**Recommender System Based on Purchasing History**

Shengshi Yuan (ssy293@nyu.edu)

Tiantian Zhang (tz921@nyu.edu)

Xinyi Tan (xt2038@nyu.edu)

December 5, 2020

Introduction to Data Science Course

https://github.com/MinnieTan/Recommender-System-Based-on-Purchasing-Behavior-Data

**1 Business Understanding**

In the era of information explosion, consumers are surrounded with almost countless choices. Understanding consumers preferences and making personalized recommendations are crucial keys to enhance user experience. Accurate recommendations would raise customer loyalty to the platforms and boost sales. As estimated by McKinsey in 2017, 35% of the consumer purchases on Amazon comes from personalized product recommendations [7]. Amazon had $37.9 billion revenue in the second quarter of 2017, so it has approximately $13.265 billion sales generated through product recommendation in that single quarter.

Given the original datasets from a medium online cosmetics shop, we plan to build two recommender systems: one recommends products for customers while the other recommends the brands. With such recommender systems, the online store can choose to recommend products or brands to a particular customer. For example, the retail platform can send emails recommending particular products or offering special coupon codes on particular brands.

Collaborative filtering is one of the most popular models for recommender systems. It predicts the preference of a user via taste information from other users, assuming that a user would prefer similar items and an item would attract users with similar interests. Primary approaches of collaborative filtering are the *memory-based* *(neighboring-based)* method, *model-based* method, *Matrix Factorization* method, and *deep-learning* method.

The memory-based approach utilizes user-item data to calculate similarities between user or item pairs. This technique consists of two approaches: user-based and item-based. The user-based approach finds similar users who interact with the same group of items in the same way (e.g. purchase, rate 5 stars). For example, if Faye purchased brand A’s shampoo, the system would find users who also purchased brand A’s shampoo and what other products those users have purchased in order to recommend Faye products. On the other hand, the item-based method would find Faye’s preference for an item based on Faye’s purchasing history of similar items.

Matrix Factorization is one of the state-of-the-art solutions for collaborative filtering. It uses latent factors to decompose the user-item interaction data [3]. Prediction of the user-item interaction can be realized through machine learning algorithms such as Stochastic Gradient Descent (SGD), Alternating Least Square (ALS), and Implicit ALS. Implicit ALS can address the challenge of implicit user behavioral data (purchase counts, etc.) where a low numerical value on preference between a user-item pair doesn’t necessarily indicate dislikes [2].

**2 Data Understanding**

**2.1 Data Source**

The original datasets, which contain 5 months (Oct 2019 - Feb 2020) behavior data among 1639358 customers and 53904 unique products from a medium online cosmetics store, were obtained from Kaggle [5]. (See Table 1. in Appendix)Each data instance represents an event (view, add to cart, remove from cart, or purchase) between a user