CS 421 - Project Report

MARKET BASKET ANALYSIS FOR INSTACART

BU WENDE
JIANG HANYU
QI HAODI
WANG YIXING
ZHANG CHENGZI
School of Information System,
Singapore Management University

November 22, 2020

1. Introduction

1.1 Overview

Instacart is an American company that provides online grocery shopping. It released a dataset that contains a sample of over 3 million grocery orders from more than 200,000 users. This papers illustrates the use of machine learning techniques to predict the products that will appear in the users' next order. The models used include XGBoost for reordered products, TF-IDF and matrix factorization methods with co-clustering for new products. With the use of our models, Instacart can offer more personalized product recommendations to users, and therefore achieve a greater sales revenue.

1.2 Data Description and Splitting

In the original dataset by Instacart, the csv files used were,

1. Order products (order id, product id, add to cart order, reordered)
2. Orders (order id, user id, evaluation set, order number, order day of week, order hour of day, days since prior order)

We merged the two csv based on order id and split the dataset as follows,

- Train set: First till third last order
- Validation set: Second last order
- Test set: Last order

1.3 Evaluation Metrics

Mean Average Precision (MAP) MAP allows us to calculate the Average precision (AP) for both user's order list level and model level. Please refer to Appendix 1 for the illustration.

2. Predict Repurchased Products

2.1 Overview

Users are likely to repurchase products due to their short-term spending habits. The goal of this model is to predict products that users will repurchase in the next order based on their order history. Given the large data size and complex patterns, the parallel tree boosting in XGBoost provides a fast and accurate predictive model that is also scalable for E-commerce companies, such as Instacart.

2.2 Feature Engineering and Selection

The original dataset is at order level and we aggregated them to user-product level as described in part 1.2. The following categories of features were generated for modelling.

2.2.1 Total Buy & Ratio

Total Buy calculates the number of times a user bought a product in his/her previous order(s). There are six different such features that look into an user's recent 3 to 7 orders and all previous orders. This allows us to understand the short-term spending behaviour. The Total Buy Ratio is the percentage of purchases of a product over the number of N recent orders.

2.2.2 Period without repurchase

Period without repurchase calculates the period between orders that a user repurchase an item. There are four features under this category, the **longest** period without repurchase the item in recent 3, 5 and all orders; and **median** period for all previous orders.

2.2.3 Chance & Ratio

Chance refers to that since an user's first order of this product, how many orders (with or without the product) have been made thereafter.

Chance ratio is the percentage of orders that contains the product after first order of the product over chance. Chance and chance ratio are calculated in recent 3 and 5 orders. These features capture the probabilities of user repurchase certain product in the order history.

2.2.4 Feature Selection

We conducted feature selection based on correlation matrix and feature importance scores in terms of gain, weight and cover (Appendix 2 and 3). This allows us to interpret the importance and select the most features accordingly. For the top three features, which are total buy, its ratio, and order ratio by chance, all of them relate to the recent 5 orders. This implies an underlying purchasing habits of users that best captured by the recent 5 orders. We have plotted a correlation matrix graph to look at the correlation among all the features created. The majority of the features were unrelated which increases the diversity of analyses. Although total buy features and their ratio are highly correlated, we will still keep it as ratio provides us with additional explanatory power. Hence, we decided to maintain all features as all of them capture unique data information on the user-product level.

2.3 Model Training and Evaluation

The dependent variables are the features at the user-product level. The data label is reorder (0 for no | 1 for yes). We used the reorder in recent five orders to proxy the target variables.

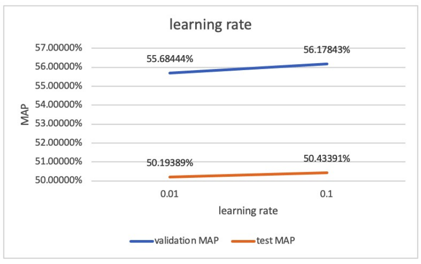
Identifiers are dropped before model training and appended back for evaluation purpose.

