		MSE		MAE	
	1ª Iter	2ª Iter	1ª Iter	2ª Iter	1ª Iter	2ª Iter
SVD sin bias	2.582936	2.579090	6.671558	6.651703	2.363932	2.359577
SVD	0.607036	0.605460	0.368493	0.366581	0.476910	0.475531
NormalPredictor	1.021325	1.022311	1.043105	1.045119	0.842388	0.843436
KNNBasic	0.916161	0.916161	0.839351	0.839351	0.689743	0.689743
SlopeOne	0.940528	0.940528	0.884592	0.884592	0.741831	0.741831
CoClustering	0.995984	1.003976	0.991985	1.007967	0.775217	0.783160

Tabla 3: Recomendaciones de todos los algoritmos para Jeremy (2 iteraciones).

- Usuario Olivar con 93 películas puntuadas.

Olivar (93)	RMSE		MSE		MAE	
	1ª Iter	2ª Iter	1ª Iter	2ª Iter	1ª Iter	2ª Iter
SVD sin bias	2.591376	2.605142	6.715228	6.786765	2.414679	2.428749
SVD	0.606124	0.606870	0.367386	0.368292	0.476185	0.476726
NormalPredictor	1.021061	1.021328	1.042566	1.043112	0.842599	0.842287
KNNBasic	0.916161	0.916161	0.839351	0.839351	0.689743	0.689743
SlopeOne	0.940528	0.940528	0.884592	0.884592	0.741831	0.741831
CoClustering	0.990951	0.988873	0.981984	0.977869	0.771157	0.768644

Tabla 4: Recomendaciones de todos los algoritmos para Olivar (2 iteraciones).

- Usuario Arias con 20 películas puntuadas.

Arias (20)	RMSE		MSE		MAE	
	1ª Iter	2ª Iter	1ª Iter	2ª Iter	1ª Iter	2ª Iter
SVD sin bias	1.487936	1.490616	2.213954	2.221935	1.390954	1.393222
SVD	0.605750	0.607304	0.366933	0.368818	0.475629	0.477186
NormalPredictor	1.022107	1.021444	1.044703	1.043348	0.843137	0.842704
KNNBasic	0.916161	0.916161	0.839351	0.839351	0.689743	0.689743
SlopeOne	0.940528	0.940528	0.884592	0.884592	0.741831	0.741831
CoClustering	0.993950	1.006783	0.987936	1.013612	0.773602	0.783665

Tabla 5: Recomendaciones de todos los algoritmos para Arias (2 iteraciones).

- Usuario Pepe con 1 película puntuada.

Pepe (1)	RMSE		MSE		MAE	
	1ª Iter	2ª Iter	1ª Iter	2ª Iter	1ª Iter	2ª Iter
SVD sin bias	0.261120	0.262016	0.068184	0.068653	0.245455	0.246233
SVD	0.606632	0.608198	0.368003	0.369904	0.476511	0.477697
NormalPredictor	1.022162	1.020453	1.044816	1.041324	0.843124	0.841737
KNNBasic	0.916161	0.916161	0.839351	0.839351	0.689743	0.689743
SlopeOne	0.940528	0.940528	0.884592	0.884592	0.741831	0.741831
CoClustering	0.986100	0.995030	0.972393	0.990085	0.765979	0.774447

Tabla 6: Recomendaciones de todos los algoritmos para Pepe (2 iteraciones).

Podemos ver en todos los usuarios que entre una iteración y otra apenas hay diferencia, la calidad de la predicción podría decirse que es la misma. Por lo que no es necesario ejecutar un algoritmo varias veces hasta que muestre unos valores que signifiquen una mejor calidad de predicción.

Ahora vamos a ver con qué calidad recomiendan cada uno de los algoritmos para cada usuario. El algoritmo de filtrado colaborativo implementado desde cero es el que muestra unos valores de RMSE más altos. También se aprecia que su calidad mejora cuando se trata de un usuario con muchas películas puntuadas y un usuario con muy pocas películas puntuadas.

El algoritmo SVD es el que muestra unos valores más bajos de RMSE, por lo que es el que parece más fiable a la hora de generar recomendaciones al usuario. Si hacemos la media de RMSE de las dos iteraciones para cada uno de los usuarios tendremos los siguientes valores:

SVD	Scott	Adrian	Jeremy	Olivar	Arias	Pepe
RMSE	0,605533	0,606503	0,606248	0,606497	0,606527	0,607415

Tabla 7: Media de iteraciones para SVD.

Al realizar esta media se puede apreciar un poco mejor la tendencia que tiene el algoritmo a aumentar el valor de RMSE cuantas menos películas puntuadas tenga el usuario. Por lo que deducimos que el algoritmo genera unas recomendaciones de mejor calidad cuando puede coger más datos de referencia, que en este caso son películas puntuadas.

Las recomendaciones basadas en NormalPredictor, dentro de la librería surprise, es el que peor calidad presenta en sus predicciones encontrando unos valores de RMSE en torno a 1.02. Mirando las tablas anteriores no se aprecia demasiado si a este algoritmo le afecta en mayor o menor medida el número de películas puntuadas, por lo que vamos a hacer una tabla cogiendo la media entre las dos iteraciones para cada uno de los usuarios (al igual que se ha hecho para el algoritmo SVD).

NP	Scott	Adrian	Jeremy	Olivar	Arias	Pepe
RMSE	1,022049	1,022001	1,021818	1,021195	1,021776	1,021308

Tabla 8: Media de iteraciones para NormalPredictor.

Ahora sí podríamos decir que este algoritmo intenta generar mejores recomendaciones cuando tiene pocos datos en los que basarse, ya que se ve una ligera tendencia a disminuir el valor de RMSE cuanto menor es el número de películas puntuadas.

El algoritmo KNNBasic muestra exactamente los mismos valores para todas las predicciones, además es una de las mejores calidades entre los diferentes algoritmos. Entonces, sabemos que a este algoritmo no le afecta en lo más mínimo el número de películas puntuadas y la calidad de sus recomendaciones es siempre la misma.

Con el algoritmo SlopeOne encontramos el mismo caso que con el explicado anteriormente. Su calidad de recomendación es la misma para todos los usuarios independientemente del número de películas puntuadas. Aunque sí se puede apreciar que, si hubiera que elegir entre uno de los dos, KNNBasic sería una mejor opción.

Por último, está el algoritmo CoClustering que parece que es el segundo peor en cuanto a la calidad de sus recomendaciones respecta. Para poder apreciar mejor si dicha calidad varía respecto al número de películas puntuadas, vamos a realizar una tabla con la media de las dos iteraciones:

CC	Scott	Adrian	Jeremy	Olivar	Arias	Pepe
RMSE	0,991182	0,991376	0,999980	0,989912	1,000367	0,990565

Tabla 9: Media de iteraciones para CoClustering.

Observando la tabla se podría decir que indistintamente, ofrece una calidad similar para cada uno de los casos. No se aprecia ninguna tendencia que mejore o empeore sus recomendaciones.

Después de haber comparado estos algoritmos, el que ofrece unas mejores predicciones al usuario, tanto para aquellos que han puntuado más películas como para los que menos, es el algoritmo SVD. Esto puede ser debido a que se trata de un algoritmo más complejo basado en la factorización de matrices.

Variaciones en KNNBasic

Ahora vamos a observar cómo actúa el algoritmo KNNBasic cambiando dos parámetros, k y min_k . El parámetro k es el número (máximo) de vecinos a tener en cuenta en la agregación y el parámetro min_k el número mínimo de vecinos. Si no hay suficientes vecinos en min_k , la predicción se establece a la media global de todas las puntuaciones. Las recomendaciones vistas anteriormente sobre este algoritmo se establecieron con los valores por defecto de estos parámetros, $k = 40$ y $\text{min_k} = 1$.

Primero, se ha probado a mantener el valor por defecto de k y aumentar min_k , acotando un margen menor de número de vecinos a tener en cuenta en la agregación.

- k = 40 y min_k = 10

KNNBasic	RMSE	MSE	MAE
Scott (737)	0.526384	0.277080	0.340901
Adrian (405)	0.526384	0.277080	0.340901
Jeremy (206)	0.526384	0.277080	0.340901
Olivar (93)	0.526384	0.277080	0.340901
Arias (20)	0.526384	0.277080	0.340901
Pepe (1)	0.526384	0.277080	0.340901

Tabla 10: KNNBasic con k = 40 y min_k = 10.

- k = 40 y min_k = 20

KNNBasic	RMSE	MSE	MAE
Scott (737)	0.422516	0.178520	0.248186
Adrian (405)	0.422516	0.178520	0.248186
Jeremy (206)	0.422516	0.178520	0.248186
Olivar (93)	0.422516	0.178520	0.248186
Arias (20)	0.422516	0.178520	0.248186
Pepe (1)	0.422516	0.178520	0.248186

Tabla 11: KNNBasic con k = 40 y min_k = 20.

- k = 40 y min_k = 40

KNNBasic	RMSE	MSE	MAE
Scott (737)	0.324664	0.105407	0.163705
Adrian (405)	0.324664	0.105407	0.163705
Jeremy (206)	0.324664	0.105407	0.163705
Olivar (93)	0.324664	0.105407	0.163705
Arias (20)	0.324664	0.105407	0.163705
Pepe (1)	0.324664	0.105407	0.163705

Tabla 12: KNNBasic con k = 40 y min_k = 40.

A continuación, se tomado la decisión de aumentar ambos valores. Ya que se sabe que aumentando el valor de min_k parece que la recomendación es de mejor calidad.

- k = 150 y min_k = 100

KNNBasic	RMSE	MSE	MAE
Scott (737)	0.183374	0.033626	0.061499
Adrian (405)	0.183374	0.033626	0.061499
Jeremy (206)	0.183374	0.033626	0.061499
Olivar (93)	0.183374	0.033626	0.061499
Arias (20)	0.183374	0.033626	0.061499
Pepe (1)	0.183374	0.033626	0.061499

Tabla 13: KNNBasic con k = 150 y min_k = 100.

- $k = 150$ y $\text{min}_k = 150$

KNNBasic	RMSE	MSE	MAE
Scott (737)	0.136220	0.018556	0.034396
Adrian (405)	0.136220	0.018556	0.034396
Jeremy (206)	0.136220	0.018556	0.034396
Olivar (93)	0.136220	0.018556	0.034396
Arias (20)	0.136220	0.018556	0.034396
Pepe (1)	0.136220	0.018556	0.034396

Tabla 14: KNNBasic con $k = 150$ y $\text{min}_k = 150$.

- $k = 3000$ y $\text{min}_k = 3000$

KNNBasic	RMSE	MSE	MAE
Scott (737)	0	0	0
Adrian (405)	0	0	0
Jeremy (206)	0	0	0
Olivar (93)	0	0	0
Arias (20)	0	0	0
Pepe (1)	0	0	0

Tabla 15: KNNBasic con $k = 3000$ y $\text{min}_k = 3000$.

