I. Manejo de Datos

Para crear un sistema de recomendaciones con el dataset: Book-Crossing Dataset, podemos tomar diferentes acercamientos.

- User-based Collaborative Filtering
- Item-based Collaborative Filtering

Eventualmente, podemos juntar las predicciones de ambos acercamientos para crear un sistema híbrido.

1. Limpieza de tablas

a. BX-Users

Primero, observo el .csv para entender la naturaleza de los datos presentes.
El separador es ";" y no parece estar presente al interior de ninguna columna, por lo que no hay necesidad de escaparlo.

Importo la tabla por medio de la función *read_csv* de la librería Pandas, usando como separador ";", y usando *unicode_escape* como encoding, y llamo ***users*** al DataFrame.
Imprimo la forma de la tabla (con users.shape): tiene 278858 filas y 3 columnas.
Imprimo los tipos de datos (con users.dtypes): int64, object y float64 respectivamente.

La columna "User-ID" contiene enteros únicos, comenzando por 1, que se autoincrementan.
Elimino los valores fuera de [1,278858], si es que hay.
Elimino duplicados, si es que hay.
Cambio el índice de mi tabla a la columna "User-ID"
Ahora mi DataFrame *users* comienza en 1 y termina en 278858.

La columna "Location" tiene un string con la forma: "ciudad, estad o región, pais".
Aprovechamos este paso para resolver la pregunta (b.):
Los datos al interior de esta columna están separados por ",". Para esto, uso la función *split* de Pandas. Luego eliminamos las filas con países incorrectos, usando el regex: [^$|\"|\.|;|,]. Lo que quita líneas donde el país está vacío, tiene comillas, puntos, punto y comas, y comas. Se quitan 6796 líneas.
Añadimos las columnas y eliminamos la columna anterior ("Location") para no tener información duplicada.

La columna "Age" contiene la edad de cada usuario, que es numérico y puedo suponer que de dos cifras. El valor puede ser NULL. Podemos suponer que los usuarios tienen entre 12 y 100 años. Por lo que vamos a convertir en null los valores que están fuera de este rango.

Transformo el tipo de la columna de float a int, primero llenando los valores vacíos con 0, y usando *astype(int)* para convertir.

Ahora la información del DataFrame es más fácil de usar. A continuación, las primeras 10 líneas de la tabla, que imprimo con *head(10)*.

	Age	City	Region	Country
1	0	nyc	new york	usa
2	18	stockton	california	usa
3	0	moscow	yukon territory	russia
4	17	porto	v.n.gaia	portugal
5	0	farnborough	hants	united kingdom
6	61	santa monica	california	usa
7	0	washington	dc	usa
8	0	timmins	ontario	canada
9	0	germantown	tennessee	usa
10	26	albacete	wisconsin	spain

Primeras 10 líneas de DataFrame users.

b. BX-Book-Rating

Esta tabla tiene 3 columnas: “User-ID”, “ISBN”, “Book-Rating”. Ninguno tiene valores únicos, por lo que no vamos a reindexar la tabla por el momento. El separador sigue siendo “;”. Luego, quitamos líneas que duplicadas.
Al ver los datos, vemos que el ISBN tiene ciertos valores que hay que limpiar. Por ejemplo, la línea 21687 tiene “\”0210000010” como valor. Usamos la expresión regular /[^\w\d]/ para identificar caracteres que no sean números o letras, y los elimino.

Con el mismo proceso, creo un DataFrame para esta tabla y lo llamo ***ratings,*** codificado con *unicode_escape.*
Dimensiones: (1 149 780, 3)
Tipos: int64, object, int64

Los ratings van del 0 al 10, por lo que eliminamos las líneas que no cumplan eso.
El 0 representa los “likes” (rating implícito) y los valores del 1 al 10 representan un puntaje, siendo 10 el puntaje más alto.

c. BX-Books
Esta tabla tiene 7 columnas: “ISBN”, “Book-Title”, “Book-Author”, “Year-Of-Publication”,”Publisher”,”Image-URL-S”,” Image-URL-M”, ”Image-URL-L”
Como no voy a tratar con Front End ni a calcular similitudes entre imágenes, boto las tres últimas columnas.
Elimino líneas con ISBN duplicados, y guardo la primer aparición (ya que el ISBN debe ser único por cada libro).
Luego, limpio el ISBN de la misma manera que para *ratings.*
Por último, indexo la tabla por el ISBN, para recorrerla más fácilmente.