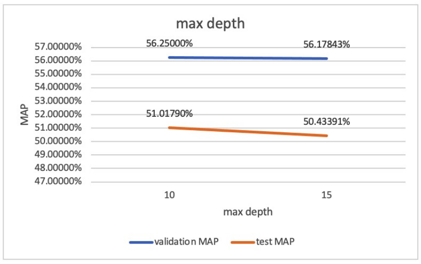
2.3.1 Model Validation and Test

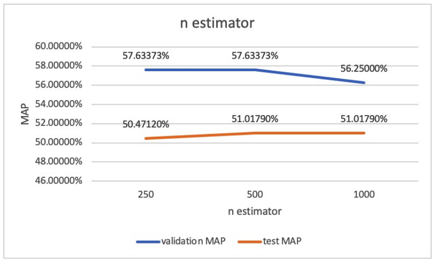
We evaluated the model using Mean Average Precision (MAP) for 500 users who repurchase products. Top five predicted products from the model are selected and compared with the list of actual reordered product. Each product can only appear once for a user.

XGBoost model is prone to overfitting. Thus, we tuned the following hyper parameters for regularisation in order of the following,

1. **Learning rate** - the lower the learning rate, the more robust the model will be in preventing overfitting. Learning rate = 0.1 harvest a better result.
2. **Max depth** - the depth of a tree. Max depth = 10 harvest a better result.
3. **N estimator** - the number of trees to fit. N estimator = 500 harvest a better result.







The final model configuration is in Appendix 4. The model performance are as follows,

- Validation MAP: 57.63%
- Test MAP: 51.02%

This means that overall, the model is able to predict on average every second reordered products correctly for each user.

2.3 Future Work

2.3.1 Time-series Data Handling

Given the nature of order history is a time-series one, we have to perform data aggregation to the user-product level while maintaining the time sensitive data through feature engineering. In future, a time series model can be developed to capture all time-sensitive information.

2.3.2 Prediction of None

It is possible that users do not repurchase any product in the next order. In future, a predictive model for the None case can be developed to make the model more comprehensive.

3. Predict New Products with Collaborative Filtering

3.1 TF-IDF

3.1.1 Model Representation

TF-IDF(term frequency - inverse document frequency) is used to identify similar users for product recommendation. Terms represent products and documents represent user orders. The TF-IDF score is defined as follow,

$$\text{TF-IDF Score}(i, j) = tf_{i,j} \cdot log(N/df_i) \tag{1}$$

Where $tf_{i,j}$ is the number of times user j purchased product i; df_i is the number of users who purchased product i; N is the total number of users.

For each user, top K similar users are identified based on cosine similarity scores. All the products in these K users' orders are then ranked based on their popularity. Popularity is defined as the number of users who purchased the item among the K similar users. Top N popular products would then be recommended to the target user.

3.1.2 Evaluation

Hyperparameter Tuning			
Parameter	Meaning	Possible Values	Best Value
K	Number of similar users	[1, 10] with a step of 1	10
N	Number of products recommended	[9,10,11]	10

Model performance with different hyperparameters on the validation set is shown in Appendix 5. Although when $k = 10$ has the best value, considering computational resources, $k = 4$ is chosen. When $k = 4, N = 10, \text{MAP} = 32.4\%$.

3.1.3 Limitations

The average cosine score between the target user and his top 10 similar users is 0.23, which means that the average angle between them is 76 degrees. This explains the low MAP score of the model, as users are not very similar to each other, the product recommendations based on items purchased by similar users would not be accurate. Better popularity measures that incorporate the cosine similarity scores among users and product TF-IDF scores may be employed to capture more relevant products.