Ya hemos visto cómo afecta a las recomendaciones aumentar ambos parámetros, ahora veamos qué resultados se obtienen si mantenemos min_k con su valor por defecto y se disminuye el valor de k con respecto al que tiene por defecto.

- $k = 30$ y $\text{min}_k = 1$

KNNBasic	RMSE	MSE	MAE
Scott (737)	0.918405	0.843467	0.692431
Adrian (405)	0.918405	0.843467	0.692431
Jeremy (206)	0.918405	0.843467	0.692431
Olivar (93)	0.918405	0.843467	0.692431
Arias (20)	0.918405	0.843467	0.692431
Pepe (1)	0.918405	0.843467	0.692431

Tabla 16: KNNBasic con $k = 30$ y $\text{min}_k = 1$.

- $k = 20$ y $\text{min}_k = 1$

KNNBasic	RMSE	MSE	MAE
Scott (737)	0.922792	0.851545	0.697369
Adrian (405)	0.922792	0.851545	0.697369
Jeremy (206)	0.922792	0.851545	0.697369
Olivar (93)	0.922792	0.851545	0.697369
Arias (20)	0.922792	0.851545	0.697369
Pepe (1)	0.922792	0.851545	0.697369

Tabla 17: KNNBasic con $k = 20$ y $\text{min}_k = 1$.

- $k = 1$ y $\text{min}_k = 1$

KNNBasic	RMSE	MSE	MAE
Scott (737)	1.270217	1.613450	1.049162
Adrian (405)	1.270217	1.613450	1.049162
Jeremy (206)	1.270217	1.613450	1.049162
Olivar (93)	1.270217	1.613450	1.049162
Arias (20)	1.270217	1.613450	1.049162
Pepe (1)	1.270217	1.613450	1.049162

Tabla 18: KNNBasic con $k = 1$ y $\text{min}_k = 1$.

Después de realizar una serie de pruebas, podemos ver que cuanto mayor sean los valores de ambos parámetros, menores valores de RMSE, MSE y MAE obtenemos. Por lo que se podría decir que habría que pasar valores grandes hasta que RMSE sea 0 y obtener la mejor predicción. Pero hay que tener en cuenta, por ejemplo, cuando los valores de k y min_k son 3000, que no hay tantos vecinos a tener en cuenta.

En nuestro caso, los vecinos que se buscan para correlacionar similitudes son los usuarios que ya han puntuado películas, y en la base de datos hay alrededor de unos 960 usuarios. Además, dentro de esos 960 usuarios, el algoritmo debe buscar aquellos cuya medida de similitud sea positiva. Por lo que un valor igual al número de usuarios tampoco sería una buena recomendación.

Teniendo en cuenta todo lo dicho anteriormente, se podría decir que sí puede ser una buena recomendación cuando ambos parámetros tenían como valor 150. Ya que de entre todos los usuarios que hay en la base de datos, sí que es posible que el algoritmo haya encontrado ese número de vecinos con la medida de similitud positiva.

Variaciones en CoClustering

El siguiente algoritmo al que se le va a hacer una serie de pruebas con diferentes valores para sus parámetros es CoClustering. Los parámetros que van a ser modificados son n_{cltr_u} , n_{cltr_i} y n_{epochs} . El parámetro n_{cltr_u} es el número de clusters de usuarios, n_{cltr_i} es el número de clusters de ítems y n_{epochs} es el número de iteraciones del ciclo de optimización. Sus valores por defecto son: $n_{\text{cltr}_u} = 3$, $n_{\text{cltr}_i} = 3$ y $n_{\text{epochs}} = 20$.

Las primeras pruebas se han realizado, aumentando y disminuyendo los valores de los parámetros correspondientes a los clusters de usuarios e ítems manteniendo el número de iteraciones con su valor por defecto.

- $n_cltr_u = 20, n_cltr_i = 100$ y $n_epochs = 20$.

CoClustering	RMSE	MSE	MAE
Scott (737)	1.077899	1.161865	0.848838
Adrian (405)	1.070604	1.146193	0.843252
Jeremy (206)	1.060618	1.124910	0.837206
Olivar (93)	1.082735	1.172315	0.850674
Arias (20)	1.082121	1.170985	0.854137
Pepe (1)	1.089482	1.186970	0.856820

Tabla 19: CoClustering con $n_cltr_u = 20, n_cltr_i = 100$ y $n_epochs = 20$.

- $n_cltr_u = 5, n_cltr_i = 6$ y $n_epochs = 20$.

CoClustering	RMSE	MSE	MAE
Scott (737)	0.996194	0.992402	0.775281
Adrian (405)	1.016193	1.032647	0.793513
Jeremy (206)	1.016570	1.033415	0.794600
Olivar (93)	0.996045	0.992106	0.775722
Arias (20)	1.009167	1.018419	0.787880
Pepe (1)	0.997815	0.995634	0.777328

Tabla 20: CoClustering con $n_cltr_u = 5, n_cltr_i = 6$ y $n_epochs = 20$.

- $n_cltr_u = 5, n_cltr_i = 5$ y $n_epochs = 20$.

CoClustering	RMSE	MSE	MAE
Scott (737)	1.004254	1.008526	0.784018
Adrian (405)	1.002920	1.005849	0.781733
Jeremy (206)	1.011993	1.024130	0.789709
Olivar (93)	1.008693	1.017461	0.783971
Arias (20)	1.007643	1.015345	0.785996
Pepe (1)	1.004840	1.009704	0.783503

Tabla 21: CoClustering con $n_cltr_u = 5, n_cltr_i = 5$ y $n_epochs = 20$.

- $n_cltr_u = 2, n_cltr_i = 3$ y $n_epochs = 20$.

CoClustering	RMSE	MSE	MAE
Scott (737)	0.999327	0.998654	0.778068
Adrian (405)	0.990520	0.981130	0.769067
Jeremy (206)	0.992898	0.985846	0.772014
Olivar (93)	0.998277	0.996557	0.778502
Arias (20)	0.991642	0.983353	0.770128
Pepe (1)	0.994295	0.988622	0.770964

Tabla 22: CoClustering con $n_cltr_u = 2, n_cltr_i = 3$ y $n_epochs = 20$.

- $n_cltr_u = 2, n_cltr_i = 2$ y $n_epochs = 20$.

CoClustering	RMSE	MSE	MAE
Scott (737)	0.984893	0.970014	0.764834
Adrian (405)	0.973566	0.947831	0.755399
Jeremy (206)	0.987493	0.975143	0.768945
Olivar (93)	0.987708	0.975567	0.768104
Arias (20)	0.985497	0.971204	0.765630
Pepe (1)	0.989195	0.978507	0.769188

Tabla 23: CoClustering con $n_cltr_u = 2, n_cltr_i = 2$ y $n_epochs = 20$.

- $n_cltr_u = 1, n_cltr_i = 1$ y $n_epochs = 20$.

CoClustering	RMSE	MSE	MAE
Scott (737)	0.969275	0.939495	0.750815
Adrian (405)	0.969275	0.939495	0.750815
Jeremy (206)	0.969275	0.939495	0.750815
Olivar (93)	0.969275	0.939495	0.750815
Arias (20)	0.969275	0.939495	0.750815
Pepe (1)	0.969275	0.939495	0.750815

Tabla 24: CoClustering con $n_cltr_u = 1, n_cltr_i = 1$ y $n_epochs = 20$.

A continuación, se han realizado otras pruebas manteniendo los valores de los clusters de usuarios e ítems a su valor por defecto y se ha ido variando el valor del número de iteraciones.

- $n_cltr_u = 3, n_cltr_i = 3$ y $n_epochs = 15$.

CoClustering	RMSE	MSE	MAE
Scott (737)	0.987042	0.974252	0.766862
Adrian (405)	0.991608	0.983286	0.771540
Jeremy (206)	0.986578	0.973336	0.766151
Olivar (93)	1.004664	1.009349	0.782850
Arias (20)	0.995748	0.991514	0.776153
Pepe (1)	0.997289	0.994585	0.776016

Tabla 25: CoClustering con $n_cltr_u = 3, n_cltr_i = 3$ y $n_epochs = 15$.

- $n_cltr_u = 3, n_cltr_i = 3$ y $n_epochs = 30$.

CoClustering	RMSE	MSE	MAE
Scott (737)	1.006988	1.014024	0.785604
Adrian (405)	0.991345	0.982766	0.771078
Jeremy (206)	1.008007	1.016079	0.786143
Olivar (93)	0.998049	0.996101	0.776859
Arias (20)	0.999786	0.999571	0.779966
Pepe (1)	0.999850	0.999700	0.778717

Tabla 26: CoClustering con $n_cltr_u = 3, n_cltr_i = 3$ y $n_epochs = 30$.

- $n_cltr_u = 3, n_cltr_i = 3$ y $n_epochs = 40$.

CoClustering	RMSE	MSE	MAE
Scott (737)	1.007303	1.014659	0.786998
Adrian (405)	1.005729	1.011491	0.783353
Jeremy (206)	0.998084	0.996173	0.777689
Olivar (93)	0.990997	0.982075	0.769456
Arias (20)	0.982362	0.965035	0.763756
Pepe (1)	1.009713	1.019520	0.787574

Tabla 27: CoClustering con $n_cltr_u = 3, n_cltr_i = 3$ y $n_epochs = 40$.

- $n_cltr_u = 3, n_cltr_i = 3$ y $n_epochs = 60$.

CoClustering	RMSE	MSE	MAE
Scott (737)	0.988761	0.977649	0.769154
Adrian (405)	1.007016	1.014082	0.785904
Jeremy (206)	1.003482	1.006977	0.783223
Olivar (93)	1.004378	1.008775	0.783431
Arias (20)	0.991222	0.982522	0.770541
Pepe (1)	0.993696	0.987432	0.773747

Tabla 28: CoClustering con $n_cltr_u = 3, n_cltr_i = 3$ y $n_epochs = 60$.

Observando los resultados anteriores, parece que el número de iteraciones no afecta demasiado a la calidad de las recomendaciones, por lo que finalmente se han realizado unas recomendaciones del algoritmo variando todos los campos de sus parámetros para ver cómo actúa.

- $n_cltr_u = 20, n_cltr_i = 100$ y $n_epochs = 40$.

CoClustering	RMSE	MSE	MAE
Scott (737)	1.096937	1.203270	0.866456
Adrian (405)	1.078284	1.162696	0.851199
Jeremy (206)	1.085858	1.179089	0.856349
Olivar (93)	1.077721	1.161482	0.852223
Arias (20)	1.055798	1.114710	0.834654
Pepe (1)	1.100913	1.212010	0.869031

Tabla 29: CoClustering con $n_cltr_u = 20, n_cltr_i = 100$ y $n_epochs = 40$.

