


Facultad de Ciencias
UNAM

Práctica 2

Profesor

Estefanía Prieto Larios
estefaniaprieto@ciencias.unam.mx

Ayudantes

Galván Gámez Edwin Antonio
agalvan@astro.unam.mx
Muñiz Patiño Andrea Fernanda
mu.andrea@ciencias.unam.mx
Robles Ríos Rafael
phalian360@hotmail.com

Integrantes y correos:

Casiano Morales Rodrigo
rodrigocasianom@ciencias.unam.mx

Cortés Jiménez Claudio Amaury
amauryc93j@gmail.com

amaurycortes_jimenez@ciencias.unam.mx

Lemus Aguilar Cuauhtémoc
clemus@ciencias.unam.mx

Puntos a cubrir

La documentación deberá atender los siguientes puntos:

1. Una breve descripción del objetivo del sistema de recomendación y por qué es necesario un sistema de recomendación.
2. Las características que considera el perfil de tu usuario.
3. Cuáles son los datos que necesitas y por qué los necesitas para poder hacer la recomendación a tu usuario.

Qué es lo que va a recomendar tu sistema y a base de qué datos o elementos.

1. Qué técnica de filtrado de información propones que se use, indicando qué tipo de algoritmo de recomendación consideran que funcionará mejor (basado en ítems) y por qué.
2. Describe mediante un ejemplo, el funcionamiento de tu sistema de recomendación. Describe tu sistema de recomendación como un agente inteligente haciendo explícita su REAS.
3. ¿Cuál es el esquema que se utiliza para representar el conocimiento? y ¿Cuáles son las propiedades de ese esquema?
4. ¿Qué tipo de esquema de representación de conocimiento es (declarativo o procedimental, etc.)?
5. ¿Qué estrategias se pueden usar para manipular este tipo de conocimiento?

Objetivo

- El objetivo general del sistema de recomendación es brindar al usuario un listado de películas que le pueden agradar basándose en sus gustos particulares
- Las películas de elección serán elegidas por medio de una inteligencia artificial que tomará un histórico de reseñas de películas con fecha de calificación entre 29 de Marzo 1996 y 24 de Septiembre de 2018.
- El usuario deberá seleccionar el número de películas que quiera y calificarlas con un rating dentro del mismo rango que el set de datos (Del 0.5 al 5 con incrementos de 0.5 i.e. 0.5,1,1.5,...,4.5,5)
- El sistema de recomendación realizará elecciones al usuario basado en la opinión de 610 usuarios activos que han calificado al menos 20 películas (como se especificará más adelante en la sección **Fuentes de datos**)

Fuentes de datos

El set de datos (ml-latest-small) describe el rating de 5-estrellas y una opción abierta de etiquetado para actividad de MovieLens, un sistema de recomendaciones de películas. Contiene 100,836 ratings y 3,683 aplicaciones de etiquetado a través de 9,742 películas. Esta información fue generada por 610 usuarios entre el 29 de Marzo de 1996 y 24 de Septiembre de 2018. El conjunto de datos fue generado el 26 de Septiembre de 2018.

Características y contexto de la información

- Los usuarios fueron seleccionados al azar
- Todos los usuarios habían rankeado o etiquetado al menos 20 películas
- Ninguna información demográfica es incluída
- Cada usuario es representado por un id, y ninguna información adicional está dada

Fuentes de datos

Los datos están contenidos en los archivos:

- `links.csv`
- `movies.csv`
- `ratings.csv`
- `tags.csv`

Es un conjunto de datos en desarrollo. Como tal, puede cambiar a través del tiempo y no es un set de datos para resultados de investigación compartidos.

Éste y otros conjuntos de datos de GroupLens son disponibles para el público en: http://grouplens.org/datasets/.

Contenido y uso de archivos

Formato y Codificación (Encoding)

- Los archivos del conjunto de datos están escritos como .csvs con encabezado de un sólo renglón. Columnas que contienen comas (,) son englobadas entre doble comilla ().
- Los archivos están codificados como UTF-8

  Observación:

  Si los datos no son desplegados correctamente caracteres acentuados en títulos de película o valores de etiqutas (Por ejemplo: Misérables, Les (1995)) hay que asegurarse de que en el momento de lectura, sea configurado el UTF-8.
- Todos los timestamps están en tiempo coordinado universal, UTC.

