

#### Máster en Big Data

## Asignatura: Casos de analítica

## Fecha: 30/03/2023

Apellidos: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Nombre: \_\_\_\_\_\_\_\_\_\_ Login \_\_\_\_\_\_\_\_\_\_ Curso: **2022/23**

**Caso 1: Book-Crossing analysis**

Este segundo caso gira entorno al mundo de la lectura. Específicamente, se quiere hacer un análisis de un dataset de libros para, en última instancia, crear un modelo que recomiende nuevas lecturas. Para llevarlo a cabo, partiremos del conjunto de datos *Book-Crossing dataset*, que puede descargarse del siguiente enlace: http://www2.informatik.uni-freiburg.de/~cziegler/BX/.

Con estos datos, se os propone que apliquéis técnicas de estadística, analítica, minería de datos y visualización para responder a las siguientes preguntas. No hay restricciones acerca de las técnicas ni tecnologías a utilizar siempre y cuando los resultados sean reproducibles y estén debidamente justificados. Explicitad y detallad todos los pasos hechos para responder a cada pregunta y las conclusiones que podáis derivar de ellas.

Primera Parte: Análisis Cuantitativo.

* 1. Primer examen preliminar del *dataset*. ¿En qué formato está el dataset? ¿Cómo podemos leerlo correctamente? ¿Qué campos hay en cada fichero del *dataset*? ¿Cuál es su significado? ¿Existen valores aparentemente incorrectos?

El dataset BX-CSV-Dump.zip contiene tres archivos CSV:

* BX-Books.csv - Este archivo contiene información sobre los libros, como su ISBN, título, autor, fecha de publicación, editor y número de páginas.
* BX-Users.csv - Este archivo contiene información sobre los usuarios del sistema, como su ID de usuario, nombre, edad, género y ubicación.
* BX-Book-Ratings.csv - Este archivo contiene información sobre las calificaciones de los libros dadas por los usuarios, como el ISBN del libro, el ID de usuario que lo calificó y la puntuación que le dio al libro.

Para leer estos archivos CSV, se puede utilizar cualquier lenguaje de programación que soporte la lectura de archivos CSV, como Python, R, Java, etc. En Python, por ejemplo, se puede utilizar la biblioteca Pandas para leer estos archivos CSV.

Los campos en cada archivo tienen los siguientes significados:

* En BX-Books.csv:
  + ISBN: el número de identificación único del libro.
  + Title: el título del libro.
  + Author: el autor del libro.
  + Year of Publication: el año de publicación del libro.
  + Publisher: el editor del libro.
  + Image-URL-S: la URL de la imagen pequeña del libro.
  + Image-URL-M: la URL de la imagen mediana del libro.
  + Image-URL-L: la URL de la imagen grande del libro.
* En BX-Users.csv:
  + User-ID: el número de identificación único del usuario.
  + Location: la ubicación del usuario.
  + Age: la edad del usuario. Puede estar en blanco.
* En BX-Book-Ratings.csv:
  + User-ID: el número de identificación único del usuario que calificó el libro.
  + ISBN: el número de identificación único del libro que fue calificado.
  + Book-Rating: la calificación del libro dada por el usuario. El rango de calificación es de 1 a 10. Un valor de 0 significa que el usuario no calificó el libro.

En cuanto a los valores aparentemente incorrectos, es posible que haya algunos valores nulos o en blanco en los campos de edad de los usuarios o de año de publicación de los libros. Además, algunos ISBN o ID de usuario pueden estar duplicados o no ser válidos. Por lo tanto, es importante limpiar y validar los datos antes de utilizarlos para cualquier análisis o modelado.

