

## **Executive Summary**

With over 100 thousand products available to H&M customers, it is important that the H&M platform provides personalized recommendations to help their customers quickly find the most relevant items. We joined a community of data scientists on Kaggle to tackle this problem. Our proposed solution involves building a recommender system consisting of (1) candidate generation via collaborative filtering and (2) ranking using a LightGBM algorithm to predict the 12 items best suited for each customer. The initial prototype of the recommender system, described in the following report, can generate data-driven, personalized recommendations to customers based on two years of past data on H&M transactions, customers, and products. In the initial test, the prototype performed successfully with a MAP@12 result of 0.0114. Ultimately, we expect that the proposed solution will help customers better identify products that suit their needs and positively impact the customer shopping experience at H&M.

## **1. Introduction**

On February 7, 2022, data scientists participating in the H&M Personalized Fashion Recommendations Kaggle competition came together with the aim of improving the customer shopping experience. Given two years worth of data on H&M transactions, products, and customers, the participants worked to create data science pipelines, models, and systems to identify the best product recommendations for customers.

This competition was motivated by the fact that shoppers can find it difficult to identify relevant products given the wide selection of offerings at H&M. Thus, providing a set of personalized recommendations improves the shopping experience. Additionally, providing personalized recommendations poses “positive implications for sustainability, as it reduces returns, and thereby minimizes emissions from transportation” [1].

Our personal motivation for joining the H&M Personalized Fashion Recommendation Kaggle competition was threefold: (1) gain an understanding of concepts and developments in the field of recommender systems, (2) participate in a Kaggle challenge, and (3) collaborate with the Kaggle data science community to tackle a meaningful, real-world problem directly impacting consumers worldwide.

In the following sections, we will provide further details on the contents of the dataset (Section 2), the steps we took to process the raw data (Section 3), and our proposed recommender system for generating personalized recommendations (Section 4). We will then provide a discussion of our results, limitations, and future steps followed by a summary of the main conclusions (Section 5 and Section 6 respectively).

## **2. Dataset**

For this competition, the H&M data consists of customers’ purchase history across two years (September 2018 to September 2020) along with metadata on articles and customers. The data also includes images of some of the articles. For the purposes of this project, however, we did not incorporate these images into our analysis.

## 2.1 Features

The tables below describe the content of the three datasets: transactions, customers and articles.

### 2.1.1 Transactions

The Transactions dataset records which products each customer purchased and on what date. Each record contains additional information about the transaction.

| Transactions<br>31,788,324 Rows |           |   |
|---------------------------------|-----------|---|
| Feature                         | Data Type | Description   |
| Date                            | date      | Date of the transaction in the format <i>YYYY-MM-DD</i>   |
| Customer ID                     | char      | ID of the customer who completed the transaction  |
| Article ID                      | int64     | ID of the item that was purchased   |
| Price                           | float64   | Amount that the customer paid for the article   |
| Sales Channel ID                | int64     | Flag indicating whether the item was purchased online or in the physical store. The flag takes one of these two values:<br>1: On-site purchase<br>2: On-line purchase |

Table 1: Transactions dataset

### 2.1.2 Customers

The Customers dataset consists of metadata for each unique H&M customer.

| Customers<br>1,371,979 Rows |           |   |
|-----------------------------|-----------|---|
| Feature                     | Data Type | Description   |
| Customer ID                 | char      | Unique identifier for the customer  |
| FN                          | float64   | Flag that takes one of the following values: <ul style="list-style-type: none"><li>• 1</li><li>• null</li></ul> |

|                        |         |  |
|------------------------|---------|--|
| Active                 | float64 | Flag that takes one of the following values: <ul style="list-style-type: none"> <li>• 1</li> <li>• null</li> </ul>                                 |
| Club Member Status     | char    | Flag that takes one of the following values: <ul style="list-style-type: none"> <li>• ACTIVE</li> <li>• PRE-CREATE</li> <li>• LEFT CLUB</li> </ul> |
| Fashion News Frequency | char    | Flag that takes one of the following values: <ul style="list-style-type: none"> <li>• NONE</li> <li>• Regularly</li> <li>• Monthly</li> </ul>      |
| Age                    | float64 | Customer's age   |
| Postal Code            | char    | Customer's residence postal code   |

Table 2: Customers dataset

### 2.1.3 Articles

The Articles dataset contains metadata for each unique H&M article available for purchase.

