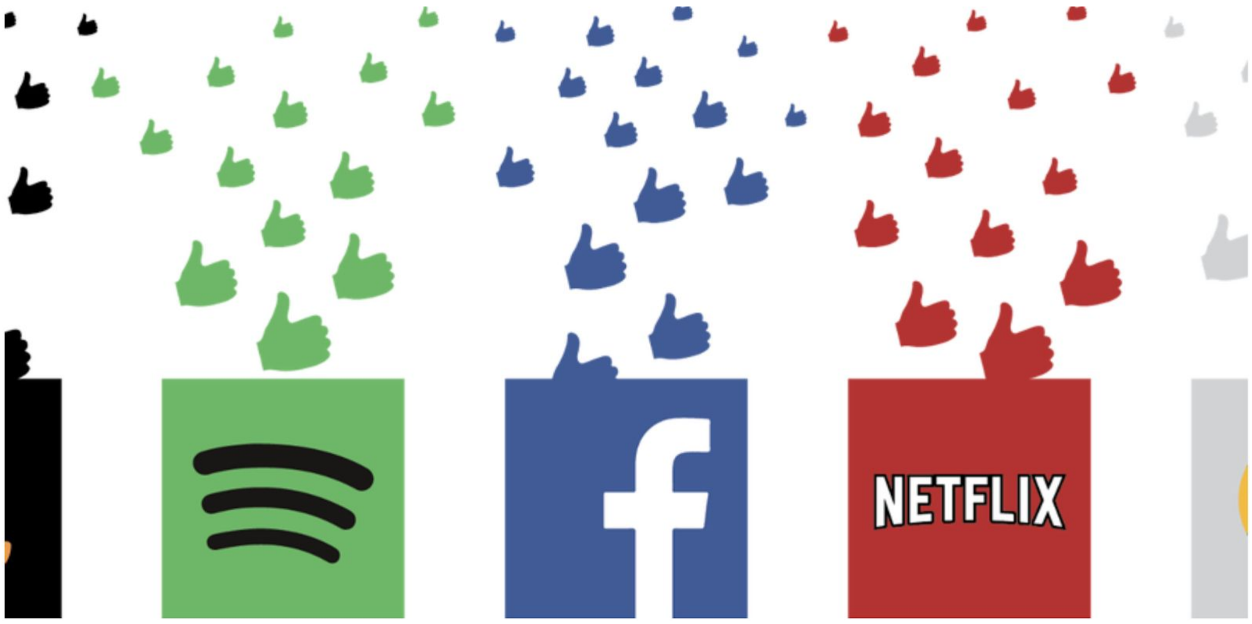
“Welcome to the Age of Recommendations!”