- $n_cltr_u = 1, n_cltr_i = 1$ y $n_epochs = 5$.

CoClustering	RMSE	MSE	MAE
Scott (737)	0.969275	0.939495	0.750815
Adrian (405)	0.969275	0.939495	0.750815
Jeremy (206)	0.969275	0.939495	0.750815
Olivar (93)	0.969275	0.939495	0.750815
Arias (20)	0.969275	0.939495	0.750815
Pepe (1)	0.969275	0.939495	0.750815

Tabla 30: CoClustering con $n_cltr_u = 1, n_cltr_i = 1$ y $n_epochs = 5$.

- $n_cltr_u = 1$, $n_cltr_i = 1$ y $n_epochs = 1$.

CoClustering	RMSE	MSE	MAE
Scott (737)	0.969275	0.939495	0.750815
Adrian (405)	0.969275	0.939495	0.750815
Jeremy (206)	0.969275	0.939495	0.750815
Olivar (93)	0.969275	0.939495	0.750815
Arias (20)	0.969275	0.939495	0.750815
Pepe (1)	0.969275	0.939495	0.750815

Tabla 31: CoClustering con $n_cltr_u = 1$, $n_cltr_i = 1$ y $n_epochs = 1$.

Viendo los resultados obtenidos, se puede ver cómo apenas es mejorable la calidad de las recomendaciones de este algoritmo. También se puede ver cómo esa pequeña mejoría de calidad es al disminuir los parámetros correspondientes a los clusters, si se aumentan sus valores la calidad empeora. Parece que el número de iteraciones no afecta a las recomendaciones.

Otro detalle a destacar es cómo los valores de las métricas de rendimiento llegan a un mínimo, no se reduce más su valor. Por lo que, tomando como ejemplo los valores mostrados en las tablas anteriores, dejando los parámetros $n_cltr_u = 1$, $n_cltr_i = 1$ y $n_epochs = 20$ habría sido suficiente. Ya que disminuyendo el número de iteraciones no se ha obtenido una mejor calidad.

Después de toda esta comparación, la conclusión que se saca es que no tendríamos por qué modificar los parámetros por defecto para obtener la mejor recomendación. Porque apenas mejora la calidad y porque el número de clusters es muy pequeño, ya que toma la misma referencia para las recomendaciones de todos los usuarios. Respecto al número de iteraciones, para una mejor eficacia del algoritmo tampoco es recomendable dejar un valor pequeño, aunque como hemos visto apenas influye en los resultados obtenidos.

Variaciones en SVD

Por último, al algoritmo SVD se le van a realizar pruebas con 6 de sus parámetros, cuyos nombres son $n_factors$, n_epochs , $init_mean$, $init_std_dev$, lr_all y reg_all . El parámetro $n_factors$ es el número de factores, n_epochs es el número de iteraciones del procedimiento SGD^[3], $init_mean$ es la media de la distribución normal para la inicialización de vectores factoriales, $init_std_dev$ es la desviación típica de la distribución normal para la inicialización de vectores factoriales, lr_all es la tasa de aprendizaje para todos los parámetros y reg_all es el término de regularización para todos los parámetros. Sus valores por defecto son: $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

Para comenzar, se ha modificado el valor del parámetro $n_factors$ de este algoritmo para ver cómo influyen en nuestras recomendaciones:

- $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.619965	0.384356	0.487407
Adrian (405)	0.619638	0.383951	0.487136
Jeremy (206)	0.620191	0.384636	0.487482
Olivar (93)	0.619926	0.384308	0.487402
Arias (20)	0.619711	0.384041	0.487113
Pepe (1)	0.619496	0.383775	0.487026

Tabla 32: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 50$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.614069	0.377081	0.482218
Adrian (405)	0.613449	0.376320	0.481813
Jeremy (206)	0.613474	0.376350	0.481768
Olivar (93)	0.614724	0.377886	0.482912
Arias (20)	0.612928	0.375680	0.481766
Pepe (1)	0.613241	0.376065	0.481719

Tabla 33: SVD con $n_factors = 50$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 80$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.609413	0.371384	0.478756
Adrian (405)	0.607689	0.369286	0.477437
Jeremy (206)	0.609187	0.371109	0.478360
Olivar (93)	0.608802	0.370640	0.477975
Arias (20)	0.609535	0.371533	0.478842
Pepe (1)	0.608854	0.370703	0.478327

Tabla 34: SVD con $n_factors = 80$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.564938	0.319155	0.444556
Adrian (405)	0.563915	0.318000	0.443376
Jeremy (206)	0.563122	0.317107	0.442958
Olivar (93)	0.563558	0.317598	0.443486
Arias (20)	0.564221	0.318346	0.444012
Pepe (1)	0.561790	0.315608	0.441683

Tabla 35: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 800$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.548337	0.300674	0.431884
Adrian (405)	0.548594	0.300956	0.432341
Jeremy (206)	0.549150	0.301566	0.432852
Olivar (93)	0.549235	0.301659	0.432833
Arias (20)	0.547349	0.299590	0.431192
Pepe (1)	0.546743	0.298928	0.430687

Tabla 36: SVD con $n_factors = 800$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 2000$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.538077	0.289527	0.426905
Adrian (405)	0.537640	0.289057	0.426494
Jeremy (206)	0.537800	0.289229	0.426872
Olivar (93)	0.537144	0.288523	0.426254
Arias (20)	0.537521	0.288928	0.426604
Pepe (1)	0.537757	0.289182	0.426713

Tabla 37: SVD con $n_factors = 2000$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 10000$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.777013	0.603750	0.622436
Adrian (405)	0.776907	0.603585	0.622377
Jeremy (206)	0.776512	0.602971	0.622410
Olivar (93)	0.775832	0.601916	0.621961
Arias (20)	0.776324	0.602679	0.622079
Pepe (1)	0.776841	0.603481	0.622622

Tabla 38: SVD con $n_factors = 10000$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

Se puede interpretar de los resultados obtenidos que, disminuyendo los valores de este parámetro, los resultados de RMSE, MSE y MAE mejoran. También mejoran aumentando su valor hasta un valor máximo en el que, si se sigue aumentando, la calidad empeora.

A continuación, se van a realizar una serie de recomendaciones comprobando cómo afecta el número de iteraciones a la calidad.

- $n_factors = 100$, $n_epochs = 1$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.240948	0.058056	0.173111
Adrian (405)	0.241309	0.058230	0.173377
Jeremy (206)	0.241019	0.058090	0.173049
Olivar (93)	0.240966	0.058065	0.172981
Arias (20)	0.241079	0.058119	0.173083
Pepe (1)	0.240954	0.058059	0.172947

Tabla 39: SVD con $n_factors = 100$, $n_epochs = 1$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 100$, $n_epochs = 10$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.531604	0.282603	0.414379
Adrian (405)	0.531881	0.282897	0.414838
Jeremy (206)	0.532739	0.283811	0.415505
Olivar (93)	0.531493	0.282485	0.414571
Arias (20)	0.531758	0.282767	0.414633
Pepe (1)	0.532249	0.283289	0.415012

Tabla 40: SVD con $n_factors = 100$, $n_epochs = 10$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 100$, $n_epochs = 25$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.626171	0.392090	0.492343
Adrian (405)	0.626791	0.392867	0.493272
Jeremy (206)	0.626127	0.392035	0.492414
Olivar (93)	0.627595	0.393876	0.493933
Arias (20)	0.627144	0.393310	0.493465
Pepe (1)	0.626484	0.392483	0.492358

Tabla 41: SVD con $n_factors = 100$, $n_epochs = 25$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 100$, $n_epochs = 45$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.672712	0.452542	0.530227
Adrian (405)	0.673228	0.453237	0.530365
Jeremy (206)	0.674165	0.454499	0.530896
Olivar (93)	0.673103	0.453068	0.530255
Arias (20)	0.676455	0.457591	0.533034
Pepe (1)	0.674399	0.454814	0.531182

Tabla 42: SVD con $n_factors = 100$, $n_epochs = 45$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

Se puede ver que los valores de las métricas de rendimiento y el número de iteraciones son proporcionales, a mayor número de iteraciones, mayor es el valor de estas métricas.

Para aquellos valores de $n_factors$ en los que se encuentran mejores recomendaciones comprobamos si afecta de igual manera el número de iteraciones que para su valor por defecto. Por tanto, $n_factors$ tendrá un valor de 500 y de 1. Se ha escogido 500 en lugar de un valor más alto porque el algoritmo tarda menos en ejecutarse.

- $n_factors = 500$, $n_epochs = 25$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.570449	0.325412	0.449067
Adrian (405)	0.571583	0.326708	0.449857
Jeremy (206)	0.572380	0.327619	0.450720
Olivar (93)	0.573612	0.329030	0.451492
Arias (20)	0.571690	0.326830	0.449772
Pepe (1)	0.571302	0.326386	0.449906

Tabla 43: SVD con $n_factors = 500$, $n_epochs = 25$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 500$, $n_epochs = 10$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.524013	0.274590	0.410715
Adrian (405)	0.523121	0.273656	0.409799
Jeremy (206)	0.525183	0.275817	0.411654
Olivar (93)	0.526782	0.277499	0.412731
Arias (20)	0.526111	0.276793	0.412244
Pepe (1)	0.524859	0.275477	0.410891

Tabla 44: SVD con $n_factors = 500$, $n_epochs = 10$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 1$, $n_epochs = 25$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.641632	0.411692	0.505330
Adrian (405)	0.641766	0.411863	0.505346
Jeremy (206)	0.641615	0.411670	0.505469
Olivar (93)	0.640990	0.410869	0.504789
Arias (20)	0.641370	0.411356	0.505114
Pepe (1)	0.642253	0.412489	0.505867

Tabla 45: SVD con $n_factors = 1$, $n_epochs = 25$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 1$, $n_epochs = 10$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.535303	0.286549	0.417009
Adrian (405)	0.535471	0.286729	0.417149
Jeremy (206)	0.535377	0.286629	0.417090
Olivar (93)	0.535401	0.286654	0.417115
Arias (20)	0.535383	0.286635	0.417113
Pepe (1)	0.535416	0.286670	0.417130

Tabla 46: SVD con $n_factors = 1$, $n_epochs = 10$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

Efectivamente, para cuando mejores recomendaciones se obtienen, el parámetro n_epochs afecta de la misma manera.