| Articles<br>105,541 Rows                              |                |   |
|---|----------------|---|
| Feature   | Data Type      | Description   |
| Article ID  | int64          | Unique identifier for the article   |
| product_code/<br>product_name                         | int64/<br>char | Name and identifier of the article (e.g. Strap top)   |
| Product_type_no/<br>product_type_name                 | int64/<br>char | Name and identifier of the type of article (e.g. Vest top)  |
| product_group_code/<br>product_group_name             | int64/<br>char | Name and identifier of the group of products that the article belongs to (e.g. Upper body, underwear) |
| graphical_appearance_no/<br>graphical_appearance_name | int64/<br>char | Name and identifier of the article's appearance/pattern (e.g. solid, stripe)                          |
| colour_group_code/<br>colour_group_name               | int64/<br>char | Name and identifier of the article's color (e.g. Light pink., beige)                                  |

|   |                |   |
|---|----------------|---|
| perceived_colour_value_id/<br>perceived_colour_value_name   | int64/<br>char | Name and identifier of the article's perceived lightness/ darkness (e.g. Dark, light, bright)                       |
| perceived_colour_master_id/<br>perceived_colour_master_name | int64/<br>char | Name and identifier of the article's predominant color (e.g. Black, white, pink)                                    |
| department_no/<br>department_name                           | int64/<br>char | Name and identifier of the article's department (e.g. Jersey, trousers, kids shocks)                                |
| index_code/<br>index_name                                   | int64/<br>char | Name and identifier of the article's section (e.g. Ladies accessories, menswear, kids size 9-12)                    |
| index_group_no/<br>index_group_name                         | int64/<br>char | Name and identifier of the article's main section (e.g. Ladieswear, menswear, baby/children)                        |
| section_no/<br>section_name                                 | int64/<br>char | Name and identifier of the article's specific section (e.g. Kids and baby shoes, women big accessories, women bras) |
| detail_desc   | char           | Brief text description of the article   |

Table 3: Articles dataset

## 2.2 Data Exploration

In the initial data exploration stage, we studied various aspects including the completeness of the data, the distributions of the categorical and numerical features, and patterns related to the frequency of transactions for each customer over the two-year period.

The analysis of data completeness reveals that the dataset is quite complete with a negligible percentage of missing values in the transactions and articles datasets. The customer dataset contains more missing data. (Over half of the values for the “Active” and “FN” features are missing and there are some empty values related to the age, status, and fashion news habits of the customers.) The techniques with which we have handled missing data are described in Section 3.1.

Additionally, we explored patterns in H&M transactions over the two-year period. As an overview, Figure 1 illustrates the number of transactions that occurred during each week between September 2018 and September 2020. The most relevant finding for the data processing and recommender system, however, is the fact that 50% of customers make no more than 9 purchases

during the two-year window. Additionally, 75% of the customers make no more than 27 transactions during the two-year span. The combined histogram-boxplot in Figure 2 illustrates these statistics. This finding informs decisions we make surrounding data processing and collaborative filtering discussed in future sections.