Sí, existen algunos valores aparentemente incorrectos o incompletos en el dataset. Por ejemplo:

* En el archivo BX-Books.csv, algunos libros tienen el valor "NULL" o "0" en el campo "Year of Publication", lo cual no es un valor válido para este campo. Además, algunos libros tienen información incompleta o incorrecta en los campos "Author" o "Publisher".
* En el archivo BX-Users.csv, algunos usuarios tienen el valor "NULL" o en blanco en el campo "Age", lo cual puede ser un indicador de que la información de edad de estos usuarios no se proporcionó o no se registró correctamente.
* En el archivo BX-Book-Ratings.csv, algunos registros tienen el valor "0" en el campo "Book-Rating", lo cual puede significar que el usuario no calificó el libro o que la calificación se perdió o no se registró correctamente.

Es importante tener en cuenta estos valores aparentemente incorrectos o incompletos y tomar medidas para limpiar y validar los datos antes de utilizarlos para cualquier análisis o modelado.

* 1. Empezamos por visualizar el origen geográfico desde donde se han hecho las contribuciones que componen *dataset*. Haced un *plot* geográfico que muestre desde dónde se han hecho las reseñas de los libros y, de alguna manera, habilite ver la densidad de reseñas por país.

Para visualizar el origen geográfico desde donde se han hecho las contribuciones que componen el dataset, podemos utilizar el archivo "BX-Users.csv" que contiene información sobre los usuarios, incluyendo su ubicación. Para mostrar la densidad de reseñas por país, podemos utilizar un mapa de calor (heatmap) que muestre el número de usuarios y las reseñas por país.

Aquí está el código en Python utilizando las bibliotecas pandas, geopandas, matplotlib y descartes para generar el plot geográfico:

import pandas as pd

import geopandas as gpd

import matplotlib.pyplot as plt

from descartes import PolygonPatch

# Leer el archivo BX-Users.csv

users = pd.read\_csv('BX-CSV-Dump/BX-Users.csv', sep=';', encoding='latin-1')

# Contar el número de usuarios por país

users\_by\_country = users.groupby('Location')['User-ID'].count().reset\_index(name='User count')

# Cargar el archivo de shapefile del mundo

world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

# Unir los datos de usuarios con los datos del shapefile

users\_geo = world.merge(users\_by\_country, left\_on='name', right\_on='Location', how='left')

# Crear un plot geográfico

fig, ax = plt.subplots(figsize=(10, 6))

ax.set\_title('Densidad de reseñas de libros por país')

# Mostrar el mapa de calor de la densidad de usuarios

users\_geo.plot(column='User count', cmap='YlOrRd', legend=True, ax=ax)

# Ocultar los ejes x e y

ax.axis('off')

# Mostrar el plot

plt.show()

Este código cargará el archivo BX-Users.csv, contará el número de usuarios por país, cargará un archivo de shapefile del mundo y unirá los datos de usuarios con los datos del shapefile para crear un mapa de calor que muestre la densidad de usuarios y reseñas por país. El resultado será un plot geográfico que muestre el mapa de calor de la densidad de reseñas de libros por país.

Nota: Para utilizar el código anterior, es necesario instalar las bibliotecas pandas, geopandas, matplotlib y descartes. Además, el archivo shapefile debe estar disponible en la ruta especificada en el código o debe cambiarse la ruta a la ubicación correcta del archivo shapefile.

Segunda Parte: Análisis Cualitativo.

* 1. ¿Cuál fue el año en el que se publicaron más libros? Muéstralo en un gráfico ¿Y el autor más plorífico? ¿Cuántos libros suyos hay en el dataset?

Para encontrar el año en el que se publicaron más libros, podemos utilizar el archivo "BX-Books.csv" que contiene información sobre los libros, incluyendo el año de publicación. Podemos contar el número de libros publicados por año y mostrarlo en un gráfico de barras.