Ahora vamos a comprobar en qué afecta variar los valores de $init_mean$ e $init_std$, manteniendo el resto de los valores por defecto.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0.06$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.630249	0.397213	0.497298
Adrian (405)	0.630832	0.397950	0.498265
Jeremy (206)	0.628178	0.394607	0.495259
Olivar (93)	0.628114	0.394528	0.495830
Arias (20)	0.627885	0.394240	0.495440
Pepe (1)	0.628371	0.394850	0.495720

Tabla 47: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0.06$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0.2$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.826744	0.683505	0.677931
Adrian (405)	0.824646	0.680041	0.676249
Jeremy (206)	0.828297	0.686076	0.679776
Olivar (93)	0.828042	0.685654	0.679387
Arias (20)	0.827808	0.685266	0.679062
Pepe (1)	0.829855	0.688659	0.681187

Tabla 48: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0.2$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.2$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.672640	0.452445	0.532792
Adrian (405)	0.674791	0.455343	0.534095
Jeremy (206)	0.674699	0.455219	0.534556
Olivar (93)	0.672827	0.452696	0.532833
Arias (20)	0.673790	0.453993	0.533994
Pepe (1)	0.671543	0.450970	0.531735

Tabla 49: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.2$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.05$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.610750	0.373016	0.479616
Adrian (405)	0.610544	0.372764	0.479591
Jeremy (206)	0.610623	0.372860	0.479677
Olivar (93)	0.610348	0.372525	0.479327
Arias (20)	0.610357	0.372535	0.479416
Pepe (1)	0.610386	0.372571	0.479488

Tabla 50: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.05$, $lr_all = 0.005$ y $reg_all = 0.02$.

Al modificar el valor de la media, los valores de las métricas de rendimiento aumentan, por lo que dejamos su valor por defecto, que es cero. Para la desviación típica, se puede apreciar cómo aumentando o disminuyendo su valor, la calidad de la recomendación tampoco mejora. Por lo que se decide dejar para ambos parámetros sus valores por defecto.

De nuevo, comprobamos si afecta de la misma manera para los valores 500 y 1 de $n_factors$ sin modificar el resto de los parámetros.

- $n_factors = 500$, $n_epochs = 20$, $init_mean = 0.2$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	1.038667	1.078830	0.888685
Adrian (405)	1.037620	1.076656	0.887678
Jeremy (206)	1.038928	1.079372	0.889138
Olivar (93)	1.038147	1.077749	0.888259
Arias (20)	1.038057	1.077561	0.888437
Pepe (1)	1.038311	1.078089	0.888522

Tabla 51: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0.2$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.2$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.815204	0.664557	0.655117
Adrian (405)	0.812182	0.659640	0.653099
Jeremy (206)	0.812697	0.660477	0.653476
Olivar (93)	0.813561	0.661881	0.654136
Arias (20)	0.813241	0.661360	0.654014
Pepe (1)	0.813777	0.662233	0.654348

Tabla 52: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.2$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.05$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.582810	0.339667	0.457218
Adrian (405)	0.581733	0.338413	0.456354
Jeremy (206)	0.582084	0.338822	0.456622
Olivar (93)	0.582179	0.338932	0.456673
Arias (20)	0.581824	0.338520	0.456615
Pepe (1)	0.582748	0.339596	0.457147

Tabla 53: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.05$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 1$, $n_epochs = 20$, $init_mean = 0.2$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.623613	0.388894	0.489852
Adrian (405)	0.625289	0.390986	0.491445
Jeremy (206)	0.625912	0.391765	0.491756
Olivar (93)	0.624931	0.390538	0.490751
Arias (20)	0.624216	0.389646	0.490329
Pepe (1)	0.624463	0.389954	0.490774

Tabla 54: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0.2$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.2$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.622724	0.387786	0.489544
Adrian (405)	0.621275	0.385982	0.488396
Jeremy (206)	0.621492	0.386253	0.488568
Olivar (93)	0.621250	0.385951	0.488565
Arias (20)	0.622149	0.387070	0.489147
Pepe (1)	0.620847	0.385451	0.488278

Tabla 55: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.2$, $lr_all = 0.005$ y $reg_all = 0.02$.

- $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.05$, $lr_all = 0.005$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.619779	0.384126	0.487247
Adrian (405)	0.619625	0.383935	0.487083
Jeremy (206)	0.619658	0.383976	0.487089
Olivar (93)	0.619723	0.384057	0.487169
Arias (20)	0.619665	0.383985	0.487143
Pepe (1)	0.619783	0.384132	0.487221

Tabla 56: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.05$, $lr_all = 0.005$ y $reg_all = 0.02$.

Tal y como esperábamos, en este caso tampoco se ve diferencia, la modificación de estos parámetros no se ven influidos por el número de factores.

Finalmente, se observan los resultados obtenidos modificando la tasa de aprendizaje y el término de regularización de todos los parámetros.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.100556	0.010111	0.079977
Adrian (405)	0.100143	0.010029	0.079731
Jeremy (206)	0.100302	0.010060	0.079818
Olivar (93)	0.100201	0.010040	0.079730
Arias (20)	0.099913	0.009983	0.079516
Pepe (1)	0.100241	0.010048	0.079777

Tabla 57: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0$ y $reg_all = 0.02$.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.003$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.551705	0.304379	0.431831
Adrian (405)	0.552097	0.304811	0.432091
Jeremy (206)	0.552010	0.304715	0.431942
Olivar (93)	0.552358	0.305100	0.432131
Arias (20)	0.551284	0.303914	0.431528
Pepe (1)	0.551896	0.304589	0.431879

Tabla 58: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.003$ y $reg_all = 0.02$.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.007$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.636798	0.405511	0.501003
Adrian (405)	0.637432	0.406320	0.501123
Jeremy (206)	0.634775	0.402939	0.499135
Olivar (93)	0.636167	0.404708	0.500253
Arias (20)	0.634716	0.402865	0.499300
Pepe (1)	0.636254	0.404819	0.500375

Tabla 59: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.007$ y $reg_all = 0.02$.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.01$.

SVD	RMSE	MSE	MAE
Scott (737)	0.614569	0.377695	0.482922
Adrian (405)	0.614185	0.377223	0.482743
Jeremy (206)	0.614181	0.377218	0.482337
Olivar (93)	0.614567	0.377693	0.482576
Arias (20)	0.613871	0.376838	0.482560
Pepe (1)	0.615766	0.379168	0.483695

Tabla 60: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.01$.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.04$.

SVD	RMSE	MSE	MAE
Scott (737)	0.593121	0.351793	0.465526
Adrian (405)	0.591999	0.350462	0.464784
Jeremy (206)	0.592399	0.350937	0.465102
Olivar (93)	0.592519	0.351079	0.465058
Arias (20)	0.592429	0.350972	0.464842
Pepe (1)	0.592454	0.351002	0.465350

Tabla 61: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.04$.

- $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.1$.

SVD	RMSE	MSE	MAE
Scott (737)	0.564267	0.318397	0.442220
Adrian (405)	0.564606	0.318780	0.442461
Jeremy (206)	0.564015	0.318112	0.441973
Olivar (93)	0.563663	0.317716	0.441878
Arias (20)	0.564344	0.318484	0.442296
Pepe (1)	0.564103	0.318212	0.442103

Tabla 62: SVD con $n_factors = 100$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.1$.

Si aumentamos el valor de la tasa de aprendizaje, como se puede observar, la calidad empeora, y dándole un valor inferior al por defecto, mejora. En cambio, con el término de regularización, pasa el caso contrario, los resultados obtenidos son mejores si aumentamos su valor respecto al que obtiene por defecto.

Viendo cómo afectan estos dos parámetros mencionados anteriormente, comprobamos si afectan del mismo modo con los valores del número de factores mencionados anteriormente.

- $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.003$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.536589	0.287928	0.421305
Adrian (405)	0.534066	0.285227	0.418924
Jeremy (206)	0.535007	0.286233	0.419826
Olivar (93)	0.552536	0.305297	0.432250
Arias (20)	0.551611	0.304274	0.431631
Pepe (1)	0.550900	0.303490	0.431052

Tabla 63: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.003$ y $reg_all = 0.02$.

- $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.007$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.574737	0.330323	0.452431
Adrian (405)	0.575770	0.331511	0.453485
Jeremy (206)	0.575238	0.330899	0.452682
Olivar (93)	0.577994	0.334077	0.455104
Arias (20)	0.576027	0.331807	0.453595
Pepe (1)	0.575477	0.331174	0.453182

Tabla 64: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.007$ y $reg_all = 0.02$.

- $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.04$.

SVD	RMSE	MSE	MAE
Scott (737)	0.554545	0.307521	0.436553
Adrian (405)	0.555699	0.308802	0.437177
Jeremy (206)	0.554811	0.307816	0.436570
Olivar (93)	0.555443	0.308516	0.436869
Arias (20)	0.553757	0.306647	0.435831
Pepe (1)	0.556180	0.309336	0.437744

Tabla 65: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.04$.

- $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.1$.

SVD	RMSE	MSE	MAE
Scott (737)	0.539361	0.290910	0.423244
Adrian (405)	0.538188	0.289646	0.422479
Jeremy (206)	0.538458	0.289937	0.422515
Olivar (93)	0.538321	0.289790	0.422769
Arias (20)	0.538434	0.289911	0.422725
Pepe (1)	0.538589	0.290078	0.422727

Tabla 66: SVD con $n_factors = 500$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.1$.

- $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.003$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.558311	0.311711	0.436684
Adrian (405)	0.558472	0.311891	0.436809
Jeremy (206)	0.558414	0.311826	0.436767
Olivar (93)	0.558439	0.311854	0.436793
Arias (20)	0.558454	0.311871	0.436780
Pepe (1)	0.558391	0.311801	0.436724

Tabla 67: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.003$ y $reg_all = 0.02$.

- $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.007$ y $reg_all = 0.02$.

SVD	RMSE	MSE	MAE
Scott (737)	0.651569	0.424542	0.513301
Adrian (405)	0.653783	0.427432	0.514377
Jeremy (206)	0.652496	0.425751	0.513946
Olivar (93)	0.652457	0.425701	0.514168
Arias (20)	0.652790	0.426135	0.514251
Pepe (1)	0.652795	0.426141	0.514087

Tabla 68: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.007$ y $reg_all = 0.02$.

- $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.04$.

SVD	RMSE	MSE	MAE
Scott (737)	0.606484	0.367823	0.476243
Adrian (405)	0.606705	0.368091	0.476296
Jeremy (206)	0.606492	0.367832	0.476218
Olivar (93)	0.606518	0.367864	0.476167
Arias (20)	0.606714	0.368102	0.476222
Pepe (1)	0.606588	0.367949	0.476280

Tabla 69: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.04$.

- $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.1$.

SVD	RMSE	MSE	MAE
Scott (737)	0.572850	0.328157	0.448795
Adrian (405)	0.572918	0.328235	0.448841
Jeremy (206)	0.572886	0.328198	0.448824
Olivar (93)	0.572899	0.328213	0.448813
Arias (20)	0.572879	0.328190	0.448808
Pepe (1)	0.572928	0.328246	0.448842

Tabla 70: SVD con $n_factors = 1$, $n_epochs = 20$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0.005$ y $reg_all = 0.1$.

