On the Effectiveness of Personalized Candidate Generation in Recommendation Systems: A Comparative Study of Feature-based Approaches and Pure Popularity Models

Maksim Karnaukh
1st year Masters at Department of Computer Science
University of Antwerp
Antwerp, Belgium
Email: maksim.karnaukh@student.uantwerpen.be

***Abstract*—Personalized recommendation systems play a crucial role in enhancing user experience and engagement. This study investigates the effectiveness of a recommendation approach that combines the features mean age per article, gender and average purchasing price per customer compared to a pure popularity-based approach. The methodology involves the creation of the three features mentioned above, followed by a customized candidate generation process. The evaluation is based on the recall metric, while also comparing the recommendations with a baseline.**

1. Introduction

In recent years, personalized recommendation systems have become integral to various online platforms, providing users with tailored suggestions based on their preferences and behavior.

At the core of any recommendation system is the crucial task of predicting user engagement with specific items. An important step, often among the first, in building an effective recommendation system is creating a good candidate generation [1] [3] method. This method aims to select a small amount of relevant candidates from an extensive pool of options. Common strategies include content-based filtering, which relies on content similarity for recommendations and collaborative filtering, which uses user and item similarities.

In my exploration, I initially delved into collaborative filtering with as goal to test the effectiveness of graph-based recommendation algorithms in capturing customer-item interactions and their potential to enhance recommendations in a cold-start scenario. The use of Graph Convolutional Networks (GCNs) [6] appeared promising. GCN models represent a class of neural network architectures designed to exploit the inherent graph structure. They excel at aggregating node information from neighboring nodes in a convolutional manner.

However, despite the initial promise, I encountered challenges in implementing and successfully concluding the GCN approach [10] [11] for this project. The complexities involved prevented me from achieving the desired results, leading to a change in strategy used. This realization prompted a shift towards focusing my research on a content-based filtering strategy, where the emphasis lies in creating and leveraging feature combinations.

This study aims to assess the effectiveness of a recommendation approach that leverages a combination of features, including age, gender, and average purchasing price. The central research question is: "How effective is the use of a combination of features for generating personalized candidates and ranking them compared to a pure popularity-based approach?"

2. Dataset Overview

In this study, we work on a comprehensive dataset to explore and analyze user interactions within a dynamic content environment. The dataset, sourced from [5], serves as the foundation for our investigation into personalized candidate selection strategies. It contains the purchase history of customers across time, along with supporting metadata. The transactions span 104 weeks, of which we will use the last ten weeks as training/validation data. Prior to analysis, the dataset underwent some preprocessing to address possible inconsistencies or missing values using [7].

3. Related Work

I base myself on and use Radek's LGBMRanker starterpack [4], which serves as a baseline that focuses on a popularity based approach for the candidate generation. Important to mention is that recency is also used to generate the first set of candidates. Since I wanted to compare against the pure popularity approach, I opted to also start with the same set of generated candidates based on recency. In the popularity-based approach, the 12 bestselling articles are generated as candidates for the next week (per week) and selected for all customers in that week, meaning each customer has the same 12 candidates.

4. Methodology

To address the research question, I designed a methodology that involves the creation of three key features:

1) **mean_price_per_c**: the mean price per customer, represents the average purchasing price of the customer.
2) **highest_count_ign_per_c**: the highest count of index group name per customer, identifies the most bought article index_group_name of the customer.
3) **mean_age_per_a**: the mean age per article, represents the average age of everyone who bought a specific article.

The entire dataset has a decent amount of feature columns, either labeled as metadata or not. This includes three feature columns that are already present: the price of an article, the index group name of an article and the age of a customer. Based on these three columns, I chose my three engineered features, since to me, they seemed to be the best for determining what a customer might like to buy.

4.1. Feature Engineering

What follows here is some explanation as to how I created my own features and their meaning. For the first feature, the mean price per customer is calculated by taking the prices from all the articles that a customer bought and calculating the mean between them. This should be useful under the assumption that customers that (usually) buy cheaper articles, will stay in that cheaper price range when buying a new article and the same applies to expensive articles. For the second feature, I wanted to use the gender of a customer to help predict whether a customer would buy an item. The problem was to be able to extract a customer's gender from his purchases since this feature wasn't explicitly present in the dataset. The best way -and most simple for me- seemed to be to make use of the index group name. This has five different values (0, 1, 2, 3 and 4 which represent 'Ladieswear', 'Baby/Children', 'Divided', 'Menswear' and 'Sports' respectively) which can function well to divide the customer in a category similar to gender. This makes my second feature be the highest count index group name per customer. Here, we look at all the articles a customer bought and count how many times each index group name appears in order to determine the highest count index group name for that customer. The mean age per article is created by taking all the customers that bought that article and calculating the mean age of all those customers. This should be useful under the assumption that customers will buy articles that are usually bought by people close to their age.