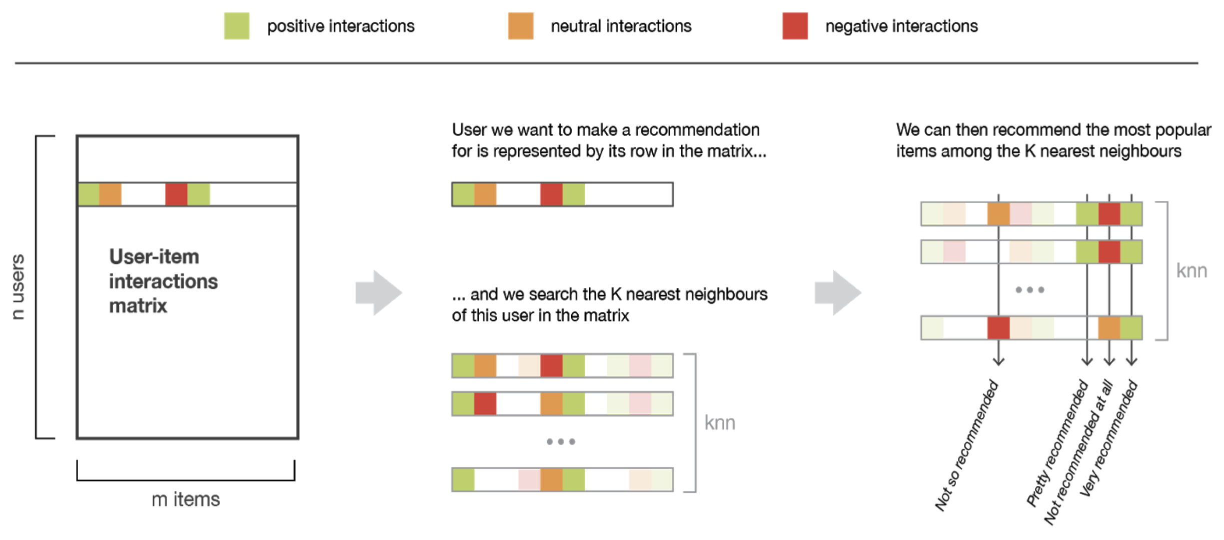
Recommendation models have existed for a couple of decades now and with time they have become more and more efficient. With so many options available on Amazon, Netflix or UberEats, it is ideal to have your preferences defined thereby reducing your screen time for every decision you want to make. Unfortunately, even today we are haunted by items we have already seen earlier or are not interested in. As an attempt to make this more efficient, we built our own model by implementing techniques we have acquired in our Business Analytics class.

We used a holistic approach by exploring previous data that was available to us, for instance Amazon product reviews from the UCSD Data repository. We chose to focus on a smaller dataset, ie. Luxury Beauty Products, which includes 3 features: *product_id, user_id* and *rating*. The dataset contains 86,771 ratings, 19,478 users that rated more than 3 products, and a total of 8,308 products. This data was leveraged to drive insights from different models we built for the sake of better recommendations, and hence increase the business value of a platform/ value of a product to customers.

Types of Recommendation systems

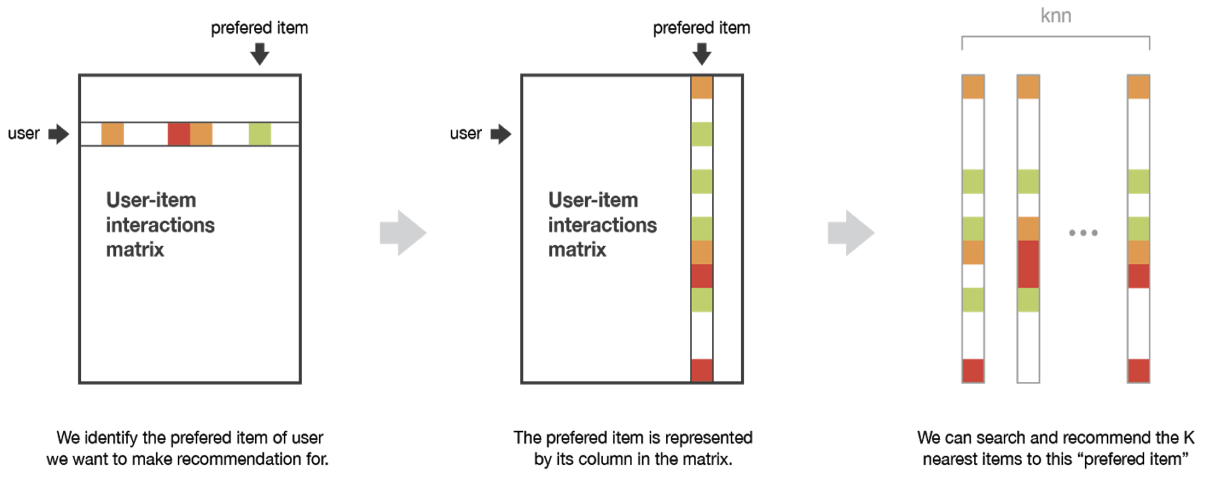
Typically there are different forms of recommendation systems, namely Content Based, Collaborative Filtering and a hybrid of the two. The Content Based recommendation system, which can be both user-centered and item-centered, relies on user data (demographics, CLV Engagement history, etc.) and item data (pricing, brand, quantity, form, color etc.). A simpler and very effective version is Collaborative Filtering. This method only requires a user-item interaction matrix and can be either Memory Based (User-user and item-item interaction) or Model based (Matrix Factorisation). Given the limitations in our dataset, we chose to focus on Collaborative Filtering.