Identificadores de Usuario

Usuarios seleccionados al azar (como previamente señalado). Sus ids han sido anonimizados y son consistentes con su relación a `ratings.csv` y `tags.csv`.

Identificadores de Película

Sólo películas con al menos un rating o etiqueta son incluídos en el conjunto de datos. Esos identificadores son consistentes con aquellos usados en con las URLs del sitio web https://movielens.org/movies/1 y son consistentes con su relación a `ratings.csv`, `tags.csv`, `movies.csv` y `links.csv`.

Estructura de Ratings - (ratings.csv)

Todos los ratings de películas están contenidos en el archivo `ratings.csv`, el archivo tiene los siguientes atributos:

- **userId** Identificador de usuario
- **movieId** Identiticador de película
- **rating** Clasificación entre 0.5 y 5.0
- **timestamp** Fecha y tiempo

Los ratings están en una escala de 5-estrellas con incrementos de media estrella,

i.e. Rango = (0.5, 1, ..., 4.5, 5.0)

Estructura de Tags - (tags.csv)

Todas las etiquetas están contenidas en el archivo `tags.csv`, el archivo tiene los siguientes atributos:

- **userId** Identificador de usuario
- **movieId** Identificador de película
- **tag** Etiqueta
- **timestamp** Fecha y tiempo

Las etiquetas son datos generados por usuarios acerca de películas. Cada etiqueta es típicamente una sóla palabra una frase corta. El significado, valor y propósito es determinado por cada usuario.

Los atributos son los siguientes:

- **movieId** Identificador de película
- **title** Título
- **genres** Géneros (atributo multivaluado)

Los géneros son los siguientes

1. Action
2. Adventure
3. Animation
4. Children's
5. Comedy
6. Crime
7. Documentary
8. Drama
9. Fantasy
10. Film-Noir
11. Horror
12. Musical
13. Mystery
14. Romance
15. Sci-Fi
16. Thriller
17. War

18. Western
19. (no genres listed)

Estructura de Links - (links.csv)

Los atributos son cualidades de películas que nos permiten relacionar por medio de la llave primaria de identificador de película, otros identificadores; y son los siguientes:

- **movieId** Identificador de película
- **imdbId** Identificador de IMDB
- **tmdbId** Identificador de TheMovieDB