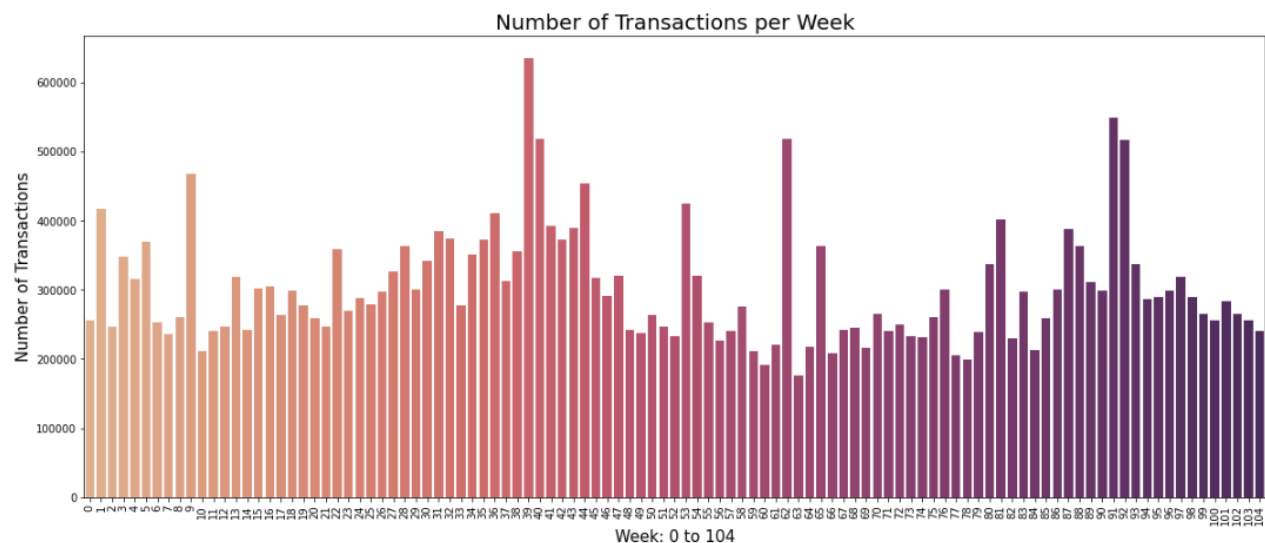


Figure 1: Number of transactions per week over the two-year span of H&M data

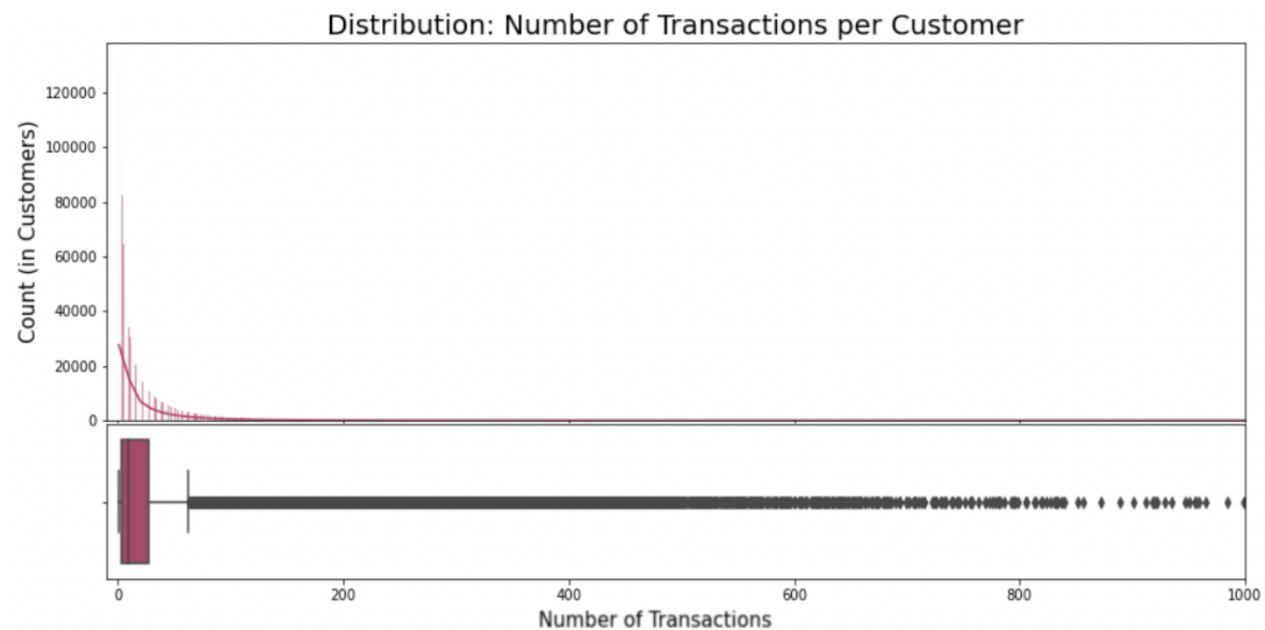


Figure 2: Number of transactions per customer over the two-year span of H&M data

### **3. Data Processing**

#### **3.1 Preprocessing**

The data processing pipeline consists of the following steps:

1. Memory optimization
2. Data imputation
3. Feature generation
4. Data Storage

##### **3.1.1 Memory Optimization**

Due to the large quantity of data on each article, customer, and transaction, memory optimizations are critical for us to build a robust recommender system. First, all the customer and article features have been integer encoded for the purpose of memory optimization and the ranking model described in Section 4.3. Additionally, all the numerical values that can be represented in fewer bits, have been converted into the corresponding data type (e.g. features previously represented as int64 have been condensed to int8).

##### **3.1.2 Data Imputation**

While our dataset was quite complete, the data exploration stage uncovered a few missing values that need to be handled. For features containing a large number of null values (“FN” and “Active”), the entire feature was dropped from the data. For the remaining columns, the missing values were handled in the following manners:

- Numerical Features: Missing values have been substituted with the mean of the corresponding column.
- Categorical Features: Missing values have been substituted with the most common value in the corresponding column.

In future iterations, we could also predict the missing values from the data that is present using supervised learning algorithms. Linear regression can be used to predict the missing numerical features, like age. Classification models like logistic regression can be used to fill the missing values within categorical columns.

### **3.1.3 Feature Generation**

This challenge focuses on making recommendations for the seven-day period immediately after the training data period. Hence, the ability to identify and track the time period of each transaction is necessary for the analysis. For this reason, the *week* column, which represents the week in which each transaction is completed, has been generated in the Transactions dataset. The new *week* feature takes on values from 1 to 104 since the training data contains transactions over a total of 104 weeks. We detail additional approaches for feature generation in Section 5: Discussion, Limitations, and Future Work.

