H&M Personalized Fashion Recommendations using Ensemble based models

Abhay D. Iyer
csci567_id78
adiyer@usc.edu

Prashant Singh
csci567_id78
singhpra@usc.edu

Subham Banga
csci567_id78
sbanga@usc.edu

1 Introduction

Over the past few years, recommendation systems (RS) have been used across multiple industries and is an active research field. Next Basket Recommendation (NBR) is a particular type of recommendation task that suggests a list of items to a user based on their past usage and/or purchase history. Specifically, e-commerce websites use recommendation engines to highlight and suggest products to customers. There are multiple strategies which can be employed to recommend products such as the season of the year, the most popular items being bought by a certain age group, based on the customer's location and suggesting items to a particular user based on what similar users might have purchased. Overall, these techniques helps personalize the user experience for customers.

Ensemble-based methods build robust models Opitz and Maclin [1999]. It can improve prediction accuracy, prevent overfitting, and can be used across multiple problem domains like recommender-systems Forouzandeh et al. [2020]. Additionally, ensemble models are generally more accurate than any of the individual models composing the ensemble Opitz and Maclin [1999]. Furthermore, Zhang et al. [2014] has also tackled the cold-start problem in RS using ensembling. RS using ensemble models have been used in multi-criteria tourist recommender systems in Nilashi et al. [2017], drug recommender systems in Zhang et al. [2016] and product recommender systems in Forouzandeh et al. [2020].

Inspired by the success of this approach across multiple domains, we propose an ensemble-based model to tackle the H&M Personalized Fashion Recommendations Kaggle competition. Our method blends strategies such as purchase-frequency, online v/s offline, clustering models based on customer age group, ranking using a LightGBM model, singular value decomposition based collaborative filtering approaches and more into a finely-tuned ensemble model. The end-goal of this work is to provide a foundation for the usage of ensemble-based models for product-based RS.

2 Methods

2.1 Data preprocessing

For a few of our initial approaches we imported the transactions_train.csv to a dataframe and specified the datatype of article_id so that the string does not implicitly get converted to int and drop the leading zeros. Additionally, we convert transaction dates to datetime objects for easier processing.

For optimal results with the the more complex models like LGBM Ranker we utilize the cudf library for GPU supported dataframe processing as proposed by Rabhi et al. [2019]. An important aspect of faster dataframe processing is modifying the datatypes of the features. For instance, after analyzing the customer ID we observed that the last 16 digits can be converted to a hexadecimal integer from a 64-byte string to reduce memory consumption. Another important data conversion was downcasting the article_id datatype from a 10-byte string to an int32 value which consumes less memory. Additionally, we create a new feature from the timestamp of the transaction to identify the week it in which the transaction took place. The starting date of the training data belongs to Week

Preprint. Under review.

0 while the last date belongs to Week 104. Moreover, in the approach where we cluster customers, we encode categorical columns like fashion_news_frequency and club_member_status to numerical values. We handle null values by assigning it a default value. Specifically, for the LSTM model due to the memory limit of Kaggle notebooks we trained the model only by utilizing data from 2020. Using the recbole library, an atomic file of iterations was created and item features were fitted into the GRU4Rec model as proposed in Tan et al. [2016]

2.2 Detailed Solution

We created an ensemble of models based on strategies mentioned below and tuned it for optimal performance. As part of our final ensemble designed, we have utilized publicly available Kaggle notebooks such as Giba [2022] to better understand the complexities of the data and optimize our approach.

2.2.1 LightGBM Ranker

A vital model utilized for candidate ranking is the Light Gradient Boosting Machine (LGBM) model. LGBM uses tree-based learning algorithms and has been shown to significantly outperform XGBoost Ke et al. [2017]. We draw inspiration from Marco [2022] and further build upon it by focusing on two candidate generation strategies within this model and fine-tuning it.

Last purchase candidates For every customer, we identify the weeks in which they purchased items and create a mapping of the current purchase week to the next purchase week. Once we create this mapping, we update the week feature in the dataframe to reflect the next purchase week for every customer instead of the current purchase's week.

Bestseller candidates For every week, we calculate the mean price of every article_id purchased in that week. Additionally, we compute the top 12 ranks based on frequency of purchase of a certain article_id in that particular week.