Ejemplo: Película - Toy Story

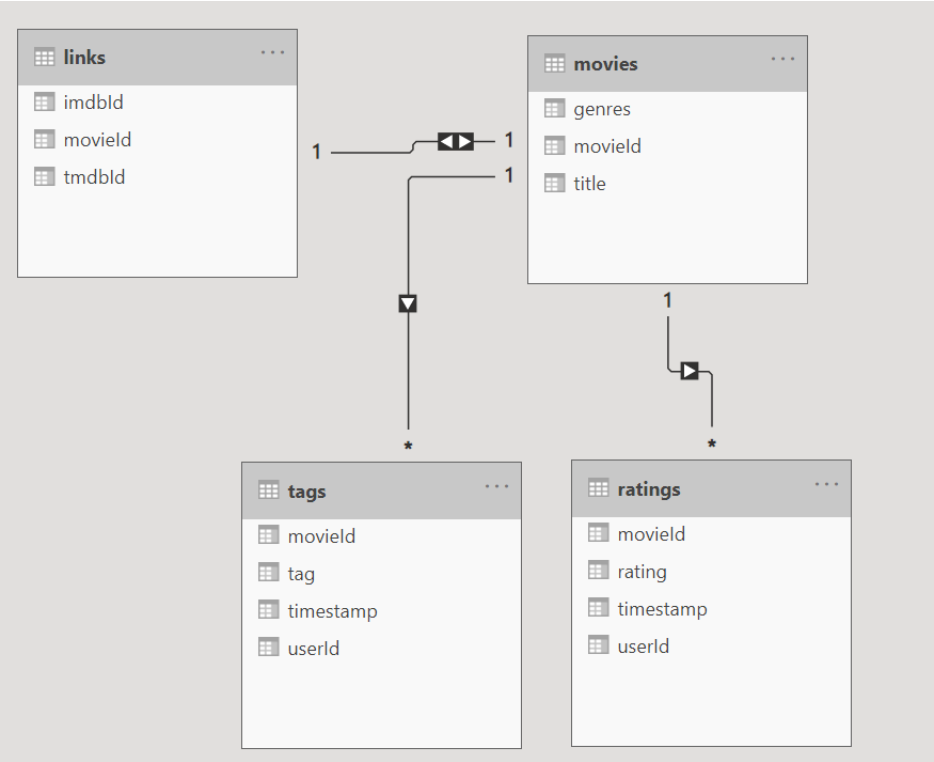
movieId	imdbId	tmdbId
1	tt0114709	862

De esta tabla se derivan las urls en donde se encuentran las películas

id	URL
movieId	https://movielens.org/movies/1
imdbId	http://www.imdb.com/title/tt0114709/
tmdbId	https://www.themoviedb.org/movie/862

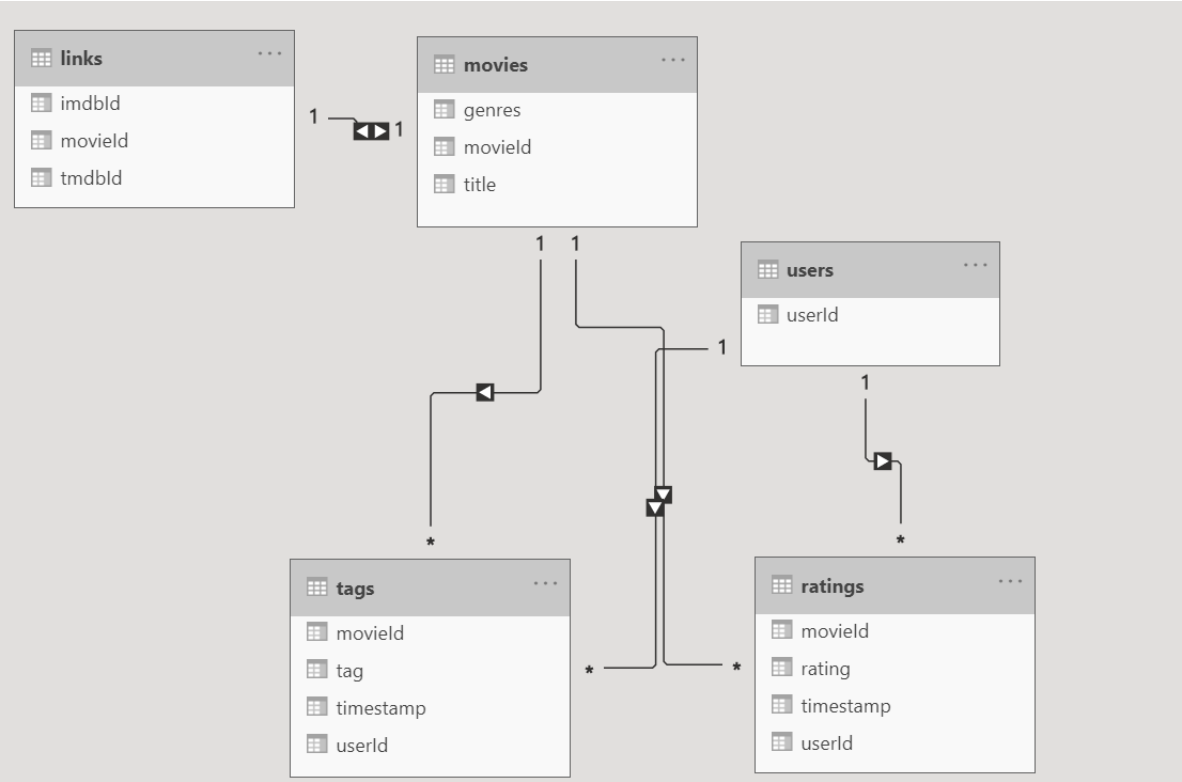
Modelo de datos

Las relaciones entre tablas en un modelo entidad-relación del modelo de datos en cuestión está representado por:



Del modelo original, se deriva otra entidad adicional importante, la de usuarios, que se genera por medio de la siguiente operación de álgebra relacional

$$\text{DISTINCT}(\text{tags}[\text{userId}]) \cup \text{DISTINCT}(\text{ratings}[\text{userId}])$$



El modelo induce a generar preguntas acerca de las dos principales entidades **películas** y **usuarios**, los modelos de aprendizaje supervisados o no supervisados que se generarán en este análisis. Se procederá a realizar un análisis exploratorio de los datos de `tags.csv` y `ratings.csv` así como agrupaciones entre la unión de `users.csv` y `movies.csv` con `ratings.csv` y `tags.csv` por medio de JOINs.