3.2 Probabilistic Matrix Factorization (PMF)

3.2.1 Model Representation

To approximate our original matrix R, which contains M products and N users, let R_{ij} represent the number of purchase of user i for product j, $U \in \mathbb{R}^{N\times K}$ and $V \in \mathbb{R}^{M\times K}$ be latent factor matrices. [3] In our Instacart case, we normalize the R matrix by mapping the purchased quantity $1,\ldots,K$ to the interval $[0,1]$ by using $(x-1)/(K-1)$ to avoid non-converging issue. Moreover, instead of using a linear-Gaussian model, we let the dot product of user and product factor vectors pass through a logistic function $g(x)$ to avoid out-of-range prediction, such as negative predictions. [3] We define the conditional distributions for R, U, V as

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_iV_j^\top,\sigma^2)\right]^{I_{ij}}, \tag{2}$$

$$p(U|\sigma_U^2) = \prod_{i=1}^{N}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}), \tag{3}$$

$$p(V|\sigma_V^2) = \prod_{i=1}^{N}\mathcal{N}(V_i|0,\sigma_V^2\mathbf{I}), \tag{4}$$

Summing up the log of posterior probabilities, the loss function is given by

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - g(U_iV_j^\top))^2 + \frac{\lambda}{2}\left(\sum_{i=1}^{N}\|U_i\|_{Fro}^2 + \sum_{j=1}^{M}\|V_j\|_{Fro}^2\right) \tag{5}$$

where λ is the regularization factor, and I_{ij} is the indicator function that is equal to 1 if user i has purchase record for product j and 0 otherwise.

3.2.2 Optimization

In our experiment, we use Stochastic Gradient Descent with mini-batch to train our model. The detailed procedure is shown in *Algorithm 1* (next page). [1]

3.2.3 Evaluation

In the tuning process for PMF model, we performed a grid search using different sets of parameters. By selecting the set of parameters which can result the lowest loss, we choose a model with $K = 40, \eta = 0.1, \lambda = 0.001$. However, when evaluating the PMF model using our Mean Average Precision (MAP) function, we found the accuracy is extremely poor indicating something wrong with the model, which we were unable to identify.

Algorithm 1: Stochastic Gradient Descent algorithm for PMF

```
    Input: $R$: user-product-score matrix
           $K$: No. of latent factors
           $N$: No. of users
           $M$: No. of products
           $\eta$: Learning rate
           $\lambda$: Regularization parameter
           $s$: No. of epoch
    Output: $U, V$
 1: Procedure SGD($K, N, M, \eta, \lambda, s$):
 2:     Initialize $U, V$ with $\mathcal{N}(\mu, \Sigma)$
 3:     for $step \in \{1, \dots, s\}$ do
 4:         for mini-batch in $R$ do
 5:             for $i, j$ in mini-batch do
 6:                 $U_i \leftarrow U_i - \eta\frac{\partial E}{\partial U_i} = U_i + \eta((R_{ij} - g(U_iV_j^\top))\frac{\partial g(U_iV_j^\top)}{\partial U_iV_j^\top}V_j - \lambda U_i)$
 7:                 $V_j \leftarrow V_j - \eta\frac{\partial E}{\partial V_j} = V_j + \eta((R_{ij} - g(U_iV_j^\top))\frac{\partial g(U_iV_j^\top)}{\partial U_iV_j^\top}U_i - \lambda V_j)$
 8:             Update loss
 9:         Update loss list until convergence or end of iteration
10:     return $U, V$
```

3.3 Non-negative Matrix Factorization

Due to the unknown issue with the PMF model, we explored another matrix factorization method: Non-negative Matrix Factorization (NMF).

3.3.1 Model Representation

Similar to PMF, NMF decomposes the original normalized matrix R with M products and N users into two matrices $U \in \mathbb{R}^{N\times K}$ and $V \in \mathbb{R}^{M\times K}$. The loss function to be minimized is

$$E = \frac{1}{2}\|R - UV^\top\|_{Fro}^2 + \alpha \cdot l1\_ratio \cdot \|\mathbf{U}\|_1 + \alpha \cdot l1\_ratio \cdot \|\mathbf{V}\|_1 + \frac{1}{2}\alpha \cdot (1 - l1\_ratio) \cdot \|U\|_{Fro}^2 + \frac{1}{2}\alpha \cdot (1 - l1\_ratio) \cdot \|V\|_{Fro}^2 \quad (6)$$

where $\|\cdot\|_{Fro}$ is Frobenius norm and $\|\cdot\|_1$ is Element wise L1 norm.