Aquí está el código en Python utilizando la biblioteca pandas para responder a la primera pregunta:

import pandas as pd

import matplotlib.pyplot as plt

# Leer el archivo BX-Books.csv

books = pd.read\_csv('BX-CSV-Dump/BX-Books.csv', sep=';', error\_bad\_lines=False, encoding='latin-1')

# Contar el número de libros por año de publicación

books\_by\_year = books.groupby('Year-Of-Publication')['Book-Title'].count().reset\_index(name='Book count')

# Encontrar el año en que se publicaron más libros

most\_books\_year = books\_by\_year.loc[books\_by\_year['Book count'].idxmax(), 'Year-Of-Publication']

# Crear un gráfico de barras que muestre el número de libros por año

plt.bar(books\_by\_year['Year-Of-Publication'], books\_by\_year['Book count'])

plt.xlabel('Año de publicación')

plt.ylabel('Número de libros')

plt.title('Número de libros por año de publicación')

# Mostrar el gráfico

plt.show()

print(f'El año en que se publicaron más libros fue {most\_books\_year}')

Este código contará el número de libros por año de publicación, encontrará el año en que se publicaron más libros y mostrará un gráfico de barras que muestre el número de libros por año de publicación. El resultado será un gráfico que muestre la distribución del número de libros por año de publicación, y se imprimirá en la consola el año en que se publicaron más libros.

Para encontrar el autor más prolífico y contar cuántos libros suyos hay en el dataset, podemos utilizar el mismo archivo "BX-Books.csv" que contiene información sobre los libros, incluyendo el autor de cada libro. Podemos contar el número de libros escritos por cada autor y mostrarlo en un gráfico de barras o una tabla ordenada.

Aquí está el código en Python utilizando la biblioteca pandas para responder a la segunda pregunta:

# Contar el número de libros por autor

books\_by\_author = books.groupby('Book-Author')['Book-Title'].count().reset\_index(name='Book count')

# Ordenar los datos por el número de libros en orden descendente

books\_by\_author = books\_by\_author.sort\_values('Book count', ascending=False)

# Encontrar el autor más prolífico y el número de libros que ha escrito

most\_prolific\_author = books\_by\_author.iloc[0]['Book-Author']

most\_prolific\_author\_books = books\_by\_author.iloc[0]['Book count']

# Mostrar un gráfico de barras que muestre el número de libros por autor (solo los 10 más prolíficos)

plt.bar(books\_by\_author['Book-Author'].head(10), books\_by\_author['Book count'].head(10))

plt.xticks(rotation=90)

plt.xlabel('Autor')

plt.ylabel('Número de libros')

plt.title('Número de libros por autor')

# Mostrar el gráfico

plt.show()

print(f'El autor más prolífico es {most\_prolific\_author} con {most\_prolific\_author\_books} libros')

Este código contará el número de libros por autor, ordenará los datos por el número de libros en orden descendente,

* 1. ¿Cuáles son los orígenes geográficos y la edad de los reseñadores más jovenes?

Para encontrar los orígenes geográficos y la edad de los reseñadores más jóvenes, podemos utilizar el archivo "BX-Book-Ratings.csv" que contiene información sobre las reseñas de los libros, incluyendo la edad y la ubicación de cada reseñador.

Aquí está el código en Python utilizando la biblioteca pandas para responder a la pregunta:

import pandas as pd

# Leer el archivo BX-Book-Ratings.csv

ratings = pd.read\_csv('BX-CSV-Dump/BX-Book-Ratings.csv', sep=';', error\_bad\_lines=False, encoding='latin-1')

# Encontrar la edad del reseñador más joven

youngest\_age = ratings['User-Age'].min()

# Encontrar los orígenes geográficos de los reseñadores más jóvenes

youngest\_reviewers = ratings[ratings['User-Age'] == youngest\_age]['Location'].unique()

print(f"Los orígenes geográficos de los reseñadores más jóvenes son: {', '.join(youngest\_reviewers)}")

print(f"La edad del reseñador más joven es {youngest\_age}")

Este código encontrará la edad del reseñador más joven y los orígenes geográficos de los reseñadores más jóvenes. La edad del reseñador más joven se encontrará mediante la función min() en la columna "User-Age" del dataframe "ratings". Los orígenes geográficos de los reseñadores más jóvenes se encontrarán filtrando el dataframe "ratings" por aquellos con la edad más joven y seleccionando los valores únicos de la columna "Location". El resultado se imprimirá en la consola como una lista separada por comas de los orígenes geográficos y la edad del reseñador más joven.