Hyperparameter tuning We utilized the optuna library Akiba et al. [2019] for tuning the hyperparameters of the LGBM model and train it across multiple trials by varying the values of parameters like the number of estimators of the LGBMRanker Model, the learning rate for the model, the amount of l1/l2 regularization, the boosting algorithm. We fit this model to the training dataset and generated the submission.

2.2.2 Online v/s Offline Customers

Customer shopping patterns can differ based on the medium of purchase namely, online shopping and offline shopping. Online shopping can be done through e-commerce websites such as H&M and Zara while offline shopping is done in-person at the retail outlet of these companies. Based on this hypothesis and Ryo [2022], customer shopping patterns is explored by developing a threefold strategy as part of this approach and the predictions these three strategies are blended in a weighted combination to generate a submission.

Last purchased articles The last 16 days' worth of data are considered in the transactions_train.csv and a subset of the training data grouped by the customer id is created and sorted in a descending fashion based on the date of purchase of articles. This produces the latest articles bought by all customers who purchased items in the given time-frame.

Other colors of frequently purchased articles This strategy is adopted for customers who have not purchased articles in the last 16 days. Articles which are most frequently purchased in the last week of the training dataset are noted and a list of those all products which are the same but available in different colors, is created. These different products could also be recommended to customers.

Popular articles of each product group Most frequently purchased articles of every product group such as garment upper body, socks and tights and other groups are considered.

2.2.3 Article Purchase-frequency

Heng [2022] creates three dataframes which hold transaction data for the last one, two, and three weeks of the original transactions. Similarly, it builds the purchase history dictionary of all customers in the three dataframes. Moreover, a list of the top 12 purchased items is created, based on the frequency of purchase for all three newly created transaction dataframes.

The results of these three dataframes are utilized to generate predictions per customer. We check if the customer has any transactions in the last one week. If they do, we choose their most frequently bought items for that period and add it to their personalized prediction list. We recommend the top 12 items and if it has less than 12 items, we fill up the remaining number of items by recommending the most frequently bought items of that week across all customers.

2.2.4 Singular Value Decomposition

Singular value decomposition (SVD) is a type of latent factor model which is a common collaborative filtering recommendation system strategy. Mayukh [2022] creates four training dataframes with each holding a week's transaction data of the last four weeks. To convert implicit feedback to explicit feedback, the purchase of low popularity items is treated as high explicit feedback per user. Similarly, the purchase of a popular item is considered to be of low explicit feedback per user.

FunkSVD algorithm by Xiaochen and Qicheng [2022] is utilized to help with sparse data recommendations. Additionally, article re-ranking per user is performed, by utilizing the learned features of the SVD model. For users who do not have enough items in their basket, the most popular items of the last week are recommended. Finally, the generated submission is included as part of our final ensemble.

2.2.5 Article Recommendation based on Customer Age

Customer demographics play an important role in modeling recommendation systems Beel et al. [2013]. For instance, customer age data can help enhance product-based recommendations since users of the same age groups tend to share similar interests. HechtJP [2022] explored this hypothesis by creating bins of customers based on the age. Additionally, a mapping of every age-group to the top 100 most recently purchased article_id's was created. During prediction, the customer's age is checked and we identify the top 12 items which could be recommended to the customer based on the previously created mapping. Finally, we utilize this strategy to recommend items to customers.

2.2.6 Article Recommendation based on Clustering Customers

Mulyawan et al. [2019] explore customer clustering strategies by grouping customers based on their frequency of purchase, the quantity of purchase and the price of the products using the KMeans algorithm data. Hiro [2022], builds upon 2.2.5 by clustering customers based on multiple features of the training data such as club_member_status, age, fashion_news_frequency, FN and Active. Moreover, for all such clusters, we identify the top 12 most frequently purchased items such that it can be utilized during prediction. Finally, while generating the predictions, for every customer we identify the clusters they belong to and return the top 12 items that can be recommended to them and utilize this strategy for submission.

2.2.7 Recommend articles trending on a weekly basis

Inspired by the approach adopted by Lunana [2022] we include it in our ensembling strategy. It creates a new feature to track the last day of every billing week and aggregates the data on a weekly basis by maintaining the number of transactions performed in a certain week for every article id. This strategy assumes that the week in the test set will have similar sales characteristics to the last week of the training data.