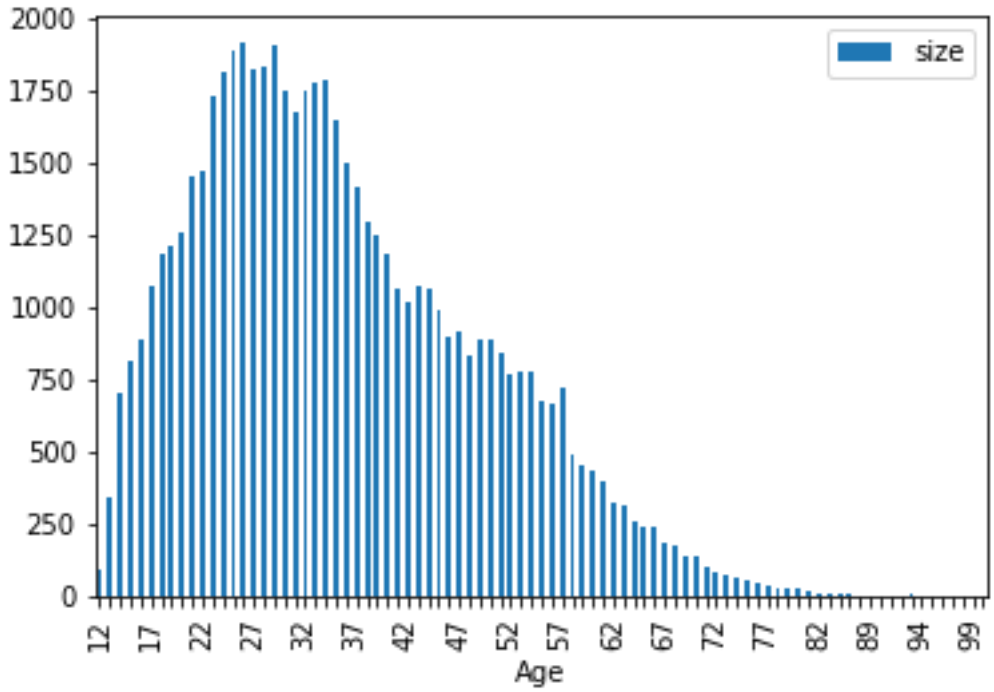
En cuanto a la columna “Year-Of-Publication”, el dataset fue creado en 2004, por lo que asigno el valor 0 al año de publicación si este es superior a 2004. De esta manera evito ruido en el cálculo de similitud.

d. Reducción de dimensión
Como vamos a crear un sistema de recomendaciones, basándonos en ratings, eliminamos usuarios que no hayan dejado ratings, y libros que no tengan.

Uso *isin()* para revisar si el ID del usuario en la tabla *users* está en la columna “User-ID” de la tabla *ratings*.
Se eliminan 173 575 usuarios.
Reviso si el ISBN del libro en la tabla *Books* está en la columna “ISBN” de la tabla *ratings.*
Se eliminan 118 269 libros.

2. Estadísticas del dataset

a. Repartición de usuarios por edad



Cuartiles:

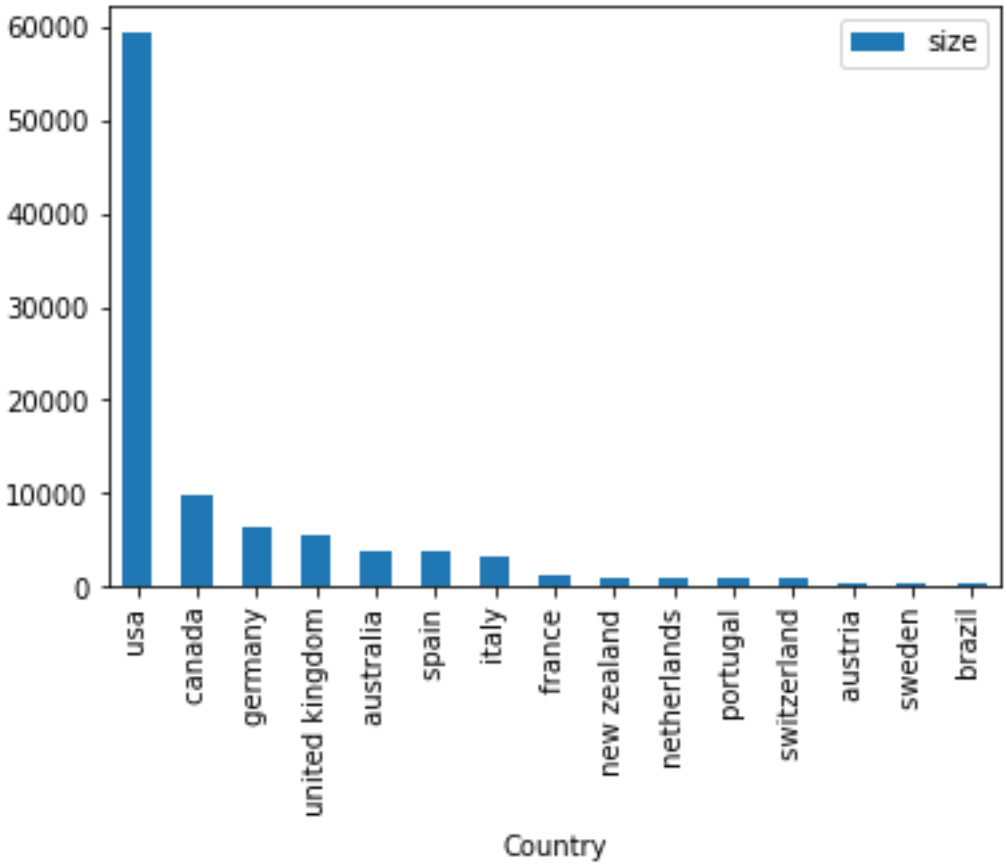
1	25 años
2	33 años
3	45 años

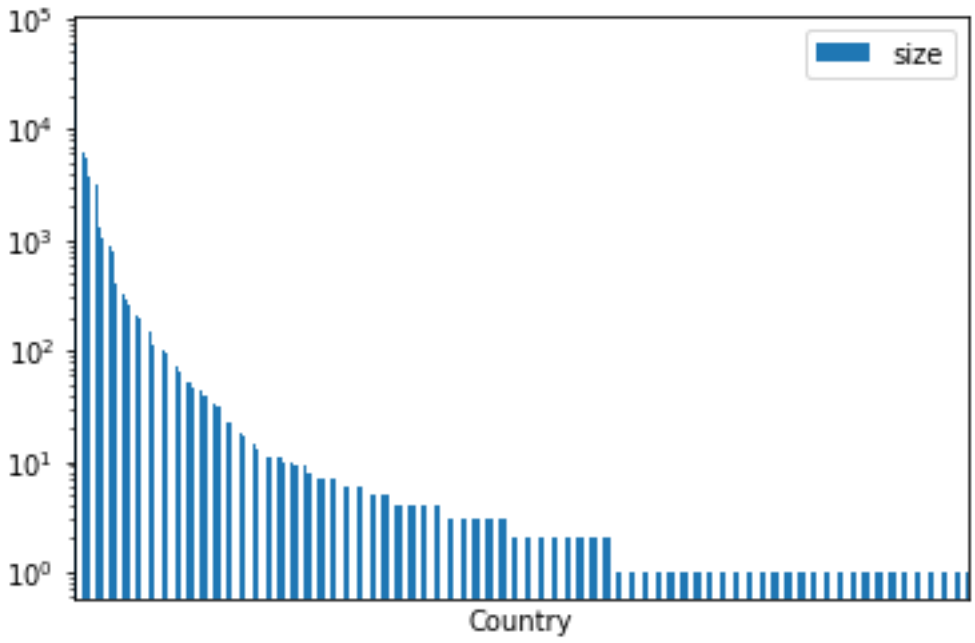
La edad promedio es de 35 años.

Moda: 26 años (1918 usuarios)

41 643 usuarios tienen una edad invalida (41% de los usuarios)

b. Usuarios por país

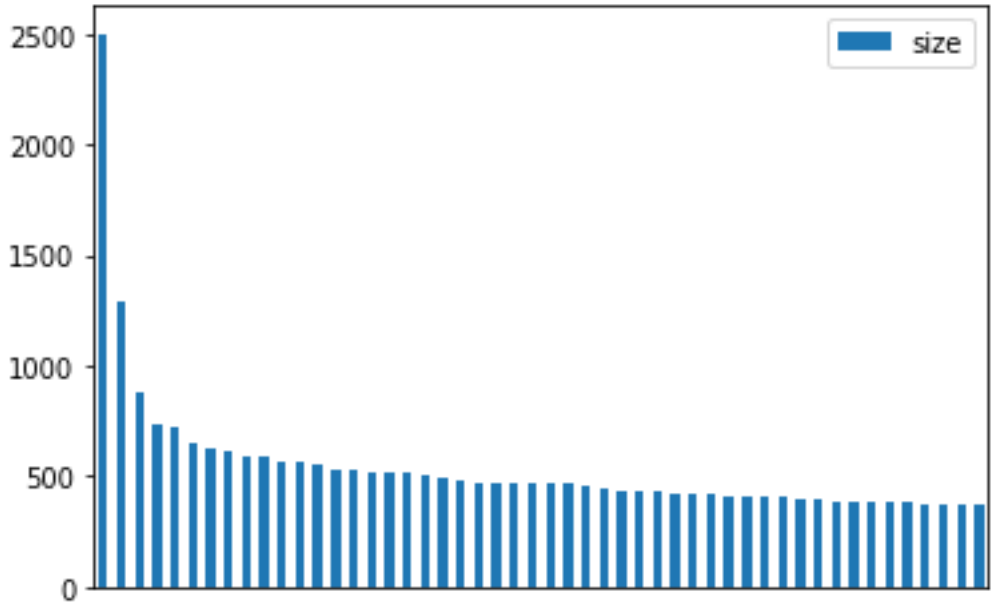




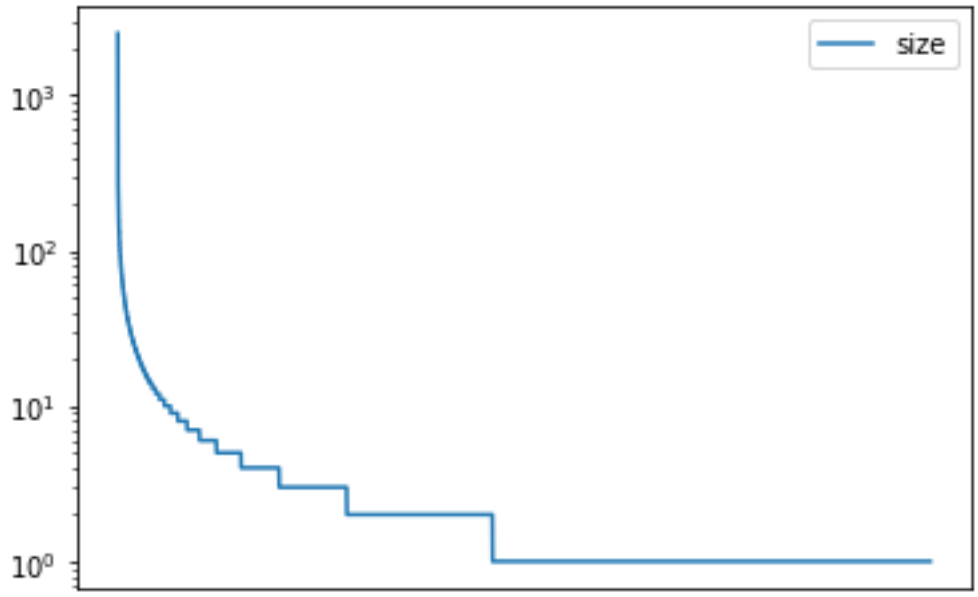
Usuarios por país, escala logarítmica.