In the user-user interaction, we arrange the users in vectors based on the products they rated. We then find the K nearest neighbors of all the closest similar users and recommend the most popular items among those K nearest users that were chosen. The below image illustrates the same:


positive interactions
neutral interactions
negative interactions
n users
User-item interactions matrix
m items
User we want to make a recommendation for is represented by its row in the matrix...
... and we search the K nearest neighbours of this user in the matrix
knn
We can then recommend the most popular items among the K nearest neighbours
knn
Not so recommended
Pretty recommended
Not recommended at all
Very recommended

Source: Introduction to recommendation systems by Baptiste Rocca

It is vital to note that the idea of item-item method is to find items similar to the ones the user already “positively” interacted with. Item-Item interaction is very similar to user-user but here we make a recommendation to the user based on the products he rated. It's the same idea but by transposing the matrix, we can compute similarities between the “preferred item” of a specific user and all the other items. Once the similarities have been computed, we can then keep the k-nearest-neighbours to the selected “best item” that are new to our user of interest and recommend these items.


prefered item
user
User-item interactions matrix
We identify the prefered item of user we want to make recommendation for.
prefered item
user
User-item interactions matrix
The prefered item is represented by its column in the matrix.
knn
We can search and recommend the K nearest items to this “prefered item”

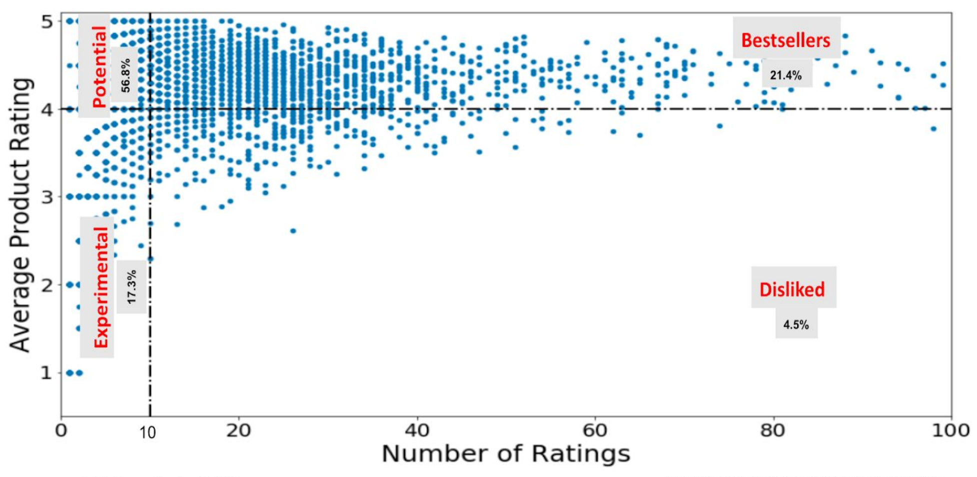
Source: Introduction to recommendation systems by Baptiste Rocca

An important advantage of the Collaborative Filtering method is that it does not need any information about users and items independently: as more data of the user/item interaction becomes available the more personalized the recommendations become. On the flipside, a big disadvantage known as the Cold Start Problem presents itself. It is impossible to recommend existing items to new users or new items to existing users. Another disadvantage is that products that are rated well get recommended more often which in other words is that “the rich get richer” which could be unfair to other products on the site.

What does the LéCUMIN do differently?