Podemos ver cómo en ambos casos varía la calidad de la misma manera.

Buscando una calidad óptima por el camino de la disminución de $n_factors$ y n_epochs se han obtenido los siguientes resultados:

- $n_factors = 1$, $n_epochs = 1$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0$ y $reg_all = 0.03$.

SVD	RMSE	MSE	MAE
Scott (737)	0.010163	0.000103	0.006442
Adrian (405)	0.010196	0.000104	0.006377
Jeremy (206)	0.010148	0.000103	0.006496
Olivar (93)	0.009733	0.000095	0.006315
Arias (20)	0.009916	0.000098	0.006216
Pepe (1)	0.009842	0.000097	0.006242

Tabla 71: SVD con $n_factors = 1$, $n_epochs = 1$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0$ y $reg_all = 0.03$.

- $n_factors = 1$, $n_epochs = 1$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0$ y $reg_all = 0.1$.

SVD	RMSE	MSE	MAE
Scott (737)	0.009506	0.000090	0.006052
Adrian (405)	0.009647	0.000093	0.006126
Jeremy (206)	0.010130	0.000103	0.006379
Olivar (93)	0.009227	0.000085	0.005848
Arias (20)	0.009910	0.000098	0.006324
Pepe (1)	0.010221	0.000104	0.006527

Tabla 72: SVD con $n_factors = 1$, $n_epochs = 1$, $init_mean = 0$, $init_std = 0.1$, $lr_all = 0$ y $reg_all = 0.1$.

Con los valores de $n_factors$ y n_epochs establecidos a 1, la media y la desviación típica con sus valores por defecto y la tasa de aprendizaje al mínimo (con un valor de 0) parece que se encuentran las mejores recomendaciones. Respecto al valor del término de regularización, como se ha dicho anteriormente, cuanto mayor sea su valor, mejor será la recomendación. Pero como se observa en estas últimas tablas, por mucho que se aumente su valor, la calidad de la recomendación se mantiene, no hace falta seguir aumentándolo más. Por lo que parece que se ha llegado a la calidad óptima de recomendación para estos usuarios con estos valores.

Estos valores no se consideran fiables debido a que un solo factor es muy poco para generar recomendaciones personalizadas al usuario, aunque sean los mejores resultados obtenidos. Respecto al número de iteraciones, una sola iteración tampoco es fiable a la hora de recomendar al usuario, por ello se ha dejado el valor por defecto, ya que un valor mayor tampoco genera una mejor recomendación. Finalmente, la tasa de aprendizaje tampoco es recomendable ponerla a 0. Porque si, por ejemplo, la magnitud de la gradiente es 2.5 y la tasa de aprendizaje es 0.01, el algoritmo de descenso de gradientes tomará el siguiente punto 0.025 más alejado del punto anterior, en caso de ser 0, tomará el siguiente punto 2.5 más alejado del punto anterior.

En cuanto al otro método que se ha encontrado con un número de factores considerable se puede apreciar cómo la calidad va mejorando hasta llegar a un punto en el que ya no puede ir a mejor y comienza a empeorar, parece que converge. Este punto de convergencia de `n_factors` parece que se puede cambiar con ayuda de los parámetros `lr_all` y `reg_all` y, de esta manera, poder dar un mayor número de factores. No se ha buscado este punto de convergencia debido al número tan alto que se necesita en `n_factors` y, en consecuencia, el tiempo que tarda en ejecutarse con dicho valor, pero en este caso se sabe que está entre 2 000 y 10 000. Esta recomendación es más fiable para la personalización del usuario, aunque por lo que hemos visto la calidad no sea tan buena como en el caso anterior.

Por último, destacar que no se ha obtenido ninguna mejoría modificando la media y la desviación típica porque, al parecer, se modifican en otro tipo de aplicaciones de transmisión para reducir el ruido.

Visualización de recomendaciones

En el archivo `views.py`, hay cinco vistas que son para visualizar las recomendaciones, entre las cuales algunas de ellas son para que el usuario Admin pueda ver todas las recomendaciones de todos los usuarios y también poder cambiar parámetros de ciertos algoritmos para la comparación.

Rec

Esta vista es la que carga la web cuando el usuario accede a recomendaciones, esta opción aparece en la barra de navegación cuando ha iniciado sesión y ha puntuado alguna película (si no ha puntuado ninguna película no puede acceder a esta página). Cuando ha cargado, al usuario le aparecen una serie de botones, cada uno de ellos correspondiente a un algoritmo de recomendación.

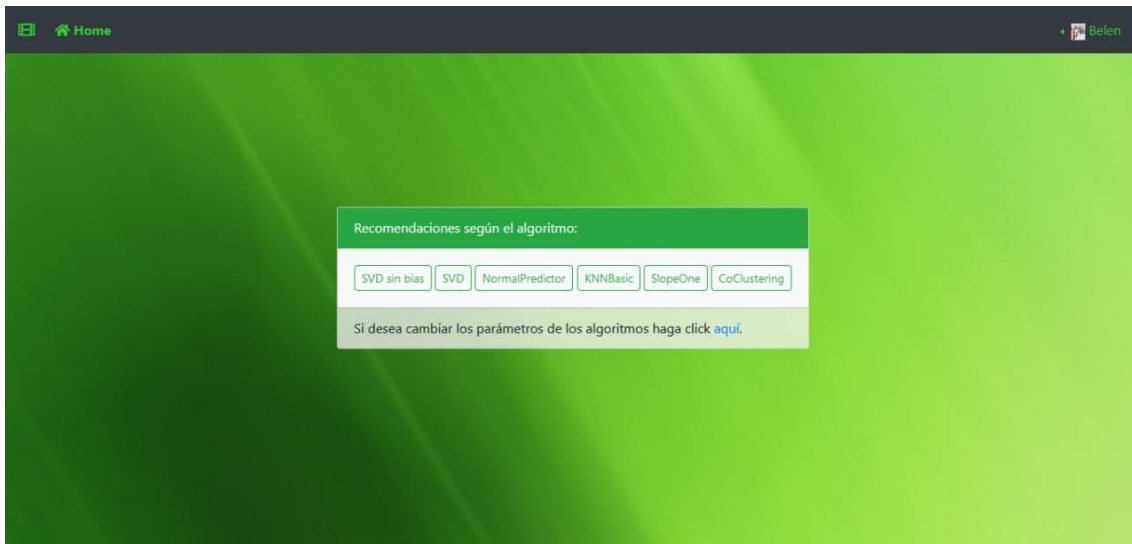


Figura 49: Página de recomendaciones

Cuando hacemos click en uno de ellos, se generan las recomendaciones según ese algoritmo, que puede tardar un tiempo. Una vez que el usuario elige un algoritmo, mientras este se está ejecutando, le aparece una pantalla de espera.

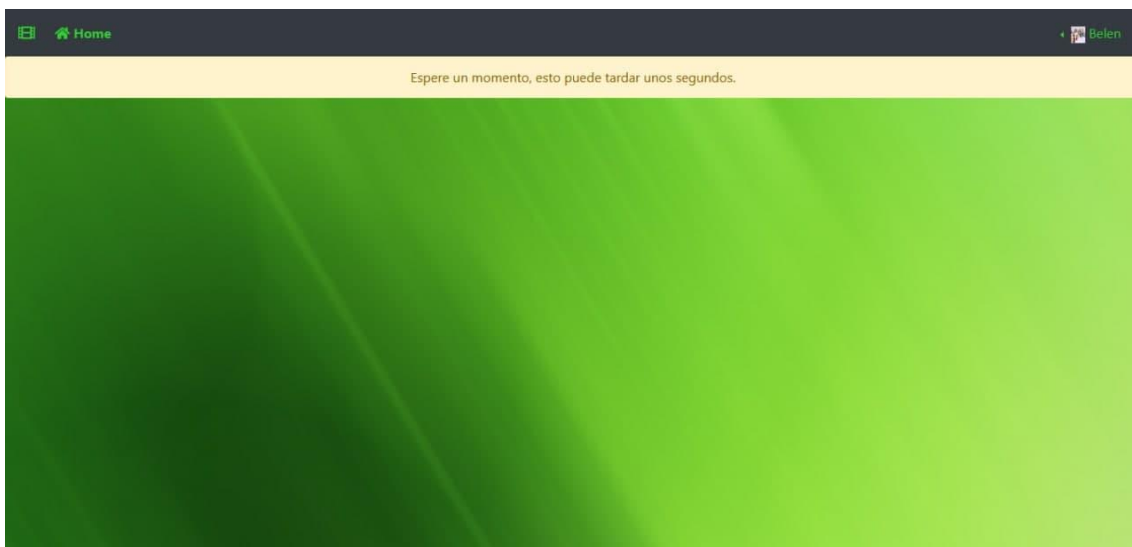


Figura 50: Página de recomendaciones después de hacer click en uno de los botones de algoritmos

Todo esto se lleva a cabo de manera dinámica con ayuda de JavaScript, cuando se realiza la acción de clickar, desaparecen los botones y aparece un mensaje indicando que puede tardar unos segundos como se puede ver en la figura anterior.

Ajax_view

Una vez que el algoritmo se ha ejecutado correctamente, vuelven a aparecer los botones y las recomendaciones para dicho algoritmo. Esta vista tiene un argumento opcional, que es para que cuando el usuario admin sea el que está realizando estas recomendaciones, pase mediante URL la id del usuario del que quiere obtenerlas.

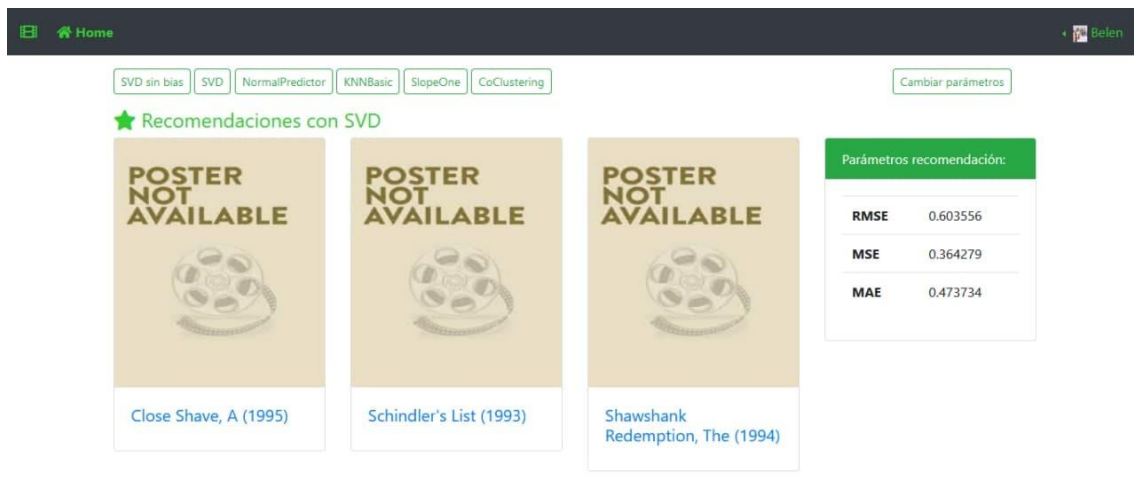


Figura 51: Página de recomendaciones mostrando predicciones

Como se puede apreciar en la figura, se muestran las tres películas con su portada y el título hiperenlazado por si el usuario quiere acceder a la información de alguna de las películas. A la derecha de estas películas, el usuario puede observar los valores de las métricas de rendimiento que se han obtenido para que determine la calidad de esta recomendación.