59% de los usuarios viven en Estados Unidos.
75% de los usuarios viven en 3 países

c. Libros más populares



Libros



Libros

Libros más populares, escala logarítmica.

El libro con más ratings tiene 2502 ratings.
Promedio: 4 ratings
5% de los libros tienen más de 12 ratings.

Para conocer los libros con más ratings, agrupamos la tabla *ratings* por libros, y contamos cada grupo, los ordenamos por orden decreciente y tomamos los 10 primeros ISBN. Luego buscamos el ISBN en la tabla *books* para conocer las características de cada libro(ver fig. 3)

Top 10 libros con más ratings:

ISBN	Título	Autor
0971880107	Wild Animus	Rich Shapero
0316666343	The Lovely Bones: A Novel	Alice Sebold
0385504209	The Da Vinci Code	Dan Brown
0060928336	Divine Secrets of the Ya-Ya Sisterhood: A Novel	Rebecca Wells
0312195516	The Red Tent (Bestselling Backlist)	Anita Diamant
044023722X	A Painted House	John Grisham
0142001740	The Secret Life of Bees	Sue Monk Kidd
067976402X	Snow Falling on Cedars	David Guterson
0671027360	Angels & Demons	Dan Brown
0446672211	Where the Heart Is (Oprah's Book Clup (Paperback))	Billie Letts

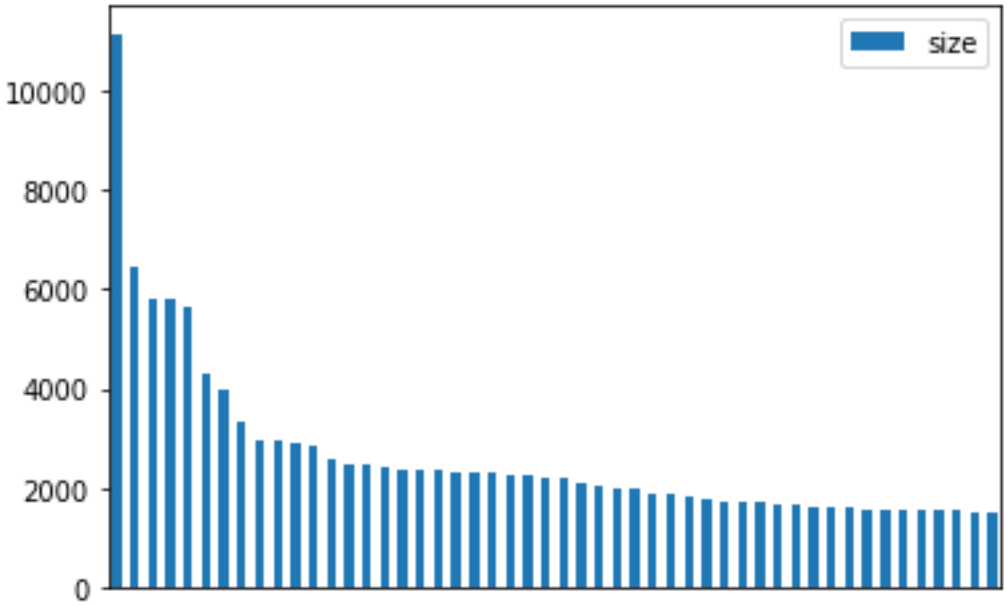
d. Libros con mejor rating

Para conocer los libros con mejores ratings, agrupamos *ratings* por "Book-Rating", seleccionamos el grupo de los que tienen una puntuación de 10, luego agrupamos este grupo por ISBN para eliminar los ratings repetidos por libro. Después, contamos el número de ratings por libro, dentro del grupo, y ordenamos en orden decreciente. Por último, leemos los nombres de los grupos adentro de este para obtener el ISBN. De esta manera, no solo obtenemos el top 10 mejores ratings, sino que el top 10 mejores ratings más relevantes.

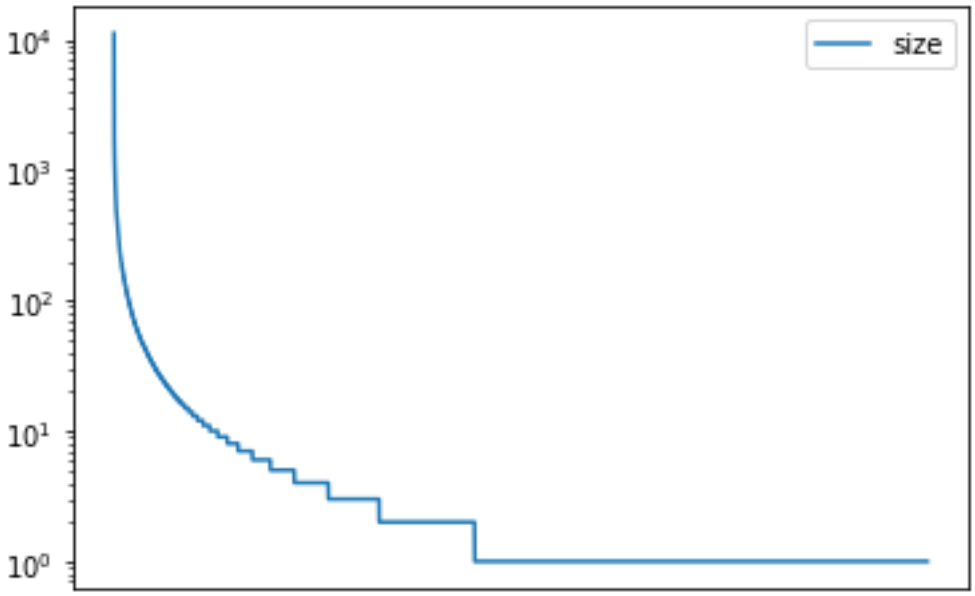
Top 10 libros con mejores ratings :