The focal point will be how to address the Cold Start Problem, knowing that different models perform better in different circumstances. Before expanding our model and to get a better understanding of our dataset, we segmented both items and products in four different categories. Products will be segmented into “Potential”, “Bestsellers”, “Experimental” and “Disliked”; and Users titled as “Optimistic” (fewer high ratings given), “Champions” (several high ratings given), “New” (rated fewer products with an average rating under 3.5) and “At Risk” (multiple bad reviews given).

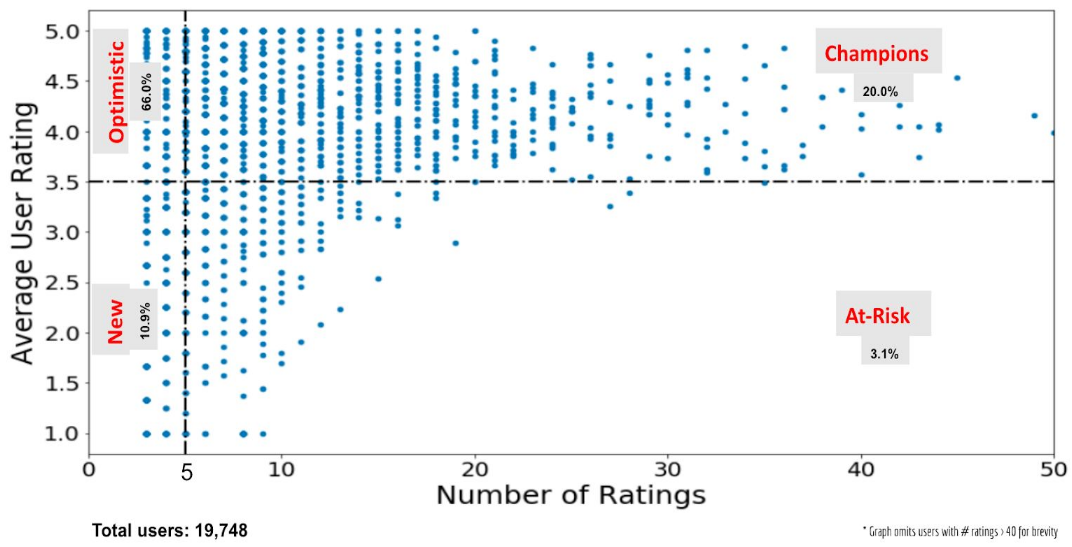
Product Segmentation



Total products: 8,308

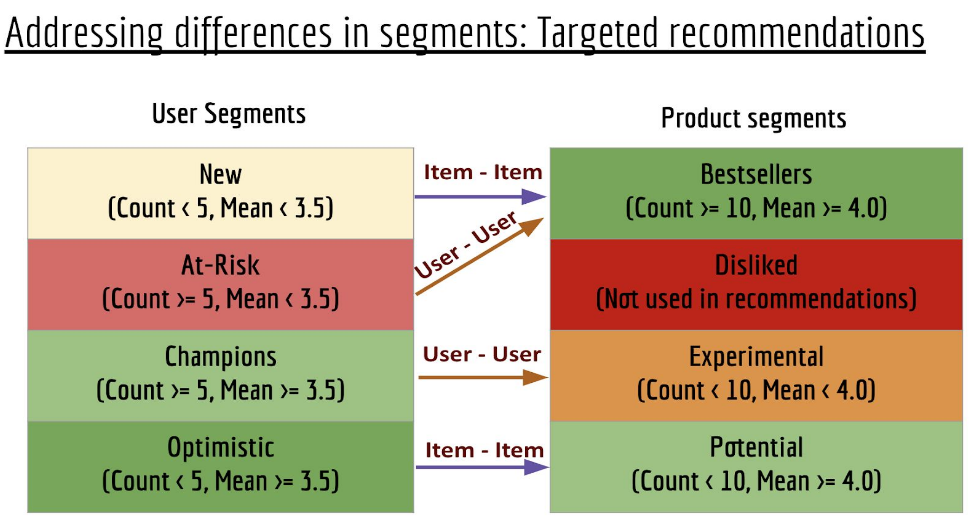
* Graph omits products with # ratings › 200 and average rating ‹ 2.5 for brevity