Upon decomposing the normalized matrix R, for a given user u_p, we compute the product scores for all products with $u_p \cdot V^\top$ and choose the top 10 products for recommendation.

3.3.2 Evaluation

Due to the large dataset size and limited computational resources available, we conducted grid search with a narrow range of values to tune the three hyparameters.

We used the MAP of the first 10,000 users from the validation set to select the best values.

Hyperparameter Tuning			
Parameter	Meaning	Possible Values	Best Value
n_components	Number of latent factors	[10, 120] with a step of 10	120
alpha	Regularization parameter for both W and H	[0.001, 0.01, 0.1]	0.001
l1_ratio	Ratio of weights assigned to l1 regularization	[0, 0.2, 0.5]	0

The MAP of the NMF model with the best sets of parameters on the test set is 50.57%, meaning that the user will purchase every second product recommended on average.

3.4 NMF with Co-Clustering

One possible issue we identify with NMF model is that we treat each item as a class, and therefore we have nearly 50,000 classes. For a particular user, most of the products are not relevant and thus those irrelevant products may create noise and affect our model performance.

To narrow down the possible products for recommendation, we performed information-theoretic co-clustering with the *CoClust* package before NMF. [2]

3.4.1 Model Representation

The information-theoretic co-clustering uses mutual information between two categorical variables to create clusters with the given row and column cluster numbers.

Our matrix R is a contingency table and let $p_{u,p}$ be the joint probability distribution associated with users and products. The mutual information compares the observed frequencies under the contingency table with the expected frequencies under the null hypothesis of no association. As such, the mutual information can be expressed as $\mathcal{I}(P_{U,P}) = \sum_{u,p} p_{u,p} \log \frac{p_{u,p}}{p_u p_p}$. A measure $delta\_kl = \frac{p_{u,p}}{p_u p_p}$ measures the importance of a cluster to a row or column of clusters. [2]

3.4.2 Evaluation

The number of user and product clusters are chosen to be 40 and 10 respectively, following the ratio of 4:1 for the number of users to products.

Given a user id, we identify the corresponding user cluster, and the best product cluster based on the *delta_kl* values. We then performed NMF with the best sets of parameters on the submatrix of the user product cluster and identify the top 10 products for recommendation.

However, the MAP on the test set is only 37.51%, which is significantly lower than the NMF model. It means that the user will purchase every third product recommended on average. The possible reason for the worse performance is the non-optimal cluster number used, which may lead to too few products to recommend from a particular user product cluster.

3.5 Model 2 Limitations and Future Directions

3.5.1 About Models

While we expected PMF model to perform the best, we could not proceed with it due to the unknown problem. For NMF, further hyperparameter tuning may be conducted with greater computational power. Lastly, hyperparameter tuning should be conducted on the number of row and column clusters for co-clustering. Moreover, the matrix used for co-clustering currently is the normalized matrix. The original matrix may have a better performance.

3.5.2 About Collaborative Filtering

In general, Collaborative Filtering models suffer a problem that it is difficult to recommend products for users who has very few purchase records. One possible solution can be removal of such users in model training until they have sufficient large number of records.

When the model is deployed, PMF model will have a considerate advantage as it can scale linearly with the number of observations, while NMF have difficulty to handle very large matrix.

4. Overall Evaluation and Conclusion

4.1 Overall Evaluation

The overall recommendation is a combined result from the two models:

1. Model 1: XGBoost with $learningrate = 0.1$, $max\_depth = 10$, $n\_estimator = 500$
2. Model 2: NMF with $n\_comonents = 120$, $alpha = 0.001$, $l1\_ratio = 0$

The recommendation consists of top 5 products from Model 1 (reordered) and top 3 new products from Model 2. The test performance is 47.32%, indicating that with our models, the user will purchase every second product recommended on average.

4.2 Conclusion

In order to offer the most relevant and personalized product recommendations, we had two approaches targeting previously purchased products and new products separately, using various machine learning techniques. The overall performance is promising.