No se utilizará la tabla links ya que sólo contienen otros atributos nominales que a la vez son llave primaria por película, se extraerán los atributos año de lanzamiento (Release_Year) y se generarán 19 columnnas adicionales indicando si la película es del género o no con 1s ó 0s correspondientemente.

Tampoco la tabla tags debido a que esa tabla se compone de una descripción en texto de la película, sobre la cual puede ser descriptiva indicando características como el género, actores principales, reseñas breves o texto intelegible y sólo aumentan la complejidad del problema.

Definición del problema

Se hará una descripción de las REAs de sistema de recomendación de películas que se desarrollará:

Tipo de Agente: Sistema de recomendación de películas

- **Medidas de rendimiento**

  Usuarios conformes con las recomendaciones de películas brindadas.

- **Entorno**

  Películas, usuarios, calificaciones históricas (ratings).

- **Actuadores**

  Lista de películas más recomendadas.

- **Sensores**

  Listado de películas y rating brindado por el usuario.

Descripción de la propuesta e implementación

El sistema de recomendación utilizará la técnica de matriz user-item desplegado de la siguiente manera:

Ejemplo:

Alice: **(like)** Shrek, Snow White, **(dislike)** Superman Bob: **(like)** Snow White, Superman **(dislike)** Spiderman Chris: **(like)** Spider-Man **(dislike)** Snow White Tony: **(like)** Shrek, **(dislike)** Spider-Man

	Shrek	Snow White	Spider-Man	Superman
Alice	like	like		dislike
Chris		dislike	like	
Tony	like		dislike	

Una manera de abordar el problema de recomendaciones se basará sólo con calificaciones favorables y así que esta misma matriz se generará conteniendo sólo 1s y 0s, si la calificación dada a la película es mayor a la calificación del del cuarto cuartil Q_4, i.e. el valor x_0 tal que $\mathbb{P}[X = x_0] = .75$ donde X es la variable aleatoria que representa el rating de la película . Se calcula el cuarto cuartil Q_4 por medio del siguiente código, líneas 8 y 10 del archivo **recommend_movies.py**.

```
In [8]: # Compute the Q_4 quartile for movie's ratings
        Q4_ratings = np.quantile(df_ratings[['rating']],0.75)
        Q4_ratings

Out[8]: 4.0
```

Se genera la columna evaluando el rating vs. `Q4_ratings` y se pivotea el pandas dataframe por **user_Id** en renglones y **movie_Id** por la nueva columna:

```
In [10]: # Pivot the binary rating using userId as rows and movieId columns columns
         df_user_item = df_ratings.pivot(index='userId', columns='movieId')['like']
         df_user_item
```

Out[10]:

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581	193583	193585	193587	193609
userId																					
1	1.0	NaN	1.0	NaN	NaN	1.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NaN	1.0	1.0	0.0	1.0	1.0	1.0	0.0	NaN	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Basta con rellenar los registros nulos con 0s, esto se genera con la función **fillna** de pandas

```
In [11]: df_user_item.fillna(0)
```

Out[11]:

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581	193583	193585	193587	193609
userId																					
1	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Siguiendo la metodología del artículo *A Survey of collaborative Filtering Techniques*, referenciado en la bibliografía, se tiene que aplicar un algoritmo K-NN para identificar (para el conjunto de datos de prueba, con base en el conjunto de entrenamiento) cierta k de vecinos más cercanos, después de encontrar los vecinos más cercanos, su correspondiente set de items por usuario es agregado para encontrar un conjunto de items(películas) C junto con su frecuencia, con ese conjunto utilizar recomendaciones de usuario basadas en técnicas colaborativas se emplean para recomendar el top N más usuarios frecuentes en C que el usuario no halla incluído.

