

Building and Validating a Recommendation system using Python and SAS

Patralekha Bhattacharya, Jigyasa Analytics, LLC

ABSTRACT

Recommendation systems have become increasingly popular in many industries, such as retail, movies, books, etc. In this paper, we build and validate a recommendation system for physical fitness products (e.g., exercise equipment, training manuals, books, etc.). All the recommendation models were built using implicit data, as we do not have explicit consumer ratings for the products. Instead, we use past user behavior (or purchases) as a proxy for user preferences whenever such past data is available. Our dataset consists of customers who purchased a fitness product during the year 2022 or earlier, as well as those who have never purchased any of the products in the past. Customers were divided into three groups: (i) non-purchasers, (ii) recent purchasers, and (iii) older purchasers, and different modeling methods were used to recommend items to different sets of customers. For the final solution, we used a hybrid system that used the strengths of both Python and SAS to develop the best set of recommendations for each customer.

INTRODUCTION

Consumers are often bombarded with excessive choices when buying products or services, making the purchase decision difficult. Traditionally, before the advent of the digital age, word-of-mouth advertisements from families and friends would serve as the only available recommendation system. The justification for this was that if your neighbors, friends, and families find a product pleasant or valuable, chances are you will also like it. Consumer reviews printed in newspapers and magazines were also helpful because they were from a trusted source. Today, in the information age, companies can collect and store large amounts of data on their customers' purchase behavior. Before long, they realized that this could be used to create a kind of "word-of-mouth" recommendation system where the recommendation does not originate necessarily from someone who has a close relationship with the customer but from someone who still may be similar to that customer in terms of their past purchases and preferences.

This is the foundation for the different types of recommendation systems that exist today, which can be broadly classified into three distinct types (Deepjyoti Roy & Mala Dutta, 2022): (i) content-based recommendation systems, (ii) collaborative recommendation systems and (iii) hybrid recommendation systems.

In content-based recommendation systems, the product content (or, in other words, the product's attributes) is used to classify products, each group containing products with similar characteristics. Examples are movies, which can be categorized by genre, director, lead actor, etc. Suppose the user purchases one of the products in a group. In that case, they will likely also enjoy other products from the same group; therefore, they can be recommended. For instance, other horror movies can be recommended if a user watches a horror movie.

In collaborative filtering recommendation systems, what other users have purchased becomes more important than the product's attributes. Collaborative filtering systems can be divided into two types: (i) User-User collaborative filtering and (ii) Item-Item collaborative filtering.

In a User-User collaborative filtering approach, a particular user's past purchases and preferences are used to find other users with similar preferences. The products bought by these other users that our specific user has not yet purchased will be recommended to this particular user.

In the Item-Item collaborative filtering approach, a particular item purchased by a user is used to find other items that may have been most frequently purchased along with it. These items can then be recommended to this user.

Each recommendation system has limitations; some work better than others in different scenarios. Therefore, it is helpful to use a hybrid system where two or more of these recommendation systems are combined to provide customers with better and more exhaustive suggestions.

In this paper, we tried to combine various models and techniques to build and validate a recommendation system for fitness products.

Another pertinent point is the difference between explicit and implicit ratings. Sometimes, companies encourage customers to provide a rating for the item they have purchased. This rating is usually on a predefined scale of 1 to 5 (or 1 to 10), with a rating of 1 meaning that they thoroughly disliked the product and a rating of 5 (or 10) meaning that they loved it. The customer may provide additional comments or reviews in addition to this rating. Other customers can then view these ratings and reviews to get an idea of whether the product is a fit for them. Product ratings and reviews on Amazon.com are one example of this. These are called explicit ratings, as the customer explicitly provides a numerical measure of their preference for the product. When such explicit ratings are available, they can be used directly in the recommendation models. However, not all customers comply with providing ratings; in fact, most customers who purchase products do not offer ratings for these products. On the other hand, companies typically have access to the buying behavior of their customers, and we assume that by purchasing a product, the customer spontaneously reveals their preference for that product. These are called implicit ratings, and since they are much more readily available to companies, they are most often used to build recommendation systems.

DATASET AND METHODOLOGY

Our data consists of customers who are past or current subscribers to a fitness magazine and have been sent promotional emails to purchase some fitness products. Some of these customers have already purchased these fitness products during the year 2022 or earlier, whereas some others have never purchased any of these products. The data contains information on each customer, their `customer_id`, and their purchase history for the last few years: what products were purchased (the product SKU and product description) as well as the date of purchase. We divided the data into three groups: (i) Recent purchasers (those who purchased in the past year, i.e., 2022 only), (ii) Past purchasers (those who purchased in the year 2021 and earlier), (iii) Those who have never purchased in the past. All models were built using data from 2022 and earlier. We also had promotion history for the first quarter of 2023 (i.e., emails sent to customers and the dates of these emails) as well as purchase history for these individuals for the first four months of 2023, and we used this data to do out-of-time testing and validate the results of the models.