We believe the above (but not limited to) future research directions will advance the technology presented in this thesis and contribute to academia and industry.

Bibliography

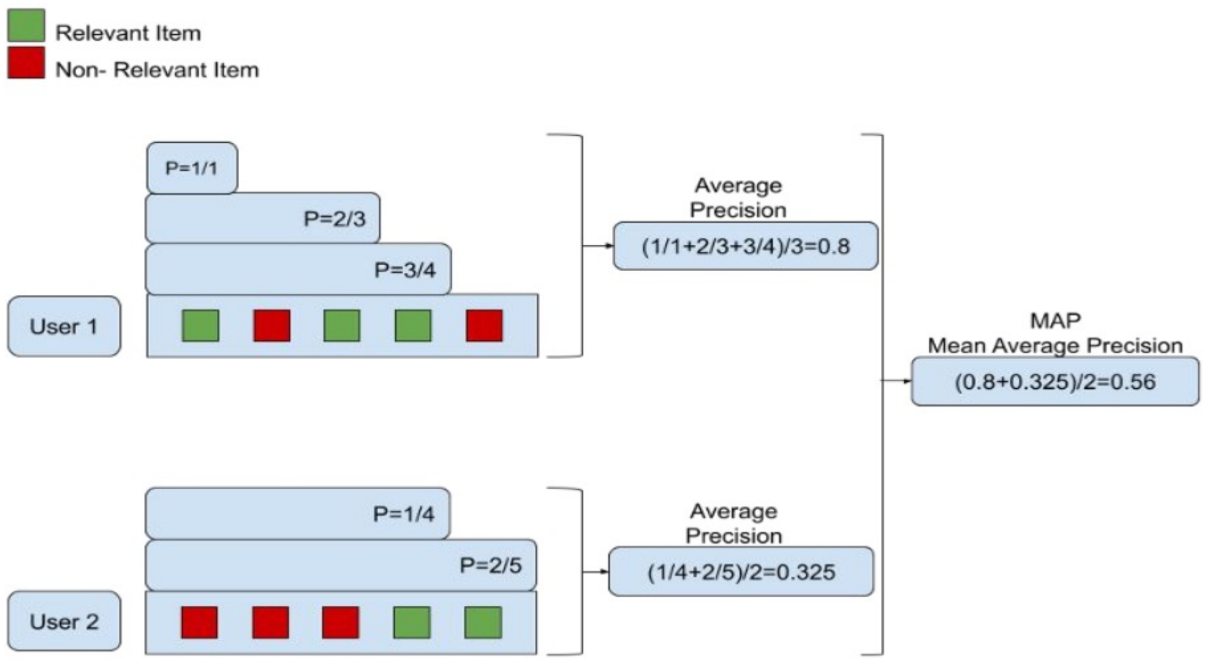
[1] R. Francesco, “Part 13: Item-to-item collaborative filtering and matrix factorization”, University of California, Irvine, Tech. Rep.

[2] R. François, M. Stanislas, and N. Mohamed, “Coclust: A python package for co-clustering”, https://hal.archives-ouvertes.fr/hal-01804528/document.