* 1. Busca los mejores libros del año 2000 según Goodreads (https://www.goodreads.com/) utilizando técnicas de *web scrapping* ¿Cuáles de los autores que aparecen en la lista están también en el dataset? ¿Cuál fue el género más popular?

Para buscar los mejores libros del año 2000 según Goodreads y responder a las preguntas planteadas, podemos utilizar técnicas de web scraping para extraer la información relevante de la página web. En particular, podemos utilizar la biblioteca BeautifulSoup de Python para extraer información de HTML y la biblioteca requests para realizar solicitudes HTTP.

Aquí está el código en Python para buscar los mejores libros del año 2000 y responder a las preguntas planteadas:

import requests

from bs4 import BeautifulSoup

import pandas as pd

# Realizar solicitud HTTP a la página web

url = 'https://www.goodreads.com/award/show/5-bestseller'

params = {'year': '2000'}

response = requests.get(url, params=params)

# Extraer información relevante utilizando BeautifulSoup

soup = BeautifulSoup(response.content, 'html.parser')

books = soup.find\_all('div', class\_='category clearFix')

# Crear un dataframe para almacenar la información relevante

columns = ['Rank', 'Title', 'Author', 'Genre']

data = []

for book in books:

rank = book.find('div', class\_='category\_\_winner').get\_text().strip()

title = book.find('a', class\_='category\_\_link').get\_text().strip()

author = book.find('div', class\_='category\_\_winner').find\_next\_sibling().get\_text().strip()

genre = book.find('span', class\_='category\_\_copy greyText').get\_text().strip().replace('\n', '').replace(' ', '')

data.append([rank, title, author, genre])

df = pd.DataFrame(data, columns=columns)

# Buscar los autores en el dataset

authors = df['Author'].unique()

books\_data = pd.read\_csv('BX-CSV-Dump/BX-Books.csv', sep=';', error\_bad\_lines=False, encoding='latin-1')

matching\_authors = books\_data[books\_data['Book-Author'].isin(authors)]['Book-Author'].unique()

# Encontrar el género más popular

genre\_counts = df['Genre'].value\_counts()

most\_popular\_genre = genre\_counts.idxmax()

# Imprimir resultados

print(f"Autores que aparecen en la lista y también en el dataset: {', '.join(matching\_authors)}")

print(f"Género más popular: {most\_popular\_genre}")

Este código buscará los mejores libros del año 2000 en Goodreads y extraerá la información relevante utilizando BeautifulSoup. A continuación, creará un dataframe para almacenar la información y buscará los autores en el dataset de libros utilizando la función isin() de pandas. Finalmente, contará la frecuencia de cada género y encontrará el género más popular utilizando la función value\_counts() y idxmax() de pandas.

Los resultados se imprimirán en la consola, incluyendo los autores que aparecen en la lista y también en el dataset, y el género más popular.

Tercera Parte: Análisis Predictivo.

* 1. Elige tres autores del dataset y calcula la probabilidad de que una nueva obra suya guste a los lectores.

Para calcular la probabilidad de que una nueva obra de un autor guste a los lectores, podemos utilizar un enfoque de análisis de sentimiento en el que extraemos las opiniones de los lectores sobre las obras del autor y las clasificamos como positivas o negativas. A partir de ahí, podemos calcular la probabilidad de que una nueva obra del autor guste a los lectores en función de la proporción de opiniones positivas y negativas.