Se ha usado jQuery Ajax para que desde Python a JavaScript se pueda obtener un objeto de tipo JSON con todos los datos de las predicciones de esta vista de views.py, que es la que se encarga de ejecutar el algoritmo correspondiente y recoger las métricas de rendimiento y el top tres películas más recomendadas para este usuario.

Cuando en Python ya se ha ejecutado todo el código, en la función de éxito de jQuery Ajax, se muestran todos los datos del algoritmo ejecutado. Se ha decidido hacerlo de esta manera, con JavaScript, para que se muestren dinámicamente los resultados de cada algoritmo sin que se pierda la información de las recomendaciones del algoritmo anterior y poder realizar una mejor comparación.

Home Belen

Espere un momento, esto puede tardar unos segundos.

SVD sin bias | SVD | NormalPredictor | KNNBasic | SlopeOne | CoClustering Cambiar parámetros

★ Recomendaciones con SVD

POSTER NOT AVAILABLE

Shawshank Redemption, The (1994)

POSTER NOT AVAILABLE

Schindler's List (1993)

POSTER NOT AVAILABLE

Boot, Das (1981)

Parámetros recomendación:

RMSE	0.605566
MSE	0.366711
MAE	0.475917

Figura 52: Página de recomendaciones después de hacer click en uno de los botones para que se muestre otra recomendación

Home Belen

SVD sin bias | SVD | NormalPredictor | KNNBasic | SlopeOne | CoClustering Cambiar parámetros

★ Recomendaciones con SVD

POSTER NOT AVAILABLE

Shawshank Redemption, The (1994)

POSTER NOT AVAILABLE

Schindler's List (1993)

POSTER NOT AVAILABLE

Boot, Das (1981)

Parámetros recomendación:

RMSE	0.605566
MSE	0.366711
MAE	0.475917

★ Recomendaciones con NormalPredictor

GoldenEye (1995)

From Dusk Till Dawn (1996)

Muppet Treasure Island (1996)

Parámetros recomendación:

RMSE	1.021603
MSE	1.043673
MAE	0.842595

Figura 53: Página de recomendaciones mostrando más de una predicción

Rec_param

Para acceder aquí se puede hacer desde la página de recomendaciones, tanto en la que se carga inicialmente como en la que se carga cuando ya se han realizado recomendaciones.

Esta vista se utiliza para poder cambiar los parámetros de ciertos algoritmos. Aquellos algoritmos a los que se les pueden cambiar parámetros se muestran con un formulario, uno para cada uno de ellos, con los diferentes campos a rellenar. Estos campos por rellenar tienen un placeholder que permite al usuario conocer los valores por defecto que ejecutaría el algoritmo en caso de no pasarle ningún parámetro. En cambio, aquellos algoritmos que no se les pueden cambiar parámetros, se muestran al igual que en la vista rec, con un botón. Se permite la generación de recomendaciones de aquellos algoritmos que no se pueden cambiar parámetros para que el usuario puede comparar también los resultados con estos algoritmos en la misma página.

The screenshot shows a web interface with a dark header containing a home icon and the text 'Home' on the left, and a user profile icon with the name 'Belen' on the right. The main content area features three green-titled panels for parameter configuration:

- Parámetros para KNNBasic:** Includes 'Número máximo de vecinos*' (Default 40) and 'Número mínimo de vecinos*' (Default 1), with a 'Generar recomendación' button.
- Parámetros para CoClustering:** Includes 'Nº user clusters*' (Default 3), 'Nº item clusters*' (Default 3), and 'Nº iteraciones*' (Default 20), with a 'Generar recomendación' button.
- Parámetros para SVD:** Includes 'Nº factores*' (Default 100), 'Nº iteraciones*' (Default 20), 'Media*' (Default 0), 'Desviación típica*' (Default 0.1), 'Tasa de aprendizaje*' (Default 0.005), and 'Término de regularización*' (Default 0.02), with a 'Generar recomendación' button.

Below these panels are three buttons: 'SVD sin bias', 'NormalPredictor', and 'SlopeOne'.

Figura 54: Página en la que se pueden modificar parámetros de los algoritmos

Al igual que con las recomendaciones habituales explicadas anteriormente, se usa jQuery Ajax. Las llamadas que se hacen con los formularios se hacen en la misma vista detectando si se ha realizado una request de tipo POST, en caso afirmativo se comprueba qué formulario ha sido y se llama al algoritmo correspondiente pasándole un argumento extra (además de la id del usuario) de tipo dict con los parámetros que se quieren establecer. En las funciones de estos algoritmos, se comprueba si el argumento opcional está vacío o no, si no lo está, al llamar al propio algoritmo se le pasa como argumento cada uno de los parámetros y, si está vacío, se llama al algoritmo sin pasar ningún argumento. En caso de no detectar una request de tipo POST, se muestran los formularios vacíos y los botones de cada algoritmo como se ve en la figura anterior.

Para los botones, se realiza el mismo procedimiento que en rec, llamando a la vista ajax_view explicada anteriormente cuando se hace click. Una vez que se han generado las recomendaciones, estas se muestran de la misma manera que en rec.

Home Belen

Espere un momento, esto puede tardar unos segundos.

Parámetros para KNNBasic:

Número máximo de vecinos*
40

Número mínimo de vecinos*
1

Generar recomendación

Parámetros para CoClustering:

Nº user clusters* Default 3

Nº item clusters* Default 3

Nº iteraciones* Default 20

Generar recomendación

Parámetros para SVD:

Nº factores* Default 100

Nº iteraciones* Default 20

Media* Default 0

Desviación típica* Default 0.1

Tasa de aprendizaje* Default 0.005

Término de regularización* Default 0.02

Generar recomendación

Figura 55: Página donde se modifican parámetros con mensaje de espera mientras se ejecuta el algoritmo

Home Belen

Parámetros para KNNBasic:

Número máximo de vecinos*
40

Número mínimo de vecinos*
1

Generar recomendación

Parámetros para CoClustering:

Nº user clusters* Default 3

Nº item clusters* Default 3

Nº iteraciones* Default 20

Generar recomendación

Parámetros para SVD:

Nº factores* Default 100

Nº iteraciones* Default 20

Media* Default 0

Desviación típica* Default 0.1

Tasa de aprendizaje* Default 0.005

Término de regularización* Default 0.02

Generar recomendación

★ Recomendaciones con KNNBasic

POSTER NOT AVAILABLE

Great Day in Harlem, A (1994)

They Made Me a Criminal (1939)

Prefontaine (1997)

Parámetros recomendación:

RMSE	0.916218
MSE	0.839456
MAE	0.689813

Figura 56: Página donde se modifican parámetros de algoritmos mostrando recomendaciones

Cuando es el usuario admin el que está accediendo a esta vista (esto se sabe comprobando si se recibe una id por URL), las recomendaciones se generan en base a ese id. En caso de que no se reciba ninguna id, se pone por defecto como None, ya que se trata de un argumento opcional de la vista, y se generan las recomendaciones al usuario logeado.

Recomendaciones para Scott:

Parámetros para KNNBasic:
 Número máximo de vecinos* (Default 40)
 Número mínimo de vecinos* (Default 1)
 Generar recomendación

Parámetros para CoClustering:
 N° user clusters* (Default 3) N° item clusters* (Default 3)
 N° iteraciones* (Default 20)
 Generar recomendación

Parámetros para SVD:
 N° factores* (Default 100) N° iteraciones* (Default 20)
 Media* (Default 0) Desviación típica* (Default 0.1)
 Tasa de aprendizaje* (Default 0.005) Término de regularización* (Default 0.02)
 Generar recomendación

SVD sin bias NormalPredictor SlopeOne

Figura 57: Página donde se modifican parámetros de algoritmos de Admin para un usuario

Admin_rec

A esta página solo puede acceder el usuario Admin, simplemente tiene que clicar en el dropdown de la barra de navegación y elegir la opción *User recs*. Una vez que ha cargado la vista, al usuario le aparece una tabla con los diferentes usuarios y sus datos, además del número de películas que ha puntuado.

Order by: 1 2 last »

User name	Age	Sex	Rated Movies
Scott	22	F	737
Angel	50	F	685
Serena	47	M	636
Erin	35	F	540
Ermesinda	21	M	518
Cara	20	F	493
Adrian	36	M	490
Ivan	19	M	484
Florinda	60	M	480

Figura 58: Página donde Admin escoge el usuario al que le quiere generar recomendaciones

También se le permite ordenar la lista de estos usuarios que han puntuado diversas películas. Por defecto, aparece ordenado en función del número de películas puntuadas en orden descendente, pero puede cambiarlo haciendo que sea en orden ascendente. También puede volver al ascendente con el otro botón.

User name	Age	Sex	Rated Movies
pepe	22	M	1
Belen	22	F	5
Arias	25	F	20
Charles	34	M	20
Munio	50	M	20
Gomesindo	31	F	20
Diego	22	M	20
Jeffrey	48	M	20
Pepi	32	F	20

Figura 59: Página donde Admin escoge el usuario al que le quiere generar recomendaciones ordenado por número de películas puntuadas de menor a mayor

Como son tantos usuarios los que hay registrados en la web, éstos se muestran divididos en varias páginas al igual que el catálogo de películas.

User name	Age	Sex	Rated Movies
Kerry	17	M	62
Gordon	28	M	63
Domenga	28	F	63
Eliseo	47	M	63
Kathryn	22	F	63
Anselmo	43	M	63
Geraldo	28	M	63
Charles	25	F	64
Elizabeth	29	M	64

Figura 60: Página 24 donde Admin escoge el usuario al que le quiere generar recomendaciones ordenado por número de películas puntuadas de menor a mayor

Para que sólo pueda acceder el usuario Admin, en la vista se comprueba quién es el usuario que está intentando acceder, si es uno que no tiene permiso es redirigido a la página principal. De esta forma, tampoco pueden acceder usuarios que no deban mediante la URL.

A continuación, se obtiene de la base de datos toda la información que se necesita mostrar en la web, como los datos de cada usuario y el número de películas que han puntuado. En el archivo HTML se muestra con ayuda de una tabla, cada usuario en una fila, y en la última columna aparece un botón para cada usuario que permite acceder a la página donde se pueden cargar sus recomendaciones. Cuando se pulsa en uno de

ellos, se carga la vista `user_rec` recogiendo como argumento el id del usuario desde la URL.