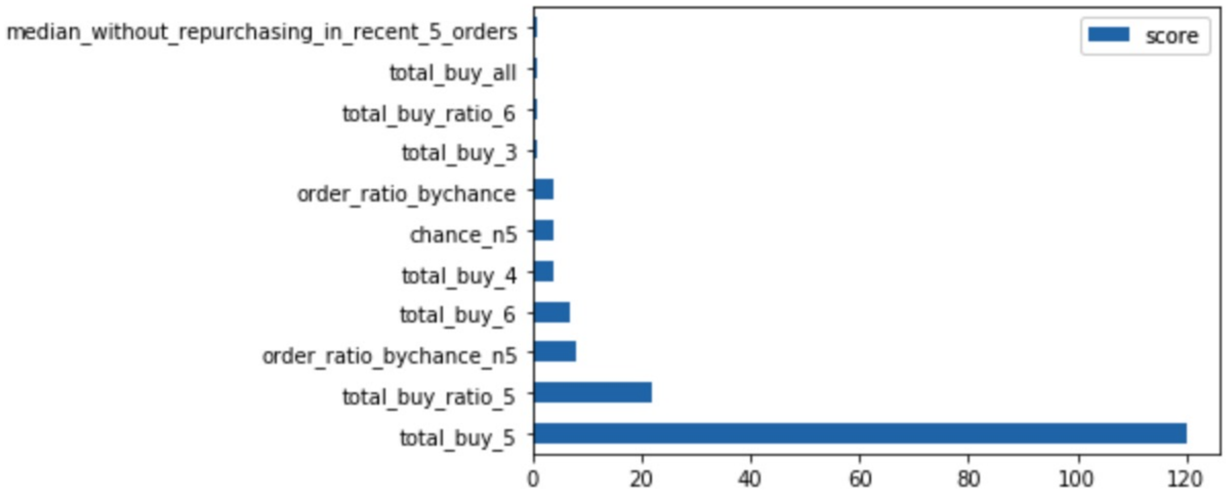
[3] S. Ruslan and M. Andriy, “Probabilistic matrix factorization”, 2007.

Appendix

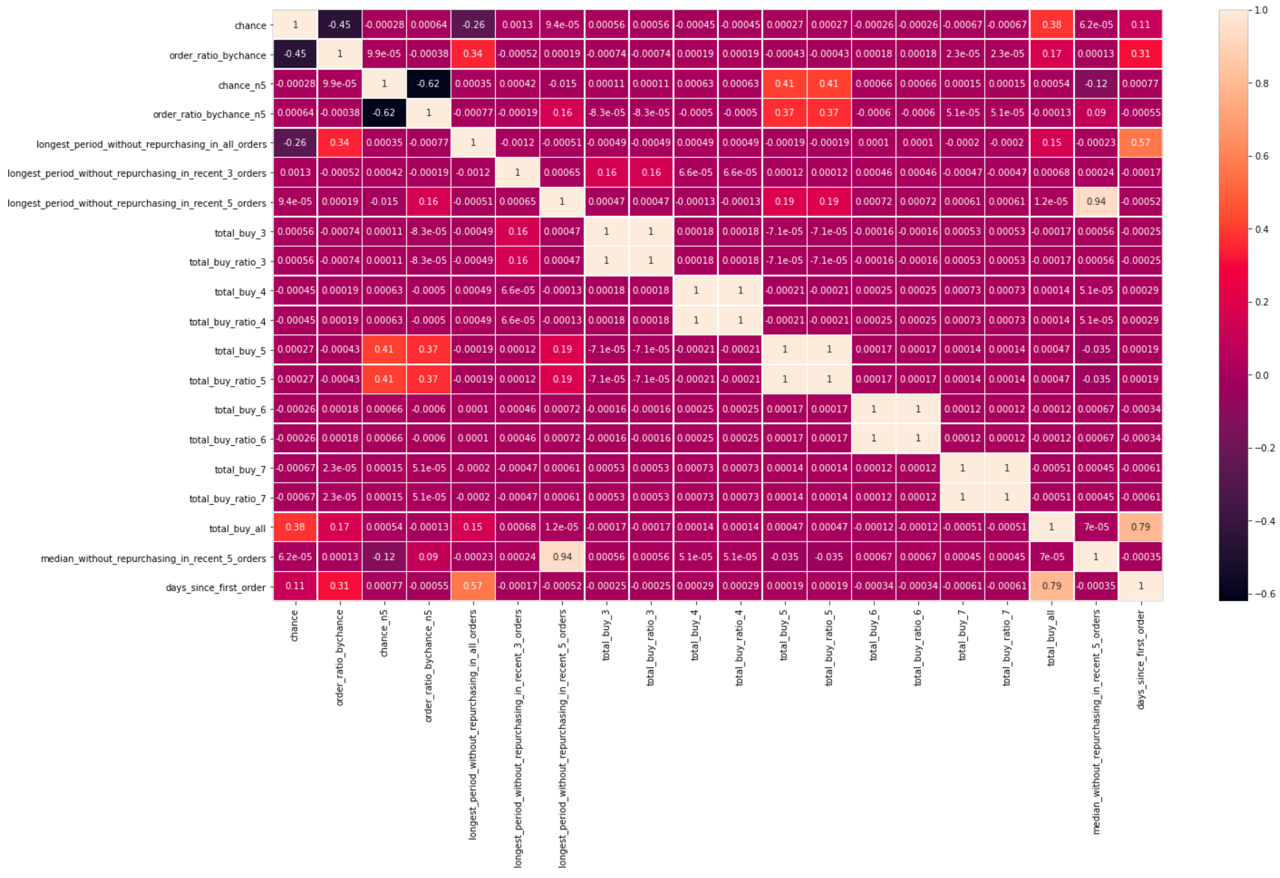
1. Evaluation Matrix - Mean Average Precision