We tried several different models and validated each to see which performed best in the out-of-time samples. Since the customers did not rate any of the products explicitly, their purchase behavior was used as implicit ratings. Eventually, we decided to use the models which performed best in validation and implemented them.

An important factor that influenced the performance of our models is that due to certain technological constraints, only a small subset of all products could be recommended to customers. In other words, even though customers had purchased many different products in the past, only a small fraction of them could be recommended.

ITEM-ITEM SIMILARITY MODEL

One of the first models we built was an Item-Item similarity model. As discussed above, in this collaborative filtering approach, a particular item purchased by a user is used to find other items that may have been most frequently purchased along with this item. A similarity score is assigned to each item that has not yet been purchased by the customer. We use the Cosine similarity function in Python to obtain similarities between items. For each item a customer has purchased, we identify other items with the highest similarity score to this item. If the customer has purchased two or more items, then the similarity scores are combined to obtain a recommended rank for each item for each customer. For each customer, the items he is yet to purchase are ordered according to the recommended rank.

CONTENT-BASED MODEL

Another model we tried was a content-based model, where we used product descriptions and feature information that were available to us. These characteristics of the different products were used to identify other products with similar features. Since many of the product features were described using words, we used text analytics methods to obtain a numerical representation of those words. More specifically, we used the TF-IDF vectorizer method, which identifies the importance of each word in a sentence or document. TF refers to Term Frequency, which measures the number of times a word appears in a document or a sentence. We assume that the greater the number of times a word appears in a document, the greater the importance of that word in that document. Document frequency identifies the number of documents containing the word. Common words (such as the, if, then, etc.) usually appear multiple times in almost every document. So, if a word is prevalent in various documents, we reduce its importance. Therefore, TF-IDF for a word increases proportionally to the number of times that word appears in the document but is counterbalanced by the number of documents that contain that word. This method was used to identify the most relevant words for each product and create a vector representing the essential words for each product. After that, a cosine similarity function was used to calculate a similarity measure between these product vectors. This measure identifies how similar the products are to each other. For each product purchased by a customer, we ranked other products in terms of highest similarity to this product. Therefore, each product yet to be purchased by the customer is assigned a similarity score. Suppose a customer has purchased multiple products in the past. In that case, all the other products (not yet purchased by this customer) get different similarity scores. All these scores are combined to obtain each product's final overall similarity score. Products yet to be purchased by the customer can then be rank ordered by descending overall similarity score to get the list of recommended items for that customer.

TRIGGER STRATEGY MODEL

This recommendation model was based on joint purchases of 2 or more products. We identified all instances where the same customer purchased two or more products within a specific time frame of each other. The time window we considered was within 30 days, and we flagged these records as joint purchases. For instance, if a customer first purchased product A and then purchased product B within 30 days, we flag this combination as a joint purchase. We examined all customers and identified the number of times these two products were jointly purchased. We also identified the total number of times all customers purchased product A. Then, we divided the joint purchases of products A and B by the number of times product A was purchased to get the joint purchase rate for products A and B. If this joint purchase rate of products A and B is high, then we implement that as a trigger strategy, and every time product A is recommended to the customer, product B is also recommended to them.

LIGHTFM MODEL

Another model we tried is the LightFM model, which is available as a Python package. This model represents users and items as latent vectors (called embeddings). It is assumed that each user can be represented by a k -dimensional vector, U , and similarly, each item can be represented by a k -dimensional vector I . Both U and I are, therefore, vectors of dimension $k \times 1$, and the elements of the vectors are called latent factors. Each latent factor of a particular user can then be multiplied with each latent factor for a specific item, and the sum of all of these products represents that user's preference (or rating) for this particular item. In other words, the user's rating for an item is obtained by calculating the dot product of the item vector on the user vector.

Once we have the ratings for all items by all users, we compare these ratings to what we already observe in the dataset and adjust any incorrect ratings by changing the values of the latent factors. We continue in this manner until we end up with optimal ratings that match what we observe. Several types of optimization methods can be used to come up with best values for these latent factors. Since our goal is to predict a customer's ranking for a set of items, we prefer to use a Learning-to-Rank optimization method (which is available in the LightFM module). In these optimization methods, the goal of the optimization exercise is to identify the correct ranking between each pair of items for each user (rather than focusing on obtaining the correct rating for each item). When the user purchases an item, we assume that the user ranks this item higher than one they have not purchased. The Learning-to-Rank optimization method is set up to adjust the latent factors to ensure the ranking of items for each user is

maintained when mapped back to the original rating matrix. Model training is accomplished via Stochastic Gradient Descent.

The LightFM module implements two Learning-to-Rank Optimization Methods. One such method is the Bayesian Personalized Ranking (BPR) method, where an adjustment to the latent factors is made every time there is a difference between the rating for a positive item (i.e., one that the user has purchased) and that for a negative item (i.e., one that the user has not purchased). Another method is the Weighted Approximate-Rank Pairwise (WARP) method, where an adjustment to the latent factors is made only if the rating for a positive item is lower than the rating for a negative item. We tried both methods and found that the WARP optimization method worked better.

