EX 3 – ARTIFICIAL INTELLIGENCE

Part 1 : Non-Personalized

1. **The data.**

For non-personalized recommender system, we will calculate the weighted average rating for each book.

The data we need is: books and ratings. In the file books.csv: book_id and title. And in the file ratings.csv: book_id and rating.

(The Non-personalized algorithm recommends according to the rating. All users have the same recommendations.)

For the following questions in the exercise, we'll also need more information on the users to target the recommendations (like place and age).

2. **Get simply recommendation.**

To get the k best recommendations,

- We first, calculated the number of voters for each book: v

- We calculated the average rating for each book: R

- We fixed the minimum number of voters, like in the Tirgul: (quantile(90)) : m

- We got the average rating of all books: C

- Finally, we calculate the weighted average rating:

$$\mathbf{WR} = \frac{v}{v+m} * R + \frac{m}{v+m} * C$$

-The function returns the k books with the highest value of WR.

The 10 recommended books are:

book_id	title	weighted_average_rating
25	Harry Potter and the Deathly Hallows (Harry Po...	4.338028
4	To Kill a Mockingbird	4.299843
102	Where the Wild Things Are	4.273212
85	The Giving Tree	4.240309
50	Where the Sidewalk Ends	4.239724
31	The Help	4.238851
144	Unbroken: A World War II Story of Survival, Re...	4.221864
27	Harry Potter and the Half-Blood Prince (Harry ...	4.213906
1	The Hunger Games (The Hunger Games, #1)	4.187383
133	Anne of Green Gables (Anne of Green Gables, #1)	4.181489

3. **Get simply place recommendations.**

We'd like to do the same, but we want to target the recommendations according to the living location of the user.

The 10 recommended books for Ohio are:

book_id	title	weighted_average_rating
126	Dune (Dune Chronicles #1)	4.367963
143	All the Light We Cannot See	4.317786
144	Unbroken: A World War II Story of Survival, Re...	4.266087
24	Harry Potter and the Goblet of Fire (Harry Pot...	4.249728
102	Where the Wild Things Are	4.226877
490	Maus I: A Survivor's Tale: My Father Bleeds Hi...	4.213664
1462	The Orphan Master's Son	4.213664
983	Between the World and Me	4.213664
119	The Handmaid's Tale	4.199565
89	The Princess Bride	4.190062

4. **Get simply age recommendation**

We want to target the recommendations according to the age of the user.

The 10 recommended books for a 28-year-old (21 to 30) user are:

book_id	title	weighted_average_rating
25	Harry Potter and the Deathly Hallows (Harry Po...	4.326251
4	To Kill a Mockingbird	4.294203
85	The Giving Tree	4.289614
89	The Princess Bride	4.244702
133	Anne of Green Gables (Anne of Green Gables, #1)	4.224914
50	Where the Sidewalk Ends	4.216411
102	Where the Wild Things Are	4.204680
70	Ender's Game (Ender's Saga, #1)	4.204095
31	The Help	4.202891
21	Harry Potter and the Order of the Phoenix (Har...	4.196385

Part 2: Collaborative filtering

Everything is in the code.

As an example, here's the recommendations we got for user 1:

book_id	title
101	Me Talk Pretty One Day
775	Just Kids
264	The Sun Also Rises
289	Watership Down (Watership Down, #1)
335	James and the Giant Peach
1084	To the Lighthouse
468	Their Eyes Were Watching God
184	Matilda
83	A Tale of Two Cities
344	Naked

Part 3: Contact based filtering

8. Features.

The features we used to work with are: **language, tags, original title and authors**.

We tried different features to choose the best ones. It seems logical that if we want a recommendation for Twilight or Harry Potter, or The Hunger Games, it will recommend us the other books of the Saga. That's why the title's book is important.

Moreover, usually, every author has its own writing style. We can see it through their different books. And if I liked a book of an author, I would like to get recommendations for its other books.

Language has also its logical impact. The way it's been written, the language the lector speaks and language can also mean culture. French books are different from American books.

Tags give us some hints about the books: like the genre. It's also important for the recommendation.

We also tried with the publication year but it adds noises. It limits us. The year is taken as a string, so it's either equal or different. That is not what we want.

10. Recommendations for Twilight.

We got the following recommendations for Twilight (different Twilight existing so for the first one):

title	book_id
The Twilight Saga: The Official Illustrated Gu...	4088
Eclipse (Twilight, #3)	52
The Host (The Host, #1)	73
Twilight: The Graphic Novel, Vol. 1 (Twilight:...	3075
Breaking Dawn (Twilight, #4)	56
The Twilight Collection (Twilight, #1-3)	2021
New Moon (Twilight, #2)	49
The Twilight Saga (Twilight, #1-4)	992
The Twilight Saga Complete Collection (Twilig...	1619
Crossroads of Twilight (Wheel of Time, #10)	1525

Part 4: Evaluations

11. Table of evaluations.

	Precision_k	ARHR	RMSE
Cosine	0.08	0.6466666666666666	0.9176794695882072
Euclidian	0.008	0.08	0.9168899103744533
Jaccard	0.08	0.6266666666666666	0.9187216947055077

12. Explanation of those results.

We get a weak precision_k for every similarity. The reason to this is that the test file is very small so it can't give us a good precision. We don't have enough information.

We could have got better results for ARHR, but we didn't for the same reasons. If we had more samples in our test file, the ARHR would be better. We still get better results than precision_k because we take into account the position of the books.

Moreover, precision_k and ARHR, use only the top 10 recommendations that have been given.

RMSE takes into account the predicted results and compare it to the actual one. Only the difference between the rankings matters.

Those results show us that our rankings are good and that we succeeded to find the right rank of the recommendation (ARHR high).