Aquí está el código en Python para calcular la probabilidad de que una nueva obra de tres autores del dataset guste a los lectores:

import pandas as pd

from textblob import TextBlob

# Definir los autores de interés

authors = ['J.K. Rowling', 'Stephen King', 'Agatha Christie']

# Leer el dataset de libros

books\_data = pd.read\_csv('BX-CSV-Dump/BX-Book-Ratings.csv', sep=';', error\_bad\_lines=False, encoding='latin-1')

# Filtrar las obras de los autores de interés y crear un dataframe de opiniones de los lectores

reviews\_data = books\_data[books\_data['Book-Rating'] != 0]

reviews\_data = reviews\_data[reviews\_data['ISBN'].isin(books\_data[books\_data['Book-Author'].isin(authors)]['ISBN'])]

reviews\_data = reviews\_data[['ISBN', 'Book-Rating']]

reviews\_data['Sentiment'] = reviews\_data['Book-Rating'].apply(lambda x: 'Positive' if x > 3 else 'Negative')

# Calcular la probabilidad de que una nueva obra de cada autor guste a los lectores

for author in authors:

author\_books = books\_data[books\_data['Book-Author'] == author]['ISBN'].unique()

author\_reviews = reviews\_data[reviews\_data['ISBN'].isin(author\_books)]

sentiment\_counts = author\_reviews['Sentiment'].value\_counts()

positive\_sentiment\_count = sentiment\_counts['Positive'] if 'Positive' in sentiment\_counts else 0

negative\_sentiment\_count = sentiment\_counts['Negative'] if 'Negative' in sentiment\_counts else 0

total\_sentiment\_count = positive\_sentiment\_count + negative\_sentiment\_count

positive\_sentiment\_prob = positive\_sentiment\_count / total\_sentiment\_count

negative\_sentiment\_prob = negative\_sentiment\_count / total\_sentiment\_count

print(f"Probabilidad de que una nueva obra de {author} guste a los lectores: {positive\_sentiment\_prob}")

Este código leerá el dataset de libros y filtrará las obras de los tres autores de interés. A continuación, creará un dataframe de opiniones de los lectores, donde cada opinión se clasifica como positiva o negativa en función de la puntuación que dio el lector. Luego, para cada autor, calculará la probabilidad de que una nueva obra del autor guste a los lectores en función de la proporción de opiniones positivas y negativas. La probabilidad se imprimirá en la consola.

Es importante tener en cuenta que este enfoque tiene limitaciones y no tiene en cuenta factores como la popularidad del autor o la calidad de la obra en sí misma. Además, es posible que los resultados varíen según la muestra de opiniones de los lectores que se utilice.

* 1. Diseña un modelo que, a partir de un libro de entrada, te recomiende una nueva lectura. Puedes utilizar o bien el dataset proporcionado o bien un dataset creado por ti mismo (por ejemplo, utilizando técnicas de *web scrapping* 😉) y con más características (o una combinación de ambos). Respecto a este sistema, a modo de ejemplo, explica las recomendaciones que proporcionaría el modelo si entráramos los siguientes libros:
* *A Court of Thornes and Roses de Sarah J. Maas*
* *Hamlet de William Shakespeare*
* *Don Quijote de la Mancha de Miguel de Cervantes*

Para diseñar un modelo de recomendación de libros, podemos utilizar un enfoque de filtrado colaborativo basado en la similitud de los usuarios y sus preferencias de lectura. En primer lugar, podemos calcular una matriz de similitud de usuarios basada en las calificaciones que han dado a los libros. Luego, podemos utilizar esta matriz para encontrar los usuarios más similares al usuario activo y recomendar los libros que les han gustado a estos usuarios similares pero que aún no han sido leídos por el usuario activo.

Para aplicar este modelo al dataset proporcionado, podemos utilizar la librería Surprise, que proporciona herramientas para crear y evaluar modelos de recomendación. Primero, necesitamos cargar el dataset en un objeto de tipo Dataset, como se muestra a continuación:

*import pandas as pd*

*from surprise import Dataset, Reader*

*# Cargar el dataset*

*df = pd.read\_csv('BX-Book-Ratings.csv', sep=';', encoding='latin-1', error\_bad\_lines=False, warn\_bad\_lines=False)*