### **3.1.4 Data Storage**

To create an efficient and modular pipeline for our system, the cleaned and processed data is saved in Apache Parquet files. “Apache Parquet is an open source, column-oriented data file format designed for efficient data storage and retrieval. It provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk” [2].

## **3.2 Train, Validation, Test Split**

Figure 3 provides an overview of how the data is split into the train, validation and test sets. For this competition, Kaggle released only the training data and withheld the test dataset to be used by the platform to assess the participants’ submissions. The withheld test set contains the transactions that occurred during week 105 (the seven-day period following the last transaction



within the training set). With this in mind, our train-validation split has the purpose of simulating the conditions in which our project is going to be evaluated.



Figure 3: Train, validation, test split

### 3.2.1 Sample data

Due to the large volume of data, we must subset the datasets that can be used in the recommender system. First, a random sample of 8% of the customers is performed. Then, over this remaining set of customers, only the customers with enough purchase history (more than 15 transactions) are retained. The articles and transactions datasets have been subsetting accordingly.

### 3.2.2 Training data

The training set consists of the data used to build our recommender system. It contains all the transactions that took place from week 1 to week 103.

### 3.2.2 Validation Data

The validation set consists of the data used to evaluate our recommender system. It contains all the transactions that took place in week 104, which corresponds to the last seven-day period of the training data. This split helps us simulate the conditions under which our model is

going to be evaluated. (The evaluation involves comparing our system-generated recommendations with the transactions that took place in week 105.)

### 3.2.3 Test data

The test set consists of all the transactions that took place in week 105. This is the data used by Kaggle to evaluate our submission.

## 4. Recommender System

To generate the set of 12 predicted articles for each customer, we built a recommender system consisting of two main stages: (1) candidate generation for identifying a small number of suitable products for each customer and (2) ranking to determine the top 12 articles to recommend to each customer (Figure 4).

Our approach aligns with the common two-stage recommender system structure, which we describe in further detail in Section 4.1. We describe our process for the retrieval stage via collaborative filtering and the ranking stage using a LightGBM model in Section 4.2 and Section 4.3 respectively. We then conclude with a description of the MAP@12 metric used for evaluating the performance of the recommender system (Section 4.4).

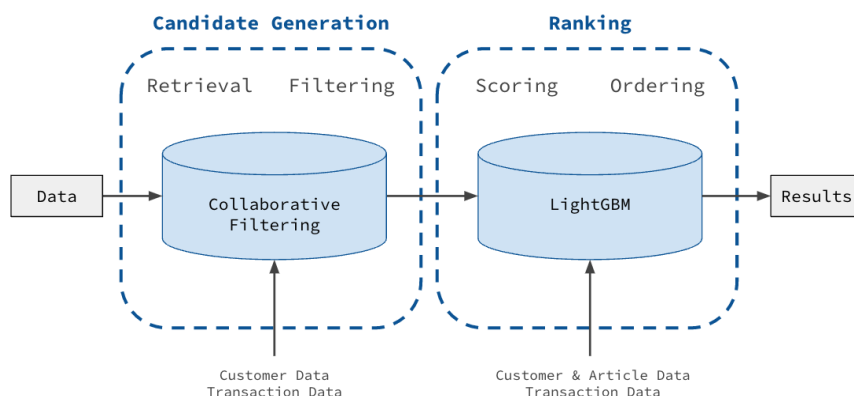


Figure 4: Depiction of the two primary steps of (1) candidate generation and (2) ranking in the proposed H&M recommender system

## 4.1 Recommender System Overview

A recommender system is commonly described as a two-stage process, which can be further broken down into a four-stage process: (1) candidate generation, which consists of retrieval and filtering and (2) ranking, which consists of scoring and ordering [9].

In our H&M recommender system, collaborative filtering is used as the primary mechanism for candidate generation. This process identifies a set of the top products we wish to recommend to each customer for the following week based on trends between various customers and their past transactions. Following candidate generation, potential customer-article pairs are ranked based on the relevance of each specific article for a given customer. For our purposes, a light gradient boosting machine (LightGBM) model is used to score which articles are most relevant for each customer. Lastly, this is followed by an ordering mechanism in which we convert the predictions from the LightGBM model to select and recommend 12 products for each customer.

## 4.2 Candidate Generation

Candidate generation is the first stage of the recommender system. This stage uses collaborative filtering to generate a reduced set of relevant candidates (articles) to recommend to each customer.

Collaborative filtering is a technique that uses customer similarity to identify other potentially relevant items. The idea is to find, for each customer, a number of similar customers based on past transactions that they had in common. To do so, we use the notion of cosine similarity [5]. Then, the system tags as relevant those articles bought by the similar users. Figure 5 illustrates this idea. In Figure 5, we observe that the red trousers are being recommended to User

A. This is because User B, who has similar interests to User A based on their previous purchases, bought those red trousers.