I didn't look for proof that these features work well beforehand. They seemed to be a logical set of 'questions to ask' someone when trying to predict as well as possible which articles that person will buy.

4.2. Candidate Generation

Concerning the candidate generation process, I first of all use the method of repurchase candidates, which the popularity-based baseline also uses. This means that articles bought in a certain week are considered candidates for the next week per customer.

The general way I generate my candidates based on my three features is to make certain calculations with different features on transactions during a certain week. Based on this, we generate candidates for the next week. How we calculate which articles to select as candidates is as follows:

1) First, for each customer we select all the articles where the highest_count_ign_per_c of the customer is equal to the index_group_name of the article.
2) We further filter these articles by first ranking them based on the smallest difference between the (last known up to that point) price of the article and the mean_price_per_c of the customer. This is called the price rank. Then we select the top x of these articles as current candidates. The exact value(s) of x used will be explained later.
3) Eventually, we refine the selection by ranking those x articles based on the smallest difference between the customer's age and the mean_age_per_a of the article (this is called the age rank) and selecting the top y of these articles as final candidates of this candidate generation method.

The result is a dataframe with y candidate articles for each unique customer present in the training data, per week. In the general run, x would be 50 and y would be 12, meaning we would eventually generate 12 candidates per customer, per week (since we generate candidates per week).

4.3. Evaluation

To evaluate the effectiveness of the personalized candidates approach, we employ the recall metric on the last week of the training data, which we use as validation data. This metric measures the ability of the system to recommend relevant items, providing insights into the system's coverage of the user's actual preferences. The comparison is made against the baseline, which employs a pure popularity-based approach.

4.4. LGBM Ranker Integration

As mentioned before, I use the same code as Radek's popularity-based starter pack which uses an LGBM ranker for the final predictions. I replaced his popularity based candidate generation method with my method after of course also creating the necessary features. I then integrated two feature columns, the price rank and age rank, into the LGBM ranker. This step aims to assess the impact of these features on the overall ranking performance of the recommendation system.

5. Results

The results of the evaluation, including recall scores and comparisons with the baseline, are presented in subsequent sections. The impact of the added features on the LGBM ranker's performance is also analyzed.

5.1. Feature analysis

First, we shall discuss the analysis of the three features used for the candidate generation. Although this is not very important to the results of the research question, it may be interesting to look at the distributions. There are around 437000 customers in the training dataset that we used.

We can see that most of the customers have a mean price between 0 and 0.1, with the majority between 0 and 0.05. The prices are obviously scaled in some (unknown) way, but if we multiply them by 1000, we are in the range of 100 dollars, which is reasonable enough to assume the correctness of the multiplier. There are some outliers with a maximum mean price per customer of around 0.5, but exploratory data analysis on the price column has shown that customers that have made a very expensive purchase, have a high mean price that is relatively close to that expensive price range. Clipping them isn't necessary since we want to use this information.

Most of the articles have a mean age that falls in the range between 20 and 60 years old, with high frequencies around 35. Concerning the age column in the customers' metadata dataset, I replaced incorrect values (NaN or -1) at the beginning with the overall mean age to ensure a more fair approach to calculating the mean age per customer feature.

Finally, we see that around 70% of the customers have 'Ladieswear' as their preferred category. Around 25% are 'Divided' and the rest of the categories all have less than 25000 customers each.

Moving on to the preferred category (= index group name) analysis, we observe that around 70% of the customers have 'Ladieswear' as their preferred category. Approximately 25% prefer the 'Divided' category, suggesting a diverse customer interest. The remaining categories each have less than 25,000 customers. Understanding the distribution of preferred categories can provide valuable insights into customer preferences.