Based on this, a new weekly feature named 'quotient' is developed which calculates the sales rate of each article. The quotient per week is computed by dividing the number of units sold for a particular article in the last week by the number of units sold for the same article in any week of the training data. The approach proposes recommending previously bought products to customers and calculate the final ranking of articles based on the product of the quotient for each article and a time decaying

value which indicates the difference between when the previous purchase was made and the last date of the training dataset.

Another notebook Ahmed [2022] that was included in our ensemble strategy improves the approach as mentioned above by generating an article_id to article_id mapping. This mapping identifies articles which are commonly purchased together. Ahmed [2022] uses this approach for situations when there are less than 12 predictions for a customer to fill the remaining values. We utilize the submission generated from both notebooks in our final weighted ensemble.

2.2.8 Time-based Exponential Decay with Alternate Articles

Tarick [2022] builds on the strategy described in section 2.2.3 by recommending the most frequently purchased items of the last four weeks and by generating a popularity ranking of articles based on the difference of the last day of the training data and the date of the transaction. The reasoning is that an article purchased recently holds more relevance to customer shopping patterns than an article purchased earlier. Additionally, last week's most popular items are recommended alternatives to current week. We utilize this approach to generate a submission which is included in our final ensemble.

2.2.9 LSTM based approach

This approach was inspired by Nguyentuananh [2022], where an LSTM/GRU model was used to test the effect of a sequential model for generating recommendations. It was observed that the LSTM/GRU sequential model can be enhanced by using item features with iterations. In this approach, a default recommendation was created for those users whose output could not be generated by the model. The recommendation for such users was generated from an ensemble of approaches such as the ones mentioned in 2.2.7 and 2.2.8. RecBole was used for generating predictions for the sequential model. A combination of trending approaches and RecBole predicted model is included in final ensemble.

3 Conclusion

3.1 Results

As observed from Table 1, the individual models were combined to create an ensemble-based approach obtained a public leaderboard score of 0.0245. Our team achieved a public rank of 231 out of 3000 teams in the competition.

3.2 Insights

This section highlights some of our main insights from this competition. For instance, utilizing only the frequency of purchase of articles for customers resulted to be a trivial approach, however, it works well when coupled with a time component using the latest transaction data. Additionally, it adds a factor of how relevant a certain product is to a customer due to the selection of the time-period. Furthermore, for the LGBM Ranker based approach 2.2.1, while the model did not score well as compared to other strategies, we noted that this could most likely be due to the few number of candidate generation strategies utilized for this model. We attempted implementing other strategies for the LGBM model based on the customer age groups and postal codes, but it did not result in significant progress and instead dropped the scores. Moreover, we attempted to build a local cross validation strategy for this model to indicate the correlation between the CV scores and the leaderboard score however we ran into huge differences in the two scores and were not able to successfully test out our other strategies due to the lack of number of attempts per day (as allowed by the competition). Another surprisingly strong approach was the online vs offline strategy 2.2.2 as it also throws light on how customer purchase patterns and habits vary across online and offline retail.

Overall, given the resources and the time period we are extremely pleased with our results and look forward to competing again.

Table 1: Leaderboard score of models

Model	Leaderboard score
LGBM Ranker 2.2.1	0.0210
Online v/s Offline 2.2.2	0.0231
Article Purchase-frequency 2.2.3	0.0220
SVD 2.2.4	0.0225
Customer Age 2.2.5	0.0227
Customer Clustering 2.2.6	0.0224
Weekly trending articles 2.2.7	0.0225
Weekly trending articles + articles purchased together 2.2.7	0.0231
Time-based Exponential Decay with Alternate Articles 2.2.8	0.0217
LSTM 2.2.9	0.0221
Ensemble-based model	**0.0245**

3.3 Steps to execute code

- Please create a directory structure as indicated below

```
|-input
|-working
```

- Since we deal with predictions generated from multiple models, the total dataset size is greater than 6 GB, thus we have uploaded the dataset on google drive. The **input** folder which contains all individual model generated CSVs can be downloaded from this Google Drive link.
- Place all code files in the **working** directory. The code is available in the D2L submission.
- Install conda from link
- Create conda environment using requirements.txt provided in the D2L submission
- Activate environment

```
conda create -n cs567 --file requirements.txt
conda activate cs567
```

- Open ensemble.ipynb from working directory available from drive or D2L to build the file. Click on Cell option and select 'Run All' option.