Another common approach for candidate generation includes content-based filtering (Figure 5). This technique focuses on article similarity and recommends other articles similar to those already bought by the customer. Additionally, hybrid techniques combine both collaborative filtering and content-based filtering methods. Further approaches are also discussed in Section 5: Discussion, Limitations, and Future Work.

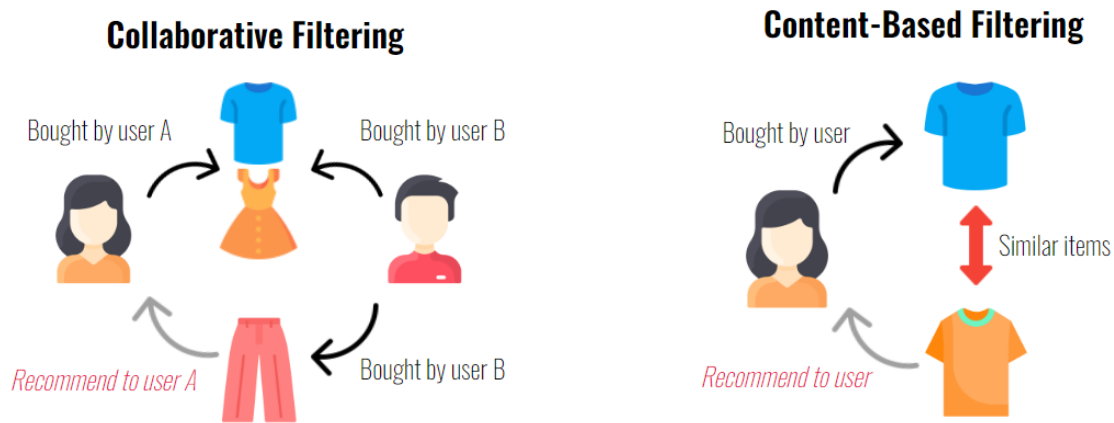


Figure 5: Collaborative Filtering and Content-Based Filtering

At the end of this stage we have a subset of potential recommendations (articles) for each of the customers, which are ready to be used in the next phase of the recommender system: the ranking task.

### 4.3 Ranking

The task of ranking to identify the top 12 product recommendations for each customer differs from the more common classification and regression tasks. Thus, the input, machine learning model, and output all differ slightly from those seen previously in our program. Our

approach ultimately uses a Light Gradient Boosting Machine (LightGBM) algorithm to perform the ranking. The inputs, outputs, and LightGBM approach are detailed in the following sections.

4.3.1 Ranking Model Input and Output

Figure 6 demonstrates the format of the input for our particular ranking task involving H&M articles. The input features are the unique product identifier, product features, and customer features. In our case, all categorical input features are integer-encoded as this encoding typically performs better than one-hot encoding when using the LightGBM algorithm [3].

The target variable takes on two values (1 or 0). In this case, the target is 1 for all instances where the given customer did in fact purchase the given item. Meanwhile, the instances with a target of 0, the negative candidates, correspond to the *potential* customer-item pairings generated during the candidate generation process. The notion of potential customer pairings is defined as those customer-item pairs where the product was identified as being potentially relevant to the given customer, but the customer did not ultimately purchase said item.





| Article   |                          | Article Features |     |                    | Customer Features |     |          | Target |
|---|--------------------------|------------------|-----|--------------------|-------------------|-----|----------|--------|
| Article ID  | Product                  | Color            | ... | Department         | Age               | ... | Zip Code |        |
|  | Comfy Winter Hoodie      | Pink             |     | Children Size 8-10 | 14                |     | 32965    | 1      |
|  | Florida Button Down      | Red              |     | Menswear           | 75                |     | 17564    | 1      |
|  | Youthful Summer Sundress | Yellow           |     | Ladieswear         | 35                |     | 02481    | 1      |
|  | Florida Button Down      | Red              |     | Menswear           | 54                |     | 18930    | 0      |

Figure 6: Input of the Ranking Task

Unlike a classification task, however, the output is a numeric value corresponding to each customer-item pairing, where the value indicates how relevant the item is for the given customer. An illustration of the outputs are illustrated in Figure 7. In this case, the output of the model or prediction can take on both positive and negative values, where higher positive values indicate items of greater relevance to the respective customer and more negative values represent items that are less relevant to the respective customer.









| Customer  | ... | Article   | .... | Prediction |
|---|-----|---|------|------------|
|    |     |    |      | 0.2556     |
|    |     |    |      | -0.0432    |
|    |     |    |      | 0.8971     |
|  |     |  |      | 0.0011     |

Figure 7: Output of the Ranking Task

### 4.3.2 LightGBM and Model Training

To perform the ranking task, we used a LightGBM or “Light Gradient Boosting Machine” algorithm. As indicated in the name, the LightGBM algorithm uses gradient boosting to build an ensemble of models, where predictors are added sequentially and each successive predictor works to minimize the errors made by the former. Additionally, LightGBM is a tree-based learning algorithm known for its efficiency and fast training speed, hence the reason we selected this approach when working with the large dataset of customer, transaction, and article information [4].