ISBN	Título	Autor
0385504209	The Da Vinci Code	Dan Brown
059035342X	Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))	J. K. Rowling
0316666343	The Lovely Bones: A Novel	Alice Sebold
043935806X	Harry Potter and the Order of the Phoenix (Book 5)	J. K. Rowling
0446310786	To Kill a Mockingbird	Harper Lee
0312195516	The Red Tent (Bestselling Backlist)	Anita Diamant
0142001740	The Secret Life of Bees	Sue Monk Kidd
0439139597	Harry Potter and the Goblet of Fire (Book 4)	J. K. Rowling
0439136350	Harry Potter and the Prisoner of Azkaban (Book 3)	J. K. Rowling
0385484518	Tuesdays with Morrie: An Old Man, a Young Man, and Life's Greatest Lesson	MITCH ALBOM

e. Cantidad de libros calificados por usuario



Usuarios

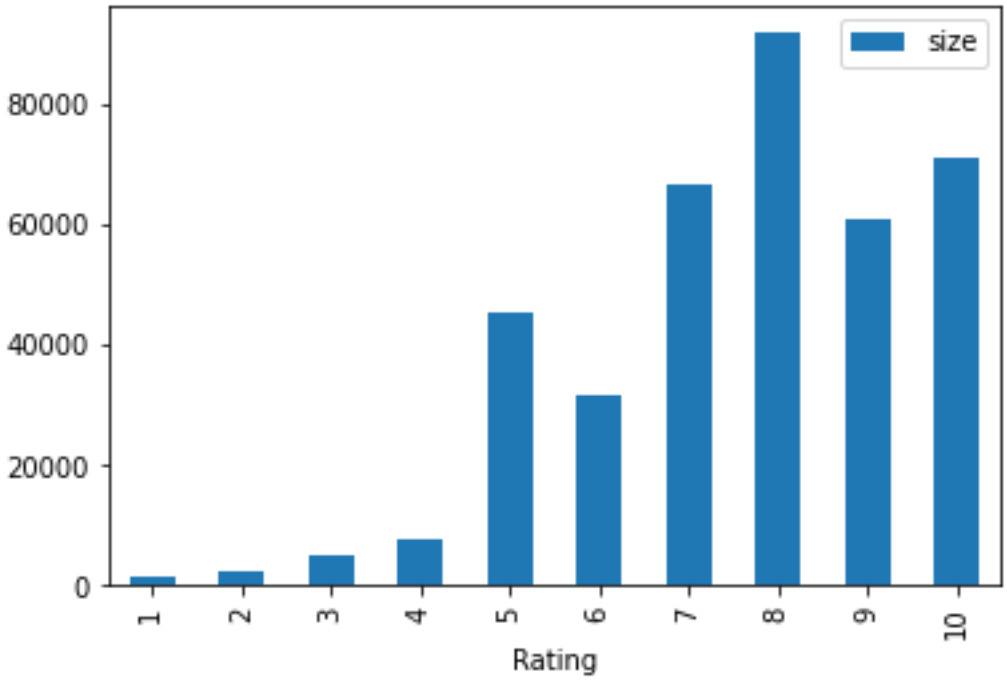


Usuarios

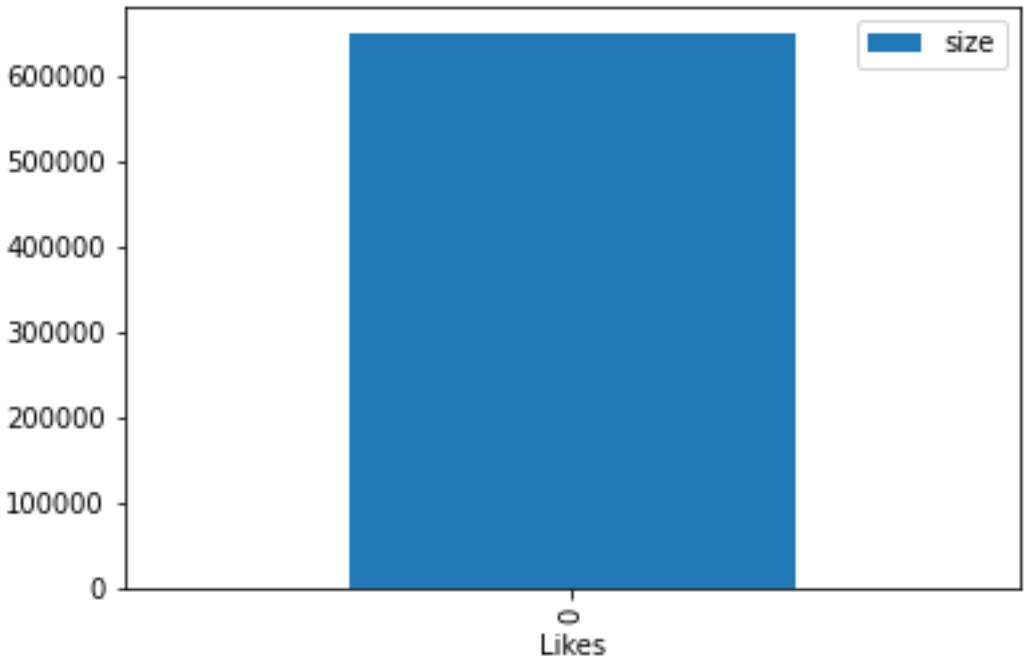
Cantidad de ratings por usuario, escala logarítmica.