User Segmentation

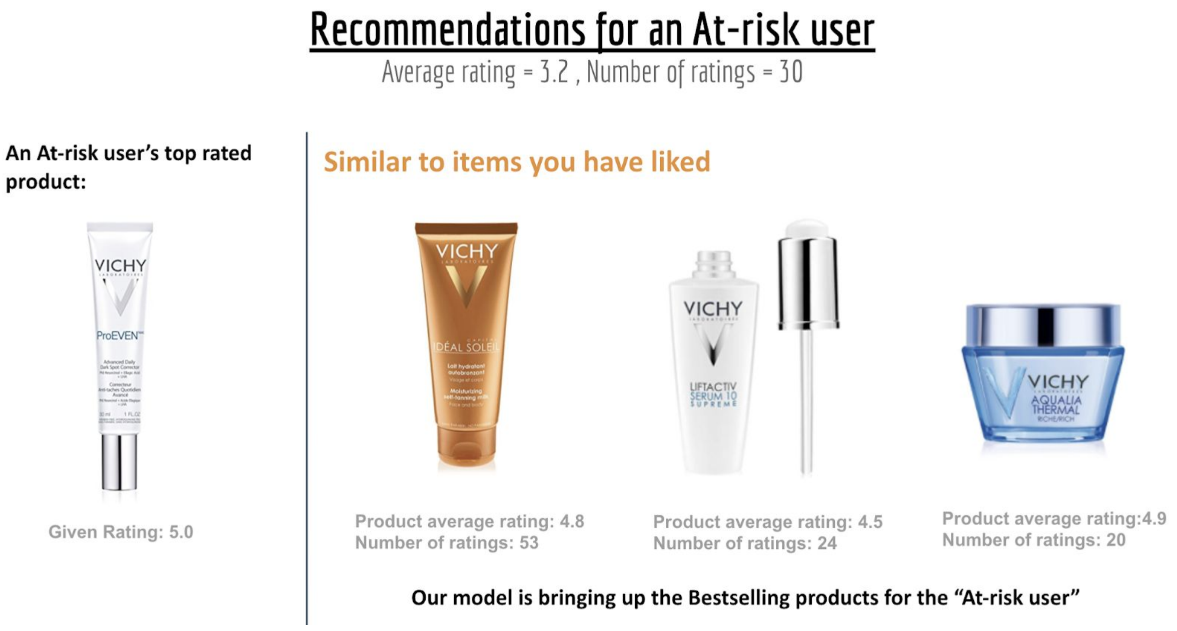


Our recommendation system, Lé CUMIN, leverages this product and user segmentation to improve recommendation accuracy. The segment partition was created to imply targeted recommendations to the different user groups. Bestseller products will be recommended to New and At-Risk users, helping in acquiring new customers or avoid losing unsatisfied ones. Experimental products shown to Champions, with whom there is more room to error in case they dislike a product. Potential products to optimistic users and disliked products will be removed from the recommendations completely.

Also, the Lé CUMIN uses the User-User model when the number of ratings per user is high (segments Champions & At-risk) and Item-Item model when there are too few ratings per user to find a similar user based on user proximity (New and Optimistic).