Then, during the training process on the 7965152 training instances, the LightGBM algorithm grows decision trees, identifying the optimal features and thresholds to split at during each stage of the iterative training process. Figure 8 depicts one such decision tree that results from

the training on the H&M data. From this approach with LightGBM, we also obtain the feature importances which indicate that the Article ID along with other article-related variables are the most important features in the ranking process as illustrated in Figure 9.

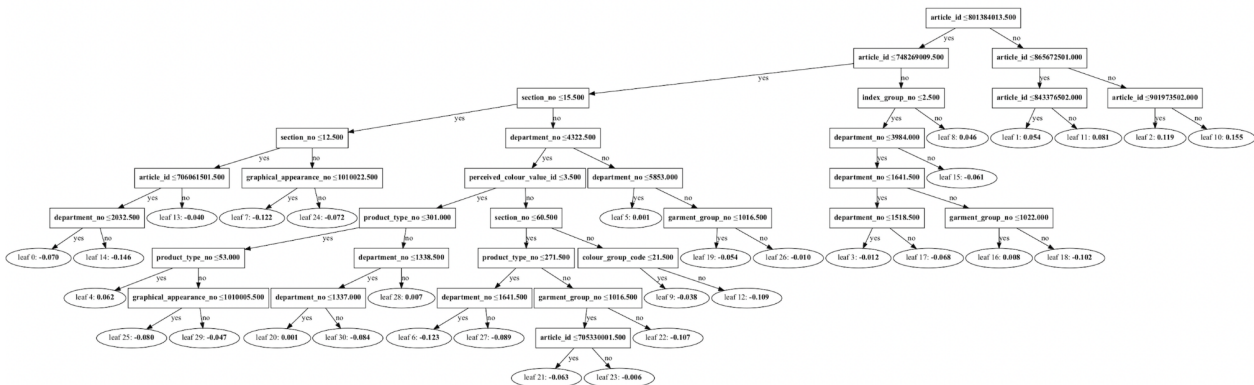


Figure 8: A decision tree constructed from the H&M data during the LightGBM training process

Ultimately, the LightGBM algorithm provides predictions, which in our case range from -0.146 to 0.155. As noted previously, these predictions indicate the relevance of each customer-product pairing and higher positive values indicate items of greater relevance to the respective customer.

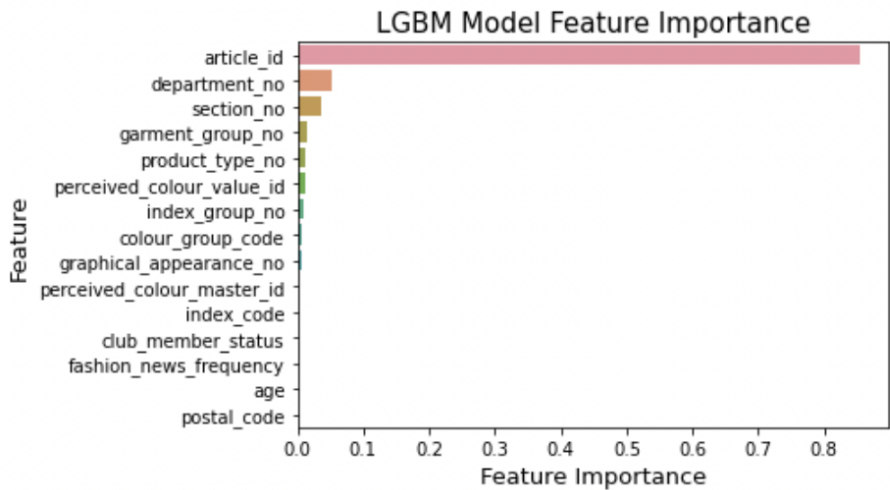


Figure 9: Importance of each training features during the LightGBM training process

### 4.3.3 Recommendations

The predictions from the LightGBM ranking process indicate the relevance of each customer-item pair. In our case, for each customer, we recommend the 12 articles that have the highest customer-article prediction. Alternatively, if the priority was to recommend a variety of products to H&M customers, we could select the top three products across different departments. With no specifications provided, however, we opted for the approach to recommend the 12 highest scoring, or most relevant, products overall for each customer.

### 4.4 Evaluation Metric

The metric used by Kaggle to evaluate the submitted models is **MAP@12**. MAP stands for “Mean Average Precision” and @12 stands for “at 12”. This means that the metric is evaluating 12 predictions (articles) for each customer. Generally, this metric is denoted as MAP@k, where k is the number of elements to be evaluated.