Tercer cuartil: 4 ratings. 75% de los usuarios ha dejado menos de 4 ratings.
El usuario con mas libros calificados ha dejado un rating para 11153 libros.

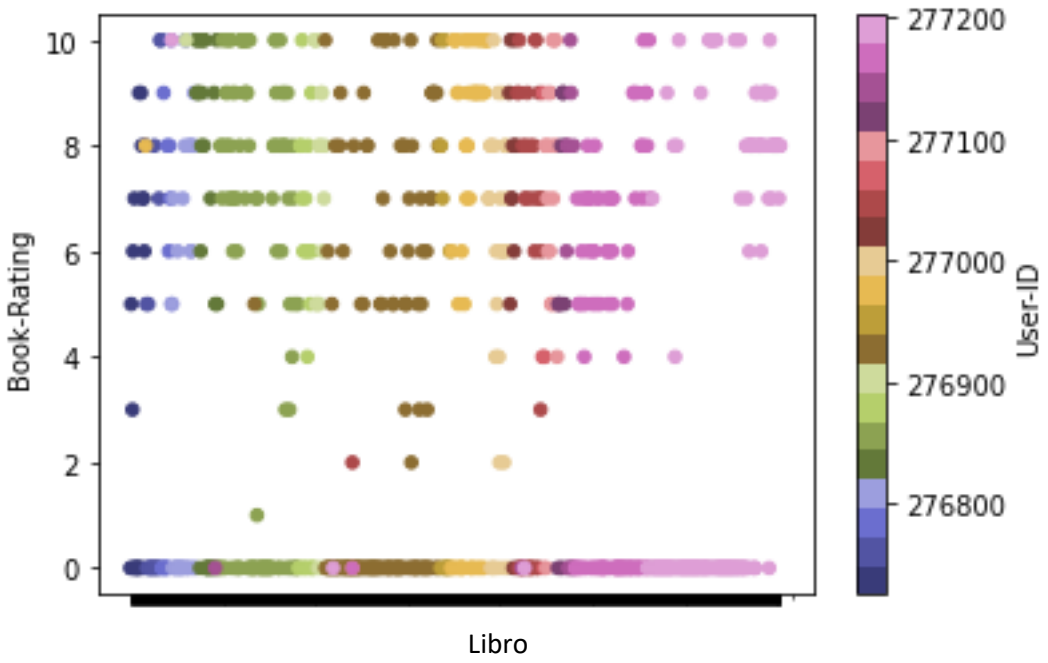
f. Usuarios por rating



Moda: 8/10
Promedio: 7.63/10 (sin contar likes)

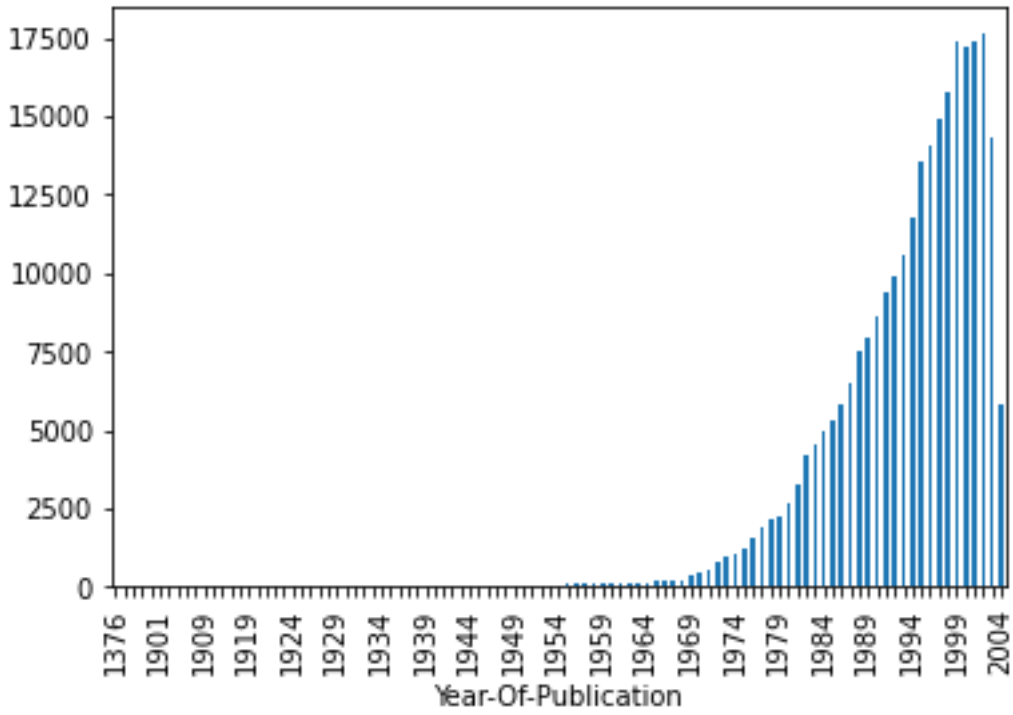


Hay 647476 likes (63% de los ratings)