In addition, LightFM allows user and item features to be incorporated within the recommendation system using additional latent vectors for each feature. While we did not have any extra features for the users (or customers) available in our datasets, we did have access to item features. We tried the LightFM model, both with and without additional item features. We found that the model performed better when we did not use the additional item features, and therefore, we decided not to use extra features in our model.

POPULARITY MODEL

Many customers in our dataset did not purchase any products, so we did not have any purchase history for them. We also only had access to three years of purchase history, so there may have been customers whose last purchase was earlier than three years, so we could not observe their purchase history. Without data on past purchases, none of the above models could be run for these customers. Therefore, we decided to develop recommendations for these customers using a different method. We looked at the most popular products purchased in the last 12 months and tried to recommend those products to these customers. However, we noticed that over time, the popularity of a product may decline, and other products take its place. Over the 12-month window, some of the most popular products may have had huge sales in the first six months and may experience reduced sales in recent months as the market for these products becomes saturated. Often, new products were introduced, and they sometimes became very popular within a span of 3 to 4 months. Therefore, instead of using only the last 12 months, we decided to identify the most popular products over multiple periods and created an average rank for products, which helped us build a unique list of the most popular products. Customers were sent recommendations from this list of popular products.

VALIDATIONS

As mentioned before, we could only select recommendations out of a small set of selected products due to certain technological constraints. Therefore, even though we used the customers' historical purchases of all products, the models output recommendations for only the selected list of products.

We were only required to recommend three products to each customer, so we had to identify which three products would be the best recommendations for each customer. Since we used multiple models and had a top list of recommendations from each model, the next thing to figure out was how to combine the results of these different models to get a unique list of three recommendations for each customer.

We used data from the first quarter of 2023 to validate our models. Email recommendations had already been sent out to customers during this quarter (before our models were implemented). We looked at how many of those recommendations had resulted in a purchase. We compared this information to the recommendations output by our models and used this to validate each model.

We found that different combinations of models worked best for various customer groups. For instance, for the Recent_Purchasers, we used a hybrid approach and combined the top-ranked products from the Content-based and LightFM models. Trigger strategy recommendations were not always available, but they superseded other models when available.

NEW PRODUCTS

Sometimes, new products were introduced into the market for which no purchase history was available. In this case, we used a content-based model where we used several product features and product

descriptions to identify similar products. For a new product, all other similar products were ranked in terms of similarity to this new product. Then we identified customers who had purchased these similar products in the past and recommended the new product to those customers. We looked at the recommendations sent to and corresponding purchases for a recently introduced new product to validate this model. We looked at people who were recommended this new product and classified them into two groups based on whether they had earlier purchased the top N most similar products. We found that when a new product is introduced, targeting those who purchased the topmost similar products is likely to produce a significantly higher response rate than targeting those who purchased other products.

SOFTWARE USED

We used both SAS and Python for the analysis done in this paper. The datasets that we worked with were quite large. So, we used SAS to do much of the initial data processing and the data manipulations and calculations needed for the final validations. We also used SAS to build the Trigger strategy model and the Popularity model. For the other models, we used Python as it has some inbuilt packages which we could use. In this manner, we were able to use the relative flexibility of Python with the data-handling strengths of SAS.

CONCLUSION

We built several types of models to generate product recommendations for the customers and validated all of them using the available out-of-time sample. Eventually, we took a hybrid approach and combined the top recommendations from the different models to develop the final recommendation strategy.

REFERENCES

- Deepjyoti Roy & Mala Dutta, 2022. A systematic review and research perspective on recommender systems Journal of Big Data volume 9, Article number: 59 (2022). <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00592-5>
- Kula, Maciej. 2015. Metadata Embeddings for User and Item Cold-start Recommendations, arXiv:1507.08439v1 [cs.LR]. <https://arxiv.org/pdf/1507.08439>
- Mtetwa, N., Awukam, A. O., & Yousefi, M. (2019). Feature extraction and classification of movie reviews. <https://doi.org/10.1109/ISCM.2018.8703235>
- Rosenthal, E. 2016, November 7. Learning to rank Sketchfab models with LightFM. Retrieved from <https://www.ethanrosenthal.com/2016/11/07/implicit-mf-part-2/>
- Shu, Amanda. 2021. LightFM Performance: Evaluating LightFM's pure and hybrid recommendation models. <https://amanda-shu.medium.com/lightfm-performance-7515e57f5cfe>
- Sun, A. and Peng, Y., 2022. A survey on modern recommendation system based on big data. *arXiv e-prints*, pp.arXiv-2206. <https://arxiv.org/html/2206.02631v2>

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Patralekha Bhattacharya

Jigyasa Analytics, LLC

E-mail: pbhattacharya@jigyasaanalytics.com , patrab@gmail.com

www.jigyasaanalytics.com