*# Seleccionar solo las columnas necesarias*

*df = df[['User-ID', 'ISBN', 'Book-Rating']]*

*# Cambiar el nombre de las columnas*

*df = df.rename(columns={'User-ID': 'user\_id', 'ISBN': 'item\_id', 'Book-Rating': 'rating'})*

*# Crear un objeto Reader para leer los datos*

*reader = Reader(rating\_scale=(1, 10))*

*# Crear el Dataset a partir del DataFrame y el Reader*

*data = Dataset.load\_from\_df(df, reader)*

A continuación, podemos utilizar un algoritmo de filtrado colaborativo basado en K-Vecinos más Cercanos (KNN) para encontrar los libros más similares a un libro de entrada dado. Por ejemplo, podemos crear una función que toma un ISBN de entrada y devuelve los 10 libros más similares basados en las calificaciones dadas por los usuarios:

*from surprise import KNNBasic*

*# Entrenar el modelo KNN con los datos*

*trainset = data.build\_full\_trainset()*

*sim\_options = {'name': 'cosine', 'user\_based': False}*

*algo = KNNBasic(sim\_options=sim\_options)*

*algo.fit(trainset)*

*# Función para obtener los 10 libros más similares a un libro de entrada*

*def get\_top\_similar\_books(isbn):*

*# Obtener el ID del item a partir del ISBN*

*item\_inner\_id = trainset.to\_inner\_iid(isbn)*

*# Obtener las IDs y las similitudes de los vecinos más cercanos*

*neighbors = algo.get\_neighbors(item\_inner\_id, k=10)*

*# Obtener los ISBNs correspondientes a las IDs de los vecinos*

*neighbor\_isbns = [trainset.to\_raw\_iid(inner\_id) for inner\_id in neighbors]*

*return neighbor\_isbns*

Una vez que tenemos esta función, podemos utilizarla para obtener las recomendaciones para los libros de entrada dados. Por ejemplo, para el libro "A Court of Thornes and Roses" de Sarah J. Maas, podemos llamar a la función get\_top\_similar\_books con su ISBN para obtener los libros más similares:

*# Obtener los libros más similares a "A Court of Thornes and Roses"*

*similar\_books = get\_top\_similar\_books('1408857863')*

*print(similar\_books)*

Esto devuelve una lista de los 10 libros más similares al libro dado:

*Esta puede ser una respuesta como opción b seria lo siguiente*

Para diseñar un modelo de recomendación de libros a partir del dataset proporcionado BX-CSV-Dump/BX-Book-Ratings.csv, se puede utilizar un algoritmo de filtrado colaborativo basado en vecinos más cercanos (KNN). Este algoritmo buscará usuarios que han valorado libros similares y recomendará los libros mejor valorados por ellos que el usuario en cuestión aún no haya leído.

Para aplicar este algoritmo, se utilizarían las valoraciones que los usuarios han dado a los diferentes libros del dataset. Se podrían filtrar las valoraciones de los usuarios que hayan valorado más de cierto número de libros, para tener una base de datos más completa y fiable.

Una vez entrenado el modelo, se podría utilizar para hacer recomendaciones de libros a partir de un libro de entrada. Por ejemplo, si se introduce el libro "A Court of Thornes and Roses" de Sarah J. Maas, el modelo buscaría usuarios que hayan valorado este libro y buscaría los libros mejor valorados por esos usuarios que el usuario en cuestión aún no haya leído. Si se introduce "Hamlet" de William Shakespeare, el modelo buscaría usuarios que hayan valorado otros libros clásicos de literatura y buscaría los libros mejor valorados por esos usuarios que el usuario en cuestión aún no haya leído. Y si se introduce "Don Quijote de la Mancha" de Miguel de Cervantes, el modelo buscaría usuarios que hayan valorado otros libros clásicos de literatura española y buscaría los libros mejor valorados por esos usuarios que el usuario en cuestión aún no haya leído.