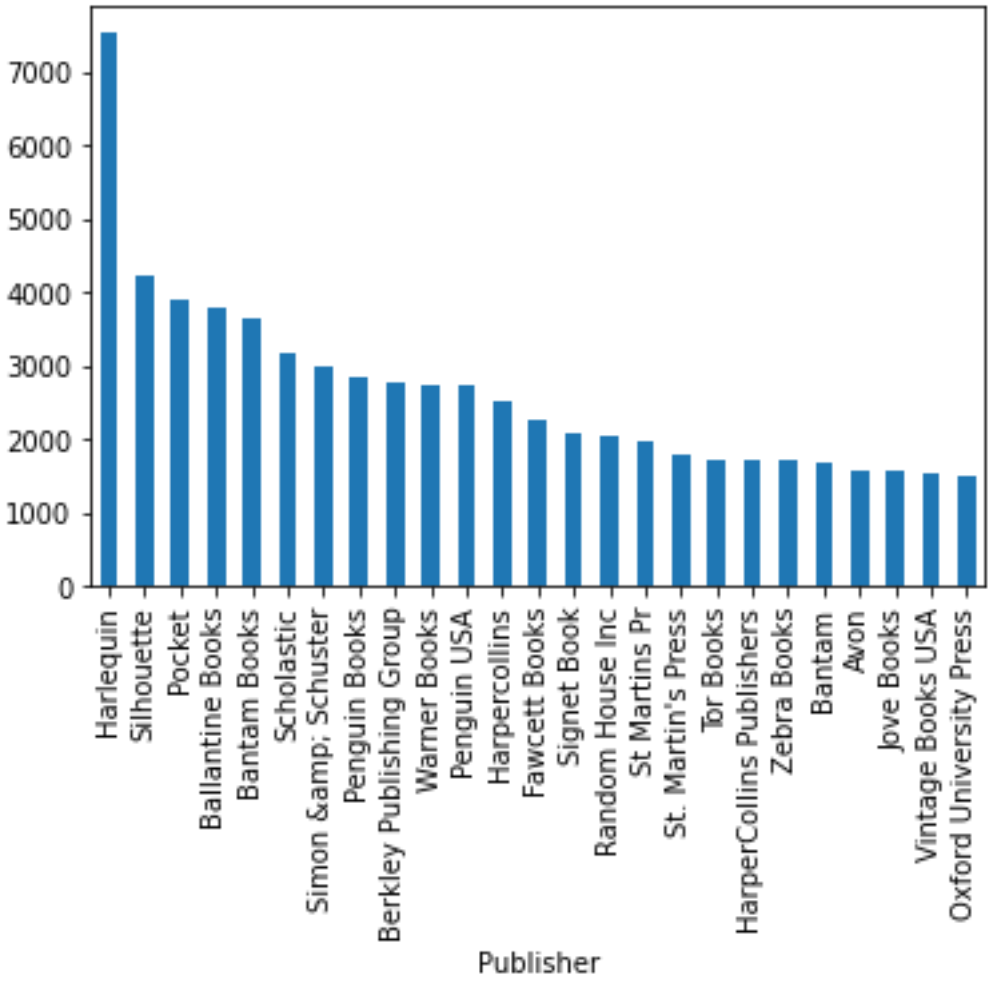
Ratings de cada usuario por libro

g. <u>Libros por año de publicación</u>



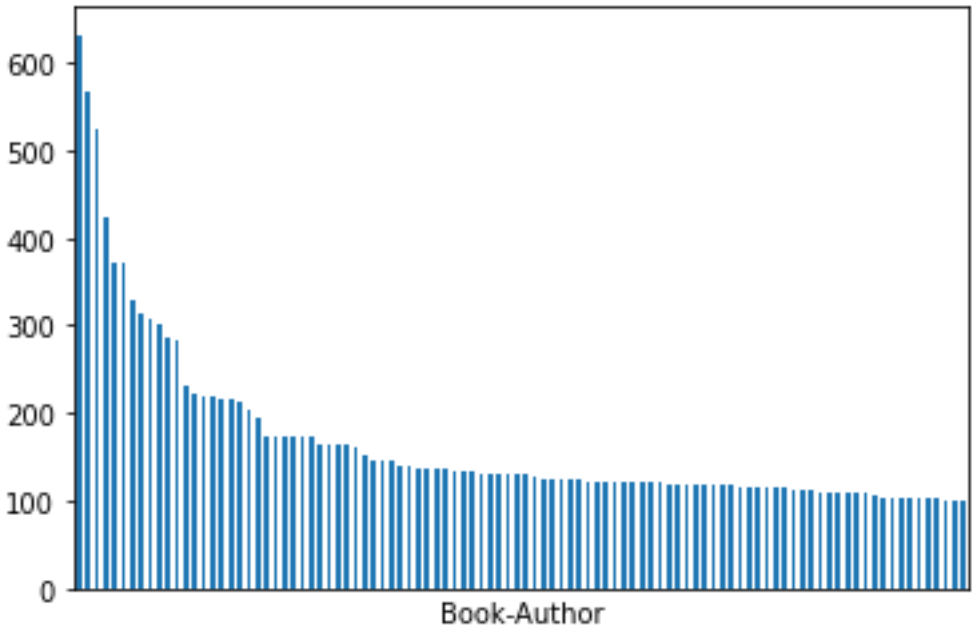
80% de los libros fueron publicados entre 1982 y 2002.
2% de los libros no tienen año registrado (4691 libros)

h. <u>Editoriales con más libros</u>



Promedio: 16.15 libros publicados

i. <u>Autores con más libros</u>



Promedio: 2.66 libros escritos por autor.

j. Recapitulativo
   - 271 379 libros en total
   - 270 172 libros con ratings
   - 101 420 usuarios en total
   - 1 031 511 ratings en total
   - 47% de los ratings están alojados en 5% de los libros
   - 13 845 de 270 172 libros tienen más de 12 ratings.