As the name suggests, the Mean Average Precision is derived from the Average Precision (AP). First, we compute the AP at a threshold k. Then, we find the sum and take the mean of AP@k to get the final MAP@k [8]:

$$mAP@k = \frac{1}{N} \sum_{i=1}^N AP@k_i$$

Figure 10 can be used to explain this concept.



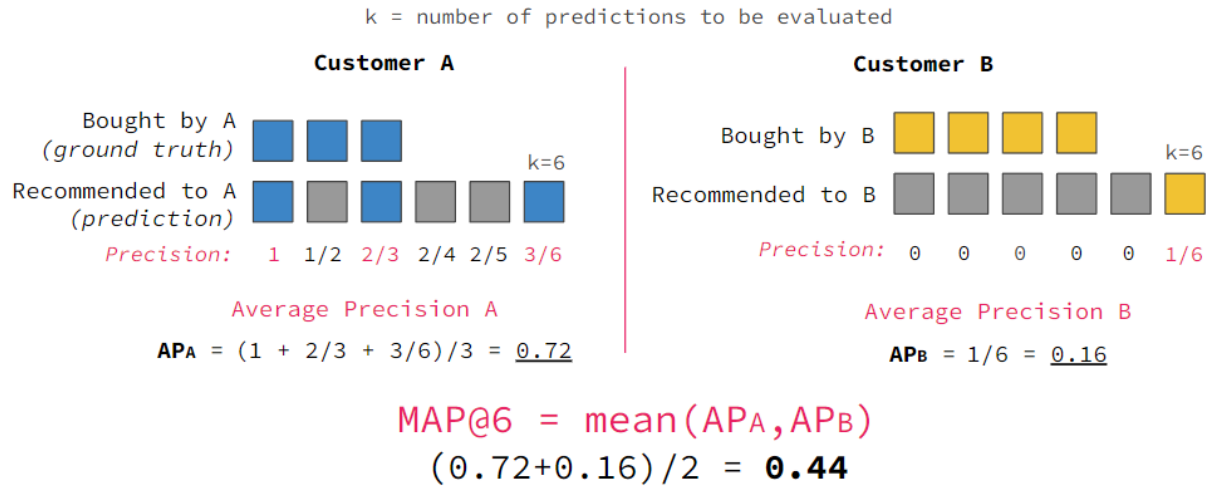


Figure 10: Illustrative example of a MAP@k calculation

In Figure 10, we have two customers A and B:

- Customer A buys the blue items during the seven-day testing period (week 105). This denotes the actual purchases and is called the ground truth.
- The model recommends 6 items to A, in the order that is shown above. The blue items correspond to good predictions (Customer A did buy those items) and the gray items correspond to bad predictions (Customer A did not buy those items).
- The Average Precision for customer A (AP.A) is calculated as the mean of the individual precisions for each true prediction (blue items).
- Average Precision for customer B (AP.B) is calculated the same way.
- Our metric, in this case MAP@6, is the mean between AP.A and AP.B.

As the example illustrates, the number of items to be evaluated as well as the order of the predictions matters. (For example, in our example had the correctly predicted item for customer B been the first recommendation, the value of AP.B would be 1 instead of 0.16.)

## 4.5 Results

Table 4 shows the performance of the recommender system on the training, validation and test sets using the MAP@12 evaluation metric.

| Data       | MAP@12 |
|------------|--------|
| Training   | 0.0072 |
| Validation | 0.0237 |
| Test       | 0.0114 |

Table 4: Performance of the recommender system

Compared to results we have observed for other machine learning tasks thus far in the program, which tend to achieve scores of 0.7 and above, these scores may appear shocking. So, why do we obtain these low results from our recommender system?

It is important to remember that the order of the predictions matters for the MAP@k metric. It is likely, however, that our recommender system does not recommend all the articles the customer truly bought and that some of our recommendations are misordered. One potential reason could be because we are not restricting our recommendations to those articles that have not already been purchased by the customer. In other words, our recommender system may only be recommending articles that the customer has already purchased rather than those a customer has never purchased. This could ultimately exclude the items that should be recommended from the twelve-item predictions and lower the performance of our recommender system.

Another factor that may contribute to the model performance is the fact that we only use 8% of the customers to build the recommender system. Ultimately, however, the performance of the recommender system is being tested on Kaggle's test set which contains 100% of the customers in the dataset.

For reference, Figure 11 shows the top 6 scores achieved in the Kaggle competition.