2. Feature Importance for Model 1



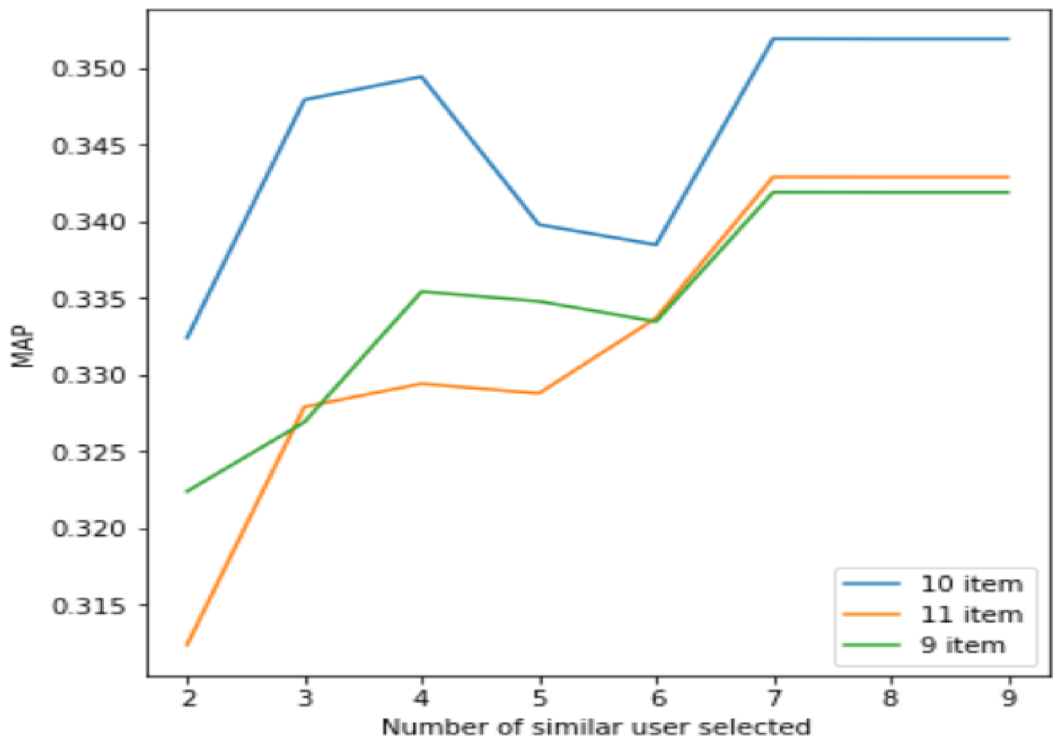
3. Feature Correlation for Model 1



4. Full Model Specifics

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.8, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=10,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=500, n_jobs=4, nthread=4, num_parallel_tree=1,
              objective='binary:logistic', random_state=27, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, seed=27, subsample=0.8,
              tree_method='approx', validate_parameters=1, verbosity=None)
```

5. TF-IDF Model Performance


MAP
Number of similar user selected
10 item
11 item
9 item