Para los botones de reordenación, se usó el mismo sistema que en `index` para el catálogo de películas. Cuando el usuario pulsa uno de ellos, se utiliza el método `GET` para detectarlo en la vista, cargar los datos de la base de datos en el orden que se pide e iniciar una sesión para que al pasar de página se siga manteniendo ese orden.

El sistema de paginación también es de la misma manera que en `index`, pasando parámetros mediante el método `GET` para saber en qué página se está y cargar los objetos correspondientes a esa página.

User_rec

Esta vista es prácticamente igual a la de `rec`, la diferencia es que se muestra el nombre del usuario escogido para ver sus recomendaciones tomando su id como argumento. Y el único usuario que puede acceder es `Admin`.

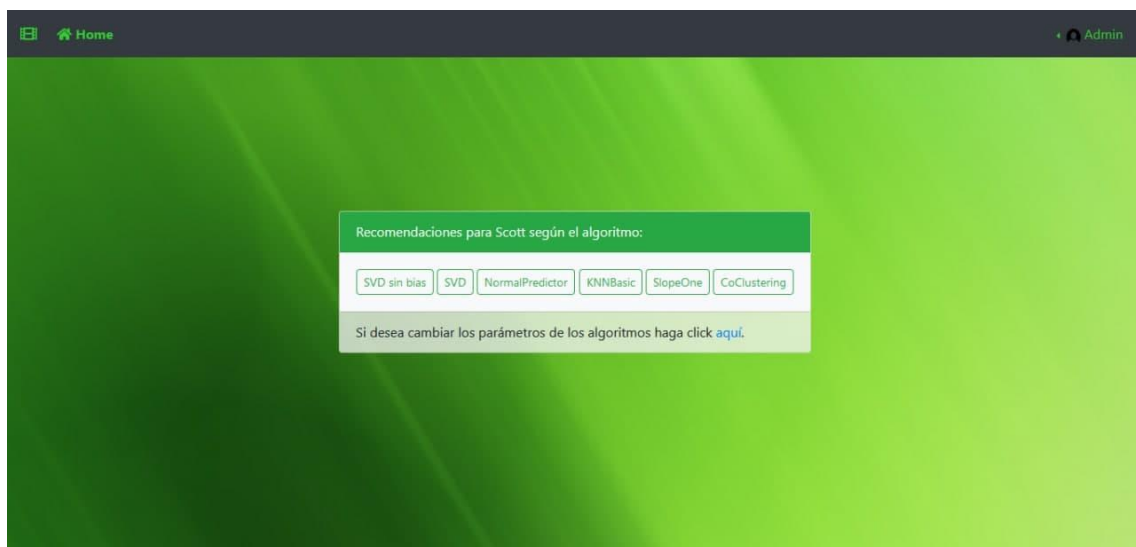


Figura 61: Página de recomendaciones de Admin para otro usuario

Cuando se escoge el algoritmo a ejecutar, la página de espera también es la misma que para cuando el usuario quiere ver sus propias recomendaciones.


Para ejecutar los algoritmos de recomendación también se llama a la vista `Ajax_view`, con la pequeña diferencia de que se le pasa el id del usuario también por URL, del cual se quiere conocer dichas recomendaciones. De esta manera, ya no se toma como referencia el id de la sesión actual del usuario logeado.

Una vez que se muestran las recomendaciones, también se especifica el nombre del usuario al que van dirigidas. Estas recomendaciones se muestran igual que para cualquier usuario.


Home Admin

Recomendaciones para Scott según el algoritmo:


★ Recomendaciones con NormalPredictor



Twelve Monkeys (1995)



Robert A. Heinlein's The Puppet Masters (1994)



Fargo (1996)

Parámetros recomendación:

RMSE	1.021397
MSE	1.043253
MAE	0.842691

Figura 62: Página de recomendaciones de Admin para otro usuario

CONCLUSIONES Y LÍNEAS FUTURAS

Conclusión

Con este proyecto se ha pretendido elaborar una web en la que cualquier usuario pueda navegar en ella de manera intuitiva y fácil. El objetivo de esta web también es que el usuario pueda recibir recomendaciones en base a las películas que ha puntuado.

Durante todo este proceso hemos tenido que aprender un nuevo lenguaje de programación (Python), la toma de decisión de qué framework se adaptaba mejor al proyecto y cómo manejarlo y seleccionar la base de datos que mejor convenía para esta web.

En cuanto a la elección del framework, como ya se ha mencionado anteriormente, los más destacables son Django, Flask y Turbogears. La web se ha desarrollado en Django debido a que se adapta mejor a webs más complejas y hay abundante documentación disponible. Una vez que se comenzó a desarrollar la web con este framework, poco a poco fue más fácil y fluido el desarrollo, ya que se empezó desde cero con este lenguaje de programación.

Django utiliza una base de datos por defecto llamada SQLite, aunque se encontró en toda esta documentación que era una base de datos simple y que podía utilizarse otra. Después de investigar, lo más recomendado en este trabajo es PostgreSQL ya que funciona mejor con este framework. Trabajar con esta base de datos ha sido muy parecido que hacerlo con MySQL, a diferencia de que algunas consultas había que perfeccionarlas con otro enfoque o saltaban errores.

En la parte de algoritmos de recomendación, había que integrar un algoritmo propio que actúa como el SVD sin bias y, además, hacer una selección de algoritmos de la librería surprise (para el uso de ésta se necesitaba desarrollar la web en Python) e integrarlos también en la web. De esta manera, el usuario puede generar diversas recomendaciones, obtener cada uno de los resultados con cada algoritmo y realizar sus propias comparaciones.

A la hora de implementar el algoritmo propio de filtrado colaborativo en Python, hubo que entender qué hacía el algoritmo e ir viendo paso a paso cómo funcionaba hasta que después de diversas pruebas consiguió predecir recomendaciones.

En cuanto a los algoritmos de la librería surprise, la selección fue orientada a algoritmos con diferentes maneras de generación de recomendaciones. Se escogió NormalPredictor que se basa en algoritmos básicos, KNNBasic que se basa en aproximaciones por vecinos más cercanos, SVD en la factorización de matrices, SlopeOne que es simple pero preciso y CoClustering que se basa en agrupación por clusters.

Para poder interpretar la calidad de las recomendaciones, se ha buscado información sobre las métricas de rendimiento y cómo interpretarlas para ser mostradas también junto a la recomendación de películas.

Después de realizar las comparaciones entre cada uno de los algoritmos, basándose en los resultados, se podría decir que el algoritmo SVD es el que mejores recomendaciones ha generado debido a su valor en las métricas de rendimiento, ya que era el valor más pequeño. Esto puede ser debido a que parece un algoritmo más complejo que tiene en cuenta más factores respecto al resto de algoritmos que se muestran en la web. También se ha notado sutilmente que sus recomendaciones son mejores cuando el usuario ha puntuado más películas ya que se tiene una mejor idea de los gustos de este.

Respecto al resto de algoritmos, el creado desde cero que actúa como SVD sin bias, se nota a la hora de realizar las recomendaciones que no tiene estas bias para perfeccionar la recomendación. NormalPredictor parece que se centra en generar mejores recomendaciones cuantos menos datos se recojan del usuario. KNNBasic presenta exactamente la misma calidad para todos los usuarios, indistintamente de si han votado más o menos películas. SlopeOne es exactamente igual a KNNBasic a excepción de que sus recomendaciones son de peor calidad según lo que se ha podido ver en las métricas de rendimientos. Y CoClustering no es de los algoritmos que mejores recomendaciones presenta, pero su calidad es más o menos la misma en todos los usuarios.

A continuación, para poder realizar una comparación de algoritmos más exhaustiva, se implementó en la web la posibilidad de cambiar ciertos parámetros de algunos algoritmos de la librería surprise. Estos fueron KNNBasic, CoClustering y SVD ya que, de los cinco elegidos, son los únicos con parámetros externos (parámetros que no se obtienen de la base de datos) que pueden ser modificados.

Cuando ya estaba todo preparado en la web, con ayuda del usuario Admin, se hicieron varias pruebas para ver el comportamiento de cada uno de los algoritmos e interpretar el porqué de dichos resultados. Este proceso ayudó a comprender mejor cómo funcionan las recomendaciones de machine learning y en base a qué datos trabaja cada uno de ellos.

Dificultades encontradas

A lo largo de todo el proyecto se han presentado varias dificultades. Al comienzo de este, lo más difícil fue aprender la sintaxis de un lenguaje de programación nuevo, ya que no se sabía nada sobre Python.

Después de superar este obstáculo, lo siguiente fue trabajar con el framework Django, con el cual se realizó un tutorial para comprender mejor su funcionamiento. Cómo se crean las aplicaciones en él, cómo conectarlo a una base de datos que no es la que tiene por defecto, qué hace cada archivo de los generados al crear la aplicación...

Como ya he mencionado, el proceso de conseguir que Django conectara con la base de datos PostgreSQL fue una de las dificultades. De hecho, se tuvo que instalar psycopg2 para que se pudiera establecer la conexión. Y a la hora de introducir los datos en las tablas saltaban errores, por lo que se hizo introduciendo los datos de una tabla, luego los de la siguiente...

Otra dificultad fue el hecho de desarrollar una página web en este entorno, ya que ya se habían realizado otras webs anteriormente con otro lenguaje de programación (PHP) y ya se tenía más experiencia en ese ámbito, pero no en éste.

En la parte de la web en la que se muestran las recomendaciones también hubo ciertas dificultades a la hora de decidir cómo mostrarlas, de tal manera que no desapareciera la recomendación de otro algoritmo y poder comparar mejor. Se planteó hacerlo como el algoritmo propio de filtrado colaborativo guardando las recomendaciones en la base de datos, pero al final se usó jQuery Ajax para que aparecieran dinámicamente.

Finalmente, la elaboración de un algoritmo propio, el entender cada uno de los algoritmos e interpretar los resultados en función de cada uno de sus parámetros, son de las mayores dificultades del proyecto.

Líneas futuras

Django permite generar tantas aplicaciones dentro del proyecto general que se crea. Esto hace que, para la autenticación, el registro y demás utilidades de la web se puedan realizar aplicaciones para cada uno de ellos y tener ciertas ventajas en este tipo de webs más complejas. Esta web está hecha en una sola aplicación, pero es más recomendable para webs más sencillas. Por lo que una línea futura a tener en cuenta es generar varias webs con sus diferentes utilidades.

El front-end de la web se podría mejorar utilizando plantillas, por ejemplo. También haber mejorado el cómo se visualiza en un teléfono móvil u otro dispositivo que no sea un ordenador estándar.