■ Prize Winners




















| # | △   | Team                 | Members   |   | Score   | Entries | Last | Code |
|---|-----|----------------------|---|---|---------|---------|------|------|
| 1 | —   | senkin13 & 30CrMnSiA |     |  | 0.03792 | 106     | 8d   |      |
| 2 | —   | hello world          |     |  | 0.03741 | 99      | 8d   |      |
| 3 | ▲ 1 | sirius               |    |  | 0.03623 | 78      | 8d   |      |
| 4 | ▼ 1 | Hongwei Zhang        |    |  | 0.03563 | 89      | 9d   |      |
| 5 | —   | HAO                  |      |  | 0.03553 | 103     | 8d   |      |
| 6 | —   | Hard2Rec             |     |  | 0.03529 | 148     | 8d   |      |

Figure 11: Kaggle Competition Top-6 scores

## 5 Discussion, Limitations, and Future Work

The recommender system proposed in Section 4 is but one approach for predicting products H&M customers may purchase in the following week. While it serves as a decent model for recommending products, we have been fortunate to learn of other potential approaches through discussions on Kaggle following the close of the competition on May 9, 2022. Many of these discussions expand on additional approaches that could be used for feature engineering, candidate generation and ranking. The most notable discussions covered:

1. Approaches to candidate generation that incorporate product popularity over different periods of time (e.g. within the last week, the last month, and the last season). These approaches appeared to be some of the most effective methods for candidate generation as they handle the fact that fashion trends are season dependent and have the tendency to change quickly.

2. Approaches for assessing the similarity between items like item-item collaborative filtering and image processing techniques. These approaches were commonly used to enhance the feature engineering and candidate generation stages of the pipeline.
3. Approaches for incorporating customer-related features that highlight the likelihood that a customer will repurchase items [6, 7].

As many of the discussions highlight, the possibilities for candidate generation and feature engineering are endless. They are also thought to contribute most significantly to the recommender system's overall performance.

Through our development process and reflection on approaches other participants have taken, we have also identified approaches to address some potential limitations of our system:

1. Integrate better feature engineering and techniques to address the cold start problem. This is particularly applicable because the majority of the customers make only a small number of transactions. Thus, it is important going forward to consider how to make predictions for these customers even if we have little information on their past transactions.
2. Extend the machine learning pipeline to incorporate downsampling the negative samples and expand the cross validation and hyperparameter tuning that the model undergoes. Downsampling is particularly important in this problem because the collaborative filtering generates numerous potential articles for each customer. Thus, the number of negative samples is significantly larger than the number of positive samples.
3. Increase the scalability of the recommender system. At this time, we use a sampling scheme as explained in Section 3.2 as the collaborative filtering can handle only a subset of the data related to customers and transactions. This restricts the amount of data that can be

used when building the recommender system and, thus, limits how well the collaborative filtering and ranking model can perform.

Going forward, in addition to expanding the recommender system to consider the approaches and limitations discussed above, future steps should focus on incorporating computer vision techniques to process the provided image data and focus on text processing or natural language processing techniques to gather additional insights from the article descriptions.

## **Conclusion**

Our proposed recommender system, aimed at providing personalized product recommendations to H&M customers, has yielded promising results with a MAP@12 score of 0.0114. Additionally, through building this system we have had the opportunity to gain an understanding of core concepts surrounding recommender systems, compete in a Kaggle competition, and connect with a community of data scientists to tackle a meaningful, real-world problem. By incorporating the additional refinements to the feature generation and candidate generation stages of the recommender system as discussed in Section 5, we believe the proposed approach has the potential to improve the H&M shopping experience for consumers worldwide.

## GitHub Repository

All the code for the proposed recommender system is publicly available in the GitHub repository:

[https://github.com/msmillan7/Kaggle\\_Competition-HM\\_Recommender\\_System](https://github.com/msmillan7/Kaggle_Competition-HM_Recommender_System).

## References

1. [H&M Personalized Fashion Recommendations Kaggle Competition Overview](#)
2. [Apache Parquet](#)
3. [LightGBM Documentation: Advanced Topics](#)
4. [LightGBM Classifier in Python](#)
5. [Cosine Similarity, the metric behind recommendation systems](#)
6. [H&M Personalized Fashion Recommendations Kaggle Competition: Discussion of 1st Place Solution](#)
7. [H&M Personalized Fashion Recommendations Kaggle Competition: Discussion of 2nd Place Solution](#)
8. [How mean Average Precision at k \(mAP@k\) can be more useful than other evaluation metrics](#)
9. [Keynote 7: Moving Beyond Recommender Models - Even Oldridge \(NVIDIA\), Karl Byleen-Higley \(NVIDIA\)](#)
10. [H&M Personalized Fashion Recommendations Kaggle Competition Starter Pack](#)
11. [Building a Product Recommendation System with Collaborative Filtering](#)
12. [Exploring Data Splitting Strategies for the Evaluation of Recommendation Models](#)
13. [Setting Goals and Choosing Metrics for Recommender System Evaluations](#)