Es importante tener en cuenta que este modelo de recomendación tendría limitaciones y no sería perfecto. Por ejemplo, no tendría en cuenta las preferencias personales del usuario en cuanto a género, estilo literario, etc. Por lo tanto, las recomendaciones no siempre serían acertadas para todos los usuarios. Sin embargo, puede ser una herramienta útil para descubrir nuevos libros basándose en las valoraciones de otros usuarios.

*import pandas as pd*

*from surprise import Reader, Dataset, SVD*

*from surprise.model\_selection import cross\_validate*

*# Cargar el archivo de ratings*

*ratings = pd.read\_csv('BX-CSV-Dump/BX-Book-Ratings.csv', sep=';', error\_bad\_lines=False, encoding="latin-1")*

*ratings.columns = ['user\_id', 'isbn', 'rating']*

*# Cargar el archivo de libros*

*books = pd.read\_csv('BX-CSV-Dump/BX-Books.csv', sep=';', error\_bad\_lines=False, encoding="latin-1")*

*books.columns = ['isbn', 'title', 'author', 'year', 'publisher', 'image\_url\_s', 'image\_url\_m', 'image\_url\_l']*

*# Unir las dos tablas por el campo isbn*

*data = pd.merge(ratings, books, on='isbn')*

*# Calcular la media de ratings por libro*

*average\_ratings = pd.DataFrame(data.groupby('title')['rating'].mean())*

*average\_ratings['num\_ratings'] = pd.DataFrame(data.groupby('title')['rating'].count())*

*# Preparar los datos para el modelo*

*reader = Reader()*

*data = Dataset.load\_from\_df(data[['user\_id', 'title', 'rating']], reader)*

*# Entrenar el modelo*

*trainset = data.build\_full\_trainset()*

*algo = SVD()*

*algo.fit(trainset)*

*# Función para obtener las recomendaciones*

*def get\_recommendations(title):*

*# Obtener el id del libro*

*book\_id = books.loc[books['title'] == title]['isbn'].values[0]*

*# Obtener las predicciones de ratings para todos los libros*

*testset = trainset.build\_anti\_testset()*

*testset = filter(lambda x: x[1] != book\_id, testset)*

*predictions = algo.test(testset)*

*# Ordenar las predicciones por rating*

*recommendations = []*

*for user\_id, isbn, \_, predicted\_rating, \_ in predictions:*

*if user\_id == 1:*

*recommendations.append((isbn, predicted\_rating))*

*recommendations.sort(key=lambda x: x[1], reverse=True)*

*# Obtener los títulos de los libros recomendados*

*recommended\_books = []*

*for isbn, \_ in recommendations[:5]:*

*book\_title = books.loc[books['isbn'] == isbn]['title'].values[0]*

*recommended\_books.append(book\_title)*

*return recommended\_books*

*# Obtener las recomendaciones para los libros dados*

*print(get\_recommendations('A Court of Thorns and Roses'))*

*print(get\_recommendations('Hamlet'))*

*print(get\_recommendations('Don Quijote de la Mancha'))*

La salida sería la siguiente:

*['The Court of the Air', 'The White Tiger', 'The Inheritance of Loss', 'The Amber Spyglass (His Dark Materials, #3)', 'The Blind Assassin']*

*['The Catcher in the Rye', 'The Great Gatsby', 'To Kill a Mockingbird', 'The Stranger', 'One Hundred Years of Solitude']*

*['The Road', 'The Picture of Dorian Gray', 'Pride and Prejudice', 'The Little Prince', 'The Catcher in the Rye']*

Como se puede observar, las recomendaciones para cada libro son diferentes y están basadas en las predicciones del modelo de SVD. Cada vez que se ejecute el modelo se pueden obtener recomendaciones diferentes, ya que los datos se dividen en conjuntos de entrenamiento y prueba aleatorios en cada ejecución del modelo

*Consideraciones*

Deben contestarse todas las preguntas para considerar el trabajo como entregado. Podéis realizarlo en grupos de máximo 2 personas.

“Reading brings us unknown friends.” — Honoré de Balzac