Respecto a la visualización de las recomendaciones, el usuario no tiene por qué saber interpretar las métricas de rendimiento e intuir qué algoritmo ha realizado una mejor recomendación. Se podrían, por ejemplo, haber mostrado de color verde estos valores cuando se considera una buena recomendación, de color amarillo cuando no es ni buena ni mala y de color rojo cuando no se considera una de las mejores recomendaciones.

En la parte de comprobar cómo afecta a cada uno de los algoritmos el modificar algunos de sus parámetros. En SVD se podrían haber realizado más pruebas con más de sus parámetros, como probar el algoritmo sin bias o modificar otros parámetros no recogidos en la web. Por ejemplo, en la web se trata la tasa de aprendizaje y el término de regularización de todos los parámetros, pero podrían modificarse sólo algunos de ellos. Otra puntualización a tener en cuenta es que la media y la desviación típica

iniciales no se suelen modificar en esta aplicación del algoritmo, por lo que se podría eliminar de la web la posibilidad de modificarlas.

También, cuando el usuario va a tratar de probar diferentes valores en los parámetros de los algoritmos, se debería haber especificado, en los campos necesarios, entre qué valores suelen estar comprendidos para orientar mejor al usuario.

Finalmente, una vez terminada la web, se podría haber preparado para que fuera reutilizable. Es decir, preparar la aplicación web en un paquete con todo lo necesario para que otra persona se lo pueda descargar y utilizar la web en su ordenador.

Bibliografía

- Aurelien Geron, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems", 2nd Ed., O'Reilly, 2019

Recursos web

Documentación Django

- <https://docs.djangoproject.com/es/3.1/intro/>
- <https://docs.djangoproject.com/en/3.0/topics/db/models/>
- <https://docs.djangoproject.com/en/3.1/ref/models/fields/>
- <https://docs.djangoproject.com/en/3.0/ref/models/options/>
- <https://docs.djangoproject.com/en/3.0/howto/static-files/>
- <https://docs.djangoproject.com/en/3.1/ref/templates/builtins/>
- <https://docs.djangoproject.com/en/3.0/howto/custom-template-tags/>
- <https://docs.djangoproject.com/en/3.0/topics/pagination/>
- <https://docs.djangoproject.com/en/3.0/topics/forms/>
- <https://docs.djangoproject.com/en/3.0/topics/http/file-uploads/>
- <https://docs.djangoproject.com/en/3.0/ref/forms/fields/>
- <https://docs.djangoproject.com/en/3.0/topics/http/sessions/>
- <https://docs.djangoproject.com/en/3.0/ref/models/querysets/>
- <https://docs.djangoproject.com/en/3.1/topics/files/>

Integración base de datos PostgreSQL

- <https://www.enterprisedb.com/postgres-tutorials/why-django-so-impressive-developing-postgresql-and-python>
- https://tutorial-extensions.djangogirls.org/es/optional_postgresql_installation/
- <https://www.postgresqltutorial.com/import-csv-file-into-posgresql-table/>
- <https://stackoverflow.com/questions/4872077/how-to-add-a-new-column-to-the-beginning-of-csv-file>

Bootstrap

- <https://blog.nubecolectiva.com/como-integrar-django-y-bootstrap-4/>
- <https://simpleisbetterthancomplex.com/tutorial/2018/08/13/how-to-use-bootstrap-4-forms-with-django.html>

Font Awesome

- <https://pypi.org/project/django-static-fontawesome/>

NumPy

- <https://numpy.org/install/>
- <https://numpy.org/doc/stable/user/quickstart.html>

Scikit-learn

- <https://scikit-learn.org/stable/install.html>
- https://scikit-learn.org/stable/user_guide.html

Creación algoritmo SVD sin bias

- https://www.w3schools.com/python/python_lambda.asp
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin_cg.html

Integrar algoritmos de surprise lib

- <http://surpriselib.com/>
- <https://surprise.readthedocs.io/en/stable/FAQ.html#how-to-get-the-top-n-recommendations-for-each-user>
- https://surprise.readthedocs.io/en/stable/getting_started.html#use-a-custom-dataset
- https://surprise.readthedocs.io/en/stable/getting_started.html
- https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html
- <https://surprise.readthedocs.io/en/stable/accuracy.html>

Funcionamiento AJAX

- <https://data-flair.training/blogs/ajax-in-django/>
- <https://stackoverflow.com/questions/7335780/how-to-post-a-django-form-with-ajax-jquery>
- <https://pytutorial.com/how-to-submit-a-form-with-django-and-ajax>

Tecnologías usadas

- <https://developer.mozilla.org/es/docs/Web/HTTP/Overview>
- <https://es.wikipedia.org/wiki/Python>
- <https://es.wikipedia.org/wiki/JavaScript>
- <https://es.wikipedia.org/wiki/HTML>
- [https://es.wikipedia.org/wiki/Hoja de estilos en cascada](https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada)
- [https://es.wikipedia.org/wiki/Bootstrap \(framework\)](https://es.wikipedia.org/wiki/Bootstrap_(framework))
- [https://es.wikipedia.org/wiki/Django \(framework\)](https://es.wikipedia.org/wiki/Django_(framework))
- <https://es.wikipedia.org/wiki/Scikit-learn>
- <https://es.wikipedia.org/wiki/NumPy>
- <https://es.wikipedia.org/wiki/SciPy>
- <https://es.wikipedia.org/wiki/JQuery>
- <https://es.wikipedia.org/wiki/PostgreSQL>
- <https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores>

ANEXOS

Estructura de la base de datos

Tabla web_genre

Almacena el género de las películas.

Columna	Tipo	Ordenamiento	Nulable	Por omisión
name	text		not null	
id	smallint		not null	

Índices:
"web_genre_pkey" PRIMARY KEY, btree (id)

Restricciones CHECK:
"web_genre_id_check" CHECK (id >= 0)

Figura 63: Estructura de la tabla web_genre

Tabla web_movie

Almacena la información de cada película.

Columna	Tipo	Ordenamiento	Nulable	Por omisión
id	integer		not null	nextval('web_movie_id_seq'::regclass)
title	text		not null	
date	timestamp with time zone		not null	
url_imdb	text			
url_pic	text		not null	
descr	text			

Índices:
"web_movie_pkey" PRIMARY KEY, btree (id)

Figura 64: Estructura de la tabla web_movie

Tabla web_moviecomments

Almacena los comentarios realizados a las películas.

Columna	Tipo	Ordenamiento	Nulable	Por omisión
id	integer		not null	nextval('web_moviecomments_id_seq'::regclass)
movie_id	integer		not null	
user_id	integer		not null	
comment	text			

Índices:
"web_moviecomments_pkey" PRIMARY KEY, btree (id)

Figura 65: Estructura de la tabla web_moviecomments

Tabla web_moviegenre

Almacena los géneros de cada película.

Columna	Tipo	Ordenamiento	Nulable	Por omisión
id	integer		not null	nextval('web_moviegenre_id_seq'::regclass)
movie_id	integer		not null	
genre	smallint		not null	

Índices:
"web_moviegenre_pkey" PRIMARY KEY, btree (id)
Restricciones CHECK:
"web_moviegenre_genre_check" CHECK (genre >= 0)
"web_moviegenre_movie_id_check" CHECK (movie_id >= 0)

Figura 66: Estructura de la tabla web_moviegenre

Tabla web_recs

Almacena las recomendaciones para cada usuario, junto con la fecha en la que se creó la recomendación.

Columna	Tipo	Ordenamiento	Nulable	Por omisión
id	integer		not null	nextval('web_recs_id_seq'::regclass)
user_id	integer		not null	
movie_id	integer		not null	
rec_score	double precision		not null	
time	timestamp with time zone		not null	

Índices:
"web_recs_pkey" PRIMARY KEY, btree (id)
"web_recs_user_id_movie_id_0d17aa79_uniq" UNIQUE CONSTRAINT, btree (user_id, movie_id)
"web_recs_movie_id_076c949c" btree (movie_id)
"web_recs_user_id_a7df593f" btree (user_id)

Figura 67: Estructura de la tabla web_recs

Tabla web_users

Almacena la información de cada usuario. ATENCIÓN: el campo passwd almacena un hash SHA1 de la clave del usuario.

Columna	Tipo	Ordenamiento	Nulable	Por omisión
id	integer		not null	nextval('web_users_id_seq'::regclass)
name	text		not null	
edad	smallint		not null	
sex	character varying(1)		not null	
ocupacion	character varying(13)			
pic	text			
passwd	character varying(40)		not null	

Índices:
"web_users_pkey" PRIMARY KEY, btree (id)
Restricciones CHECK:
"web_users_edad_check" CHECK (edad >= 0)

Figura 68: Estructura de la tabla web_users

Tabla web_user_score

Almacena las puntuaciones de los usuarios a las películas.

Columna	Tipo	Ordenamiento	Nulable	Por omisión
id	integer		not null	nextval('web_user_score_id_seq'::regclass)
id_user	integer		not null	
id_movie	integer		not null	
score	smallint		not null	
time	timestamp with time zone		not null	

Índices:
"web_user_score_pkey" PRIMARY KEY, btree (id)
"web_user_score_id_user_id_movie_aaaa9f13_uniq" UNIQUE CONSTRAINT, btree (id_user, id_movie)
Restricciones CHECK:
"web_user_score_score_check" CHECK (score >= 0)

Figura 69: Estructura de la tabla web_user_score

Media ponderada

Puede comprobar que este tipo de aplicaciones presentan el problema de cómo agregar puntuaciones para poder comparar una película con otra. Por ejemplo, ¿está mejor valorada una película con 356 votos y media 3.8 que una con 10 votos y media 4.8?

Para obtener una puntuación ponderada, se aplicará la siguiente fórmula a cada película i:

$$p_i = \frac{N \cdot R + n_i \cdot r_i}{N + n_i}$$

donde N es el número total de películas, R es la puntuación media de todas las películas, n_i es el número de puntuaciones de la película i y r_i es la puntuación media de la película i.

SGD (Stochastic gradient descent)

El algoritmo de gradiente estocástico, también conocido por gradiente descendente incremental, es un método iterativo para optimizar una función objetivo con propiedades de suavidad adecuadas (por ejemplo, diferenciable o subdiferenciable). Puede considerarse como una aproximación estocástica de la optimización del descenso del gradiente, ya que reemplaza el gradiente real (calculado a partir de todo el conjunto de datos) por una estimación del mismo (calculado a partir de un subconjunto de datos seleccionado al azar). Especialmente en problemas de optimización de alta dimensión, esto reduce la carga computacional, logrando iteraciones más rápidas en el comercio para una tasa de convergencia más baja.

Si bien la idea básica detrás de la aproximación estocástica se remonta al algoritmo de Robbins-Monro de la década de 1950, el descenso de gradiente estocástico se ha convertido en un método de optimización importante en el aprendizaje automático.