References

Abdelnaeem H. Ahmed. Trending, 2022. URL https://www.kaggle.com/code/ebn7amdi/trending/notebook?scriptVersionId=90980162.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. URL https://arxiv.org/abs/1907.10902.

Joeran Beel, Stefan Langer, Andreas Nürnberger, and Marcel Genzmehr. The impact of demographics (age and gender) and other user-characteristics on evaluating recommender systems. volume 8092, pages 396–400, 09 2013. ISBN 978-3-642-40500-6. doi: 10.1007/978-3-642-40501-3_45.

Saman Forouzandeh, Mehrdad Rostami, and Kamal Berahmand. Presentation of a recommender system with ensemble learning and graph embedding: A case on movielens, 2020. URL https://arxiv.org/abs/2008.01192.

Giba. Hm ensembling - how to, 2022. URL https://www.kaggle.com/code/titericz/h-m-ensembling-how-to.

HechtJP. Hm eda rule base by customer age, 2022. URL https://www.kaggle.com/code/hechtjp/h-m-eda-rule-base-by-customer-age.

Zheng Heng. time is our best friend v2, 2022. URL https://www.kaggle.com/code/hengzheng/time-is-our-best-friend-v2/notebook.

Ryotaro U. Hiro, Harururu. Hm eda customer clustering by kmeans, 2022. URL https://www.kaggle.com/code/hirotakanogami/h-m-eda-customer-clustering-by-kmeans.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3149–3157, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Lunana. Hm byfone's speed up, 2022. URL https://www.kaggle.com/code/lunapandachan/h-m-byfone-s-speed-up?scriptVersionId=94486600.

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

Bhattacharyya Mayukh. Svd model reranking: Implicit to explicit feedback, 2022. URL https://www.kaggle.com/code/mayukh18/svd-model-reranking-implicit-to-explicit-feedback.

Bagus Mulyawan, Viny Mawardi, and Riyan Wenas. Recommendation product based on customer categorization with k-means clustering method. *IOP Conference Series: Materials Science and Engineering*, 508:012123, 05 2019. doi: 10.1088/1757-899X/508/1/012123.

Nguyentuananh. Lstm/sequential modelwith item features tutorial, 2022. URL https://www.kaggle.com/code/astrung/lstm-sequential-modelwith-item-features-tutorial/notebook.

Mehrbakhsh Nilashi, Karamollah Bagherifard, Mohsen Rahmani, and Vahid Rafe. A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques. *Computers Industrial Engineering*, 109:357–368, 2017. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2017.05.016. URL https://www.sciencedirect.com/science/article/pii/S0360835217302206.

D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, aug 1999. doi: 10.1613/jair.614. URL https://doi.org/10.1613%2Fjair.614.

Sara Rabhi, Wenbo Sun, Julio Perez, Mads R. B. Kristensen, Jiwei Liu, and Even Oldridge. Accelerating recommender system training 15x with rapids. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*, RecSys Challenge '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450376679. doi: 10.1145/3359555.3359564. URL https://doi.org/10.1145/3359555.3359564.

Ayashi Ryo. Hm : Framework for partitioned validation, 2022. URL https://www.kaggle.com/code/negoto/h-m-framework-for-partitioned-validation.

Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 17–22, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450347952. doi: 10.1145/2988450.2988452. URL https://doi.org/10.1145/2988450.2988452.

Morty Tarick. Hnm exponential decay with alternate items, 2022. URL https://www.kaggle.com/code/tarique7/hnm-exponential-decay-with-alternate-items/notebook.

Yue Xiaochen and Liu Qicheng. Parallel algorithm of improved funksvd based on gpu. *IEEE Access*, 10: 26002–26010, 2022. doi: 10.1109/ACCESS.2022.3156969.

Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In *Proceedings of the 37th International ACM SIGIR Conference on Research amp; Development in Information Retrieval*, SIGIR '14, page 73–82, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450322577. doi: 10.1145/2600428.2609599. URL https://doi.org/10.1145/2600428.2609599.

Wen Zhang, Hua Zou, Longqiang Luo, Qianchao Liu, Weijian Wu, and Wenyi Xiao. Predicting potential side effects of drugs by recommender methods and ensemble learning. *Neurocomput.*, 173(P3):979–987, jan 2016. ISSN 0925-2312. doi: 10.1016/j.neucom.2015.08.054. URL https://doi.org/10.1016/j.neucom.2015.08.054.