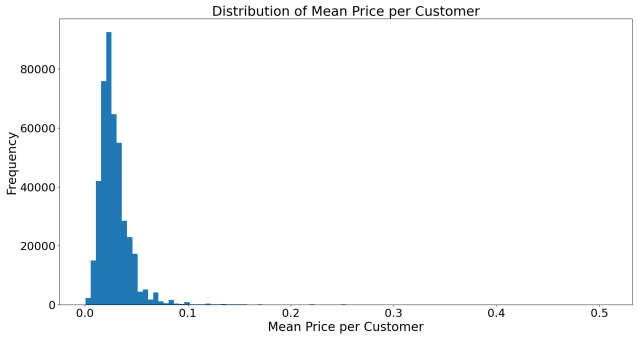

Figure 1. Distribution of mean price per customer.

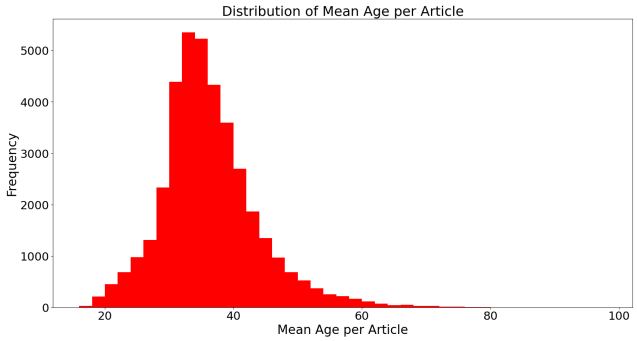

Figure 2. Distribution of mean age per article.

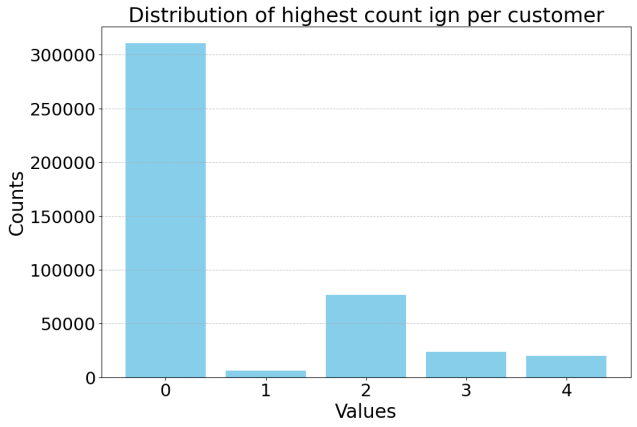

Figure 3. Distribution of highest count index group name per customer.

5.2. Recall Scores

The recall scores are computed for different configurations, varying x and y in the candidate generation process. These scores should provide insights into how well my personalized candidate generation method captures the preferences of the customers, also compared to the baseline.

5.3. LGBM Ranker Performance

The LGBM ranker's performance is assessed by comparing its ranking results to those of the baseline. This analysis aims to determine whether the additional features (in this case the price_rank and age_rank) contribute positively to the overall recommendation ranking.

6. Discussion

To see how well my three feature-based ('3Feature' in the table) worked, I calculated the recall using the last week (week 104) of the training data as validation data, both for my method and for Radek's Bestseller (popularity) candidate generation method. Here, we both generated 12 candidates. Sadly, it seems that my method has a recall score

TABLE 1. MY CANDIDATE GENERATION SCORES COMPARED TO BESTSELLER CANDIDATE GENERATION.

Validation Week	101	101	102	103	104
3Feature Candidates					
Recall	0.0	0.0	0.0	0.0	0.0
Bestseller Candidates					
Recall	0.020	0.023	0.025	0.020	0.025
Intersection	Empty	Empty	Empty	Empty	Empty

of exactly 0, which means not a single candidate is actually bought. I tried to then use the four weeks before week 104 as validation to see if something would be different there, but to no avail. The popularity-based approach seems to get recall scores between 0.02 and 0.025 for the five weeks used as validation. To see how different my personal candidates were from what Radek's baseline had, I calculated the intersection between the generated candidates per week to see if there were any similarities. However, all the intersections proved to be empty.

A final attempt at solving the problem of a recall score of 0 was to increase the number of candidates in the hope to get at least a few true positive matches. As mentioned before, the general 3 feature-based candidate generation process works by filtering by index group name, then selecting the best 50 articles based on price difference and eventually the top 12 of those based on age difference. I write this as a 50-12 selection. I thus tried to use 50-25, 100-50 and 200-100 selections, but the scores didn't improve at all and stayed 0.