3. Tratamiento

Según los gráficos y estadísticas obtenidas. Hay una enorme cantidad de usuarios inactivos, y de libros muy poco populares. Pero tampoco no interesa recomendar libros muy populares, ya que posiblemente se conozcan y generen similitudes incorrectas entre usuarios y libros.

Por lo que voy a escoger buscar similitud entre los libros que tengan entre 12 y 400 ratings (representa aproximadamente 5% del dataset de libros, pero estimo que 13 568 libros es suficiente para un sistema de recomendaciones)

Además, escojo los usuarios que hayan calificado más de 4 libros (representa 25% de la tabla *users,* 19 555 usuarios)

Filtramos entonces el DataFrame *ratings* guardando únicamente los ratings de libros y usuarios activos. Reducimos de más del 50% la tabla *ratings*.

De no hacer esta reducción, al calcular las matrices de ratings según libros y usuarios, tendríamos una cantidad ridícula de información que se aloja en la memoria RAM (la matriz fuera una matriz dispersa 100 000 x 150 000, a alojar más de 30 GB)

4. Sistema de recomendación

a. User-User Collaborative Filtering

La idea de este acercamiento es de buscar similitudes entre usuarios según los gustos de cada uno, y hacer recomendaciones en base a lo que los usuarios similares prefieren.
Por esto, construimos una matriz con el ID del usuario en líneas, las *features* en columnas. En este caso, uso como *features* los libros calificados por los usuarios, la edad y el país de cada usuario.
Usamos la función pivot_table() de *pandas* para general la tabla con esta forma.
Factorizamos la columna “Country” con factorize(), para convertir esta variable no-numérica en ordinal.
Luego, para evitar divisiones entre 0 en el calculo de la similitud coseno (o en el caso de usar el coeficiente de Pearson), asigno el valor -1 a los NaN, y sumo 1 a cada ordinal de “Country”.
Llamo a esta nueva tabla *user_item_matrix.*

Para economizar en memoria, guardo la matriz en una matriz csr, usando la librería SciPy.
La matriz tiene la forma siguiente:

User-ID	Age	Country	0028604199	...	1573228214	1857022424	1931561648
165	62.0	1.0	-1.0	...	-1.0	-1.0	-1.0
243	0.0	1.0	-1.0	...	-1.0	-1.0	-1.0
254	24.0	1.0	-1.0	...	-1.0	-1.0	-1.0
300	0.0	1.0	-1.0	...	-1.0	-1.0	-1.0
383	0.0	1.0	-1.0	...	-1.0	-1.0	-1.0
...	...	...	...	...	...	...	...
278545	0.0	1.0	-1.0	...	-1.0	-1.0	-1.0
278633	0.0	1.0	-1.0	...	-1.0	-1.0	-1.0
278771	0.0	1.0	-1.0	...	-1.0	-1.0	-1.0
278774	33.0	3.0	-1.0	...	-1.0	-1.0	-1.0
278843	28.0	1.0	-1.0	...	-1.0	-1.0	-1.0

b. Item-Item Collaborative Filtering

Hacemos un proceso simétrico al anterior, pero poniendo los libros como filas, y features (usuarios que han calificado los libros, Autor del libro, Año de publicación, editor) en columnas, en nuestra matriz *item_user.*
Factorizamos y la tratamos de la misma manera.
La matriz tiene la forma siguiente:

	Book-Author	Year-Of-Publication	Publisher	...	278771	278774	278843
0441783589	1	1987	1	...	-1	-1	-1
0061099325	2	1992	2	...	-1	-1	-1
0446601640	3	1994	3	...	-1	-1	-1
0449911004	4	1996	4	...	-1	-1	-1
0375719180	5	2002	5	...	-1	-1	-1
...	...	...	...	...	...	...	...
0552999458	157	2002	76	...	-1	-1	-1
0553584375	158	2003	18	...	-1	-1	-1
0446354732	9	1995	3	...	-1	-1	-1
0553290789	11	1999	18	...	-1	-1	-1
0310205719	159	2002	77	...	-1	-1	-1

c. Obtención de los K-Nearest Neighbors

Usamos la función NearestNeighbors de Scikit-Learn, (metric='cosine',algorithm='brute' en parámetros) para obtener los usuarios más similares a un usuario ingresado, o los libros más similares a un libro ingresado.

d. Recomendaciones

En el momento que un usuario califique un libro, recalcularía las matrices *item_user* y *user_item* después de añadir el nuevo registro, únicamente si,respectivamente, el libro tiene más del mínimo de ratings, o si el usuario ha dejado más del mínimo de ratings establecido. De no ser el caso, no se pueden dar recomendaciones personalizadas y se deberían de mostrar los mas populares.
Luego de recalcular las matrices, vuelvo a fit el modelo KNN con la nuevas matrices y busco los K Nearest neighbors del libro o usuario que se ingresó por ultimo.

En el caso del approach item-item, se recomendarían los libros más parecidos, que son ya resultados del modelo. Hay que tomar libros que no hayan sido calificados por el usuario, y de no haber, hay que buscar los KNN siguientes.
Para el approach user-user, el KNN nos devuelve los usuarios más parecidos al que se ingresó por último, por lo que habría que buscar cuales son los libros que más han gustado a los usuarios más parecidos. Se podría, para este caso, asignar pesos a los libros recomendados por los usuarios similares según su popularidad, y recomendar ignorando los libros que el usuario ya ha calificado.