Addressing differences in segments: Targeted recommendations
User Segments
Product segments
New (Count < 5, Mean < 3.5)
At-Risk (Count >= 5, Mean < 3.5)
Champions (Count >= 5, Mean >= 3.5)
Optimistic (Count < 5, Mean >= 3.5)
Item - Item
User - User
User - User
Item - Item
Bestsellers (Count >= 10, Mean >= 4.0)
Disliked (Not used in recommendations)
Experimental (Count < 10, Mean < 4.0)
Potential (Count < 10, Mean >= 4.0)

Representation of the business rules in LéCumin


Recommendations for an At-risk user
Average rating = 3.2 , Number of ratings = 30
An At-risk user's top rated product:
Given Rating: 5.0
Similar to items you have liked
Product average rating: 4.8
Number of ratings: 53
Product average rating: 4.5
Number of ratings: 24
Product average rating:4.9
Number of ratings: 20
Our model is bringing up the Bestselling products for the "At-risk user"

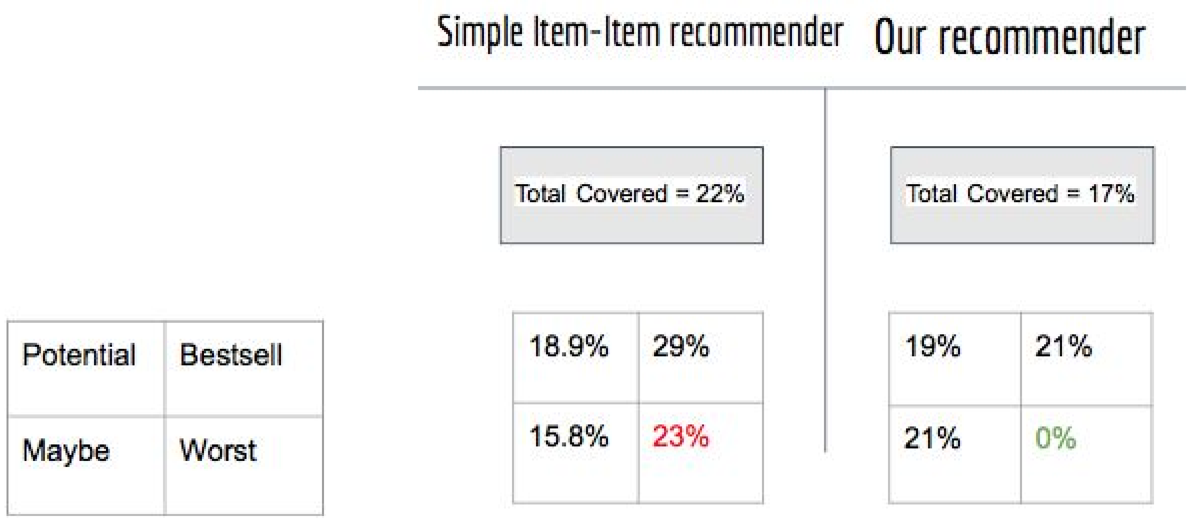
Business rules + Improvements made

This model can be used for different businesses by tweaking the threshold to creating user and product segments based on the different business contexts. While we took a stab at making recommendations generally better, there is still scope to refine the model further. One can leverage matrix factorisation to recreate the user-item interaction matrix. With more information, one can build a content based model or in a more ideal case, a hybrid model that combines both.

How did our leCUMIN recommender give us better product coverage ?

Coverage : Percentage of items included in the recommendation list over the total number of items.

We created a simulation to compare the coverage result for a simple Item-Item Recommender and our Recommender.



Potential	Bestsell
Maybe	Worst

Simple Item-Item recommender

Total Covered = 22%

18.9%	29%
15.8%	23%

Our recommender

Total Covered = 17%

19%	21%
21%	0%

Benefits :

- ➔ Our recommender never shows **Disliked** products. We increase users' overall happiness.
- ➔ We also show more **potential** and **Experimental** products to our clients. We give a better platform for suppliers to launch new products and get them to bestsellers.
- ➔ We recommend the "bestsellers" to new users, overcoming the cold-start problem. We also recommend similar products to at-risk users, so that we don't lose a valuable customer.

Future scope:

- Combine our model with a Content Based model to give more personalized results.
- Use methods like A/B testing or Reinforcement Learning to make a better model
- Extend the same model to be used for overall products on the website.

How would you build your own recommendation system? Share your thoughts!