Concerning the comparison of the ranking results, since the ranker code ran out of memory for the large size of the dataset for me, I used 6000 generated candidates by my method to compare the MAP@12 score (Mean Average Precision) with the one from the popularity baseline. Radek's baseline had a score of 0.02046, while I got 0.01612 which is slightly worse. Important to remember is that the recency method (repurchase candidates) is included with both (mine and Radek's) used candidate generation methods. Of the two newly added feature columns for the ranker to consider (price rank and age rank), price rank scored quite high on the importance list with an importance of 68%, which was the highest in this case. The age rank turned out to be quite useless with an importance score of 0.

The answer to my research question then seems rather easy, although the exact reason for it I do not know for sure. The answer is that my features used and/or the exact ways the candidates are generated using these features do not work well at all to predict which articles a customer will buy. The rest here will be discussed in the conclusion.

7. Conclusion

Although my candidate generation method using my three features didn't turn out to work very well, I learned a lot about the process of generating candidates. At first I was confused about the exact way to engineer the necessary features in a correct manner and the structure of the candidate generation process, but it happens in a rather elegant way. I read a lot about Graph Convolutional Networks for my first research question and they are indeed very interesting, maybe that they would have given me a better score.

Could my project be improved? I would be at the very least, very arrogant if I would say no. Should my engineered features actually have some potential for which I have lost hope right now, then maybe the way in which the features are created could be improved or the way in which the articles get selected in the candidate generation process e.g. change the selection order or do the ranking differently. One thing I learned here too, is that my candidate selection process is very precise. With this I mean that I select the most perfect candidates (using my candidate selection process), but if we were to look at the actual purchases of the customers, which I did, we would see that they buy articles that are not close to their mean price range or that they buy articles whose mean age is considerably higher or lower than the age of the customer in question. It should in my opinion definitely be improved in the way that it should be robust and more about finding a balance between precision and the potential randomness surrounding my chosen features (or something like that) of the customer's mind when he/she wants to buy a certain article.

I used only three features because I wanted primarily to build a strong base for prediction. This research project made me realize however that this probably just isn't enough. You need more information on a customer to have that robustness in your candidate generation that I talked about.

Acknowledgments

The author would like to thank Jens Leysen and Lien Michiels for their help and support during the making of the project and Prof. Bart Goethals for the organization and overseeing of the project.

References

[1] Aman Chadha. Recommendation Systems • Candidate Generation, 2020. https://aman.ai/recsys/candidate-gen/#:~:text=In%20recommender%20systems%2C%20candidate%20generation,to%20a%20more%20manageable%20number.

[2] Amine Dadoun. Introduction to knowledge graph-based recommender systems, 2023. https://towardsdatascience.com/introduction-to-knowledge-graph-based-recommender-systems-34254efd1960.

[3] Google. Candidate Generation Overview, n.d. https://developers.google.com/machine-learning/recommendation/overview/candidate-generation.

[4] Marco Gorelli. Radek's LGBMRanker starter-pack, 2022. https://www.kaggle.com/code/marcogorelli/radek-s-lgbmranker-starter-pack.

[5] H&M GROUP. H&M Personalized Fashion Recommendations, 2022. https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations/data.

[6] n.d. Graph convolutional networks for collaborative filtering, 2022. https://usa.visa.com/content/dam/VCOM/regional/na/us/about-visa/documents/graph-convolutional-networks-for-collaborative-filtering.pdf.

[7] Radek Osmulski. personalized fashion recs, 2022. https://github.com/radekosmulski/personalized_fashion_recs/blob/main/01_Solution_warmup.ipynb.

[8] Yan Wang Xiangnan He Quan Z. Sheng Mehmet A. Orgun Longbing Cao Francesco Ricci Philip S. Yu Shoujin Wang, Liang Hu. Graph learning based recommender systems: A review, 2021. https://arxiv.org/abs/2105.06339.

[9] Nathan Tsai and Abdullatif Jarkas. Graph-based product recommendation, 2021. https://nhtsai.github.io/graph-rec/.

[10] Xiang Wang Yan Li YongDong Zhang Meng Wang Xiangnan He, Kuan Deng. LightGCN-PyTorch, 2020. https://github.com/gusye1234/LightGCN-PyTorch/blob/master/code/model.py.

[11] Xiang Wang Yan Li YongDong Zhang Meng Wang Xiangnan He, Kuan Deng. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation, 2020. https://arxiv.org/pdf/2002.02126v4.pdf.