Para generar ésta clasificación se utilizará la paquetería **surprise**, ya que tiene varios métodos de similaridad de preferencias de usuario ya implementadas, para poder utilizarla es necesario instalar Visual C++ Data Tools y después ejecutar desde la consola de Anaconda lo siguiente:

pip install surprise

Una vez instalado exitosamente se utilizarán una matriz de búsqueda de cross-validation **GridSearchCV** para optimizar los parámetros de un modelo **KNNBasic**.

Se generará la predicción y la optimización de parámetros en el archivo **recommend_movies_2.py**.

Podemos ver que en la línea 1 se cargan los datos y en la línea 2 y 3 se entrena un modelo predefinido de KNNBasic y se hace la predicción del usuario 274 sobre la película 7930.

In [2]:
```
# Train on a whole trainset and the predict() method
from surprise import KNNBasic


# Build an algorithm, and train it.
algo = KNNBasic()
algo.fit(trainset)
```

```
Computing the msd similarity matrix...
Done computing similarity matrix.
```

Out[2]:
```
<surprise.prediction_algorithms.knns.KNNBasic at 0x223020c7ef0>
```

In [3]:
```
uid = str(274)  # raw user id (as in the ratings file). They are **strings**!
iid = str(7930)  # raw item id (as in the ratings file). They are **strings**!

# get a prediction for specific users and items.
pred = algo.predict(uid, iid, r_ui=4, verbose=True)
```

```
user: 274        item: 7930       r_ui = 4.00   est = 3.53   {'was_impossible': True, 'reason': 'User and/or item is unknown.'}
```

La predicción es 3.53 y sí existe una calificación dada en el conjunto de datos 4.0.

Ahora, para la elección de parámetros se buscará por medio de la **GridSearchCV** los parámetros óptimos en la parrilla tridimensional con **n_epochs** $\in \{5, 6, 7, 8, 9, 10\}$, **lr_all** $\in \{0.002, 0.003, 0.004, 0.005\}$

y **reg_all** $\in \{0.4, 0.5, 0.6\}$

In [6]:
```
from surprise.model_selection import GridSearchCV

data = Dataset.load_builtin('ml-100k')

param_grid = {'n_epochs': [5, 10], 'lr_all': [0.002, 0.005],
              'reg_all': [0.4, 0.6]}

gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=3)

gs.fit(data)

# best RMSE score
print(gs.best_score['rmse'])

# combination of parameters that gave the best RMSE score
print(gs.best_params['rmse'])
```

```
0.9643219966480064
{'n_epochs': 10, 'lr_all': 0.005, 'reg_all': 0.4}
```

In [7]:
```
results_df = pd.DataFrame.from_dict(gs.cv_results)
results_df
```

Out[7]:

	split0_test_rmse	split1_test_rmse	split2_test_rmse	mean_test_rmse	std_test_rmse	rank_test_rmse	split0_test_mae	split1_test_mae	split2_test_mae	mean
0	0.998202	0.996112	0.997912	0.997409	0.000925	7	0.806842	0.806375	0.805391	
1	1.004924	1.001814	1.003870	1.003536	0.001291	8	0.815838	0.814796	0.813799	

Conclusiones

- Hay dos maneras de abordar los problemas de recomendación usuario-artículo, basada en usuarios, basada en artículos o modelos híbridos de usuario-artículo pero son más complejos y tienen limitación cuando se escala a grandes volúmenes de datos.

  Éste aumento de complejidad se debe principalmente al cálculo de similaridad ya sea por cosenos o algún otro método entre las preferencias de cada usuario con todos los demás usuarios dados.

- En el caso del conjunto de datos dado, los parámetros óptimos para predecir la preferencia de películas con base en la calificación de usuarios dada son

  - **n_epochs** = 10
  - **lr_all** = 0.005 y
  - **reg_all** = 0.4

Referencias bibliográficas

1. **Conjunto de datos:**

   F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872

2. **A Survey of Collaborative Filtering Tecnhiques**

   Xiaoyuan Su and Taghi M. Khoshgoftaar 2009, *Department of Computer Science and Engine*

3. **Documentación de surprise**

   https://surprise.readthedocs.io/en/stable/knn_inspired.htmlhttps://surprise.readthedocs.io/en/stable/knn_inspired.html

   https://surprise.readthedocs.io/en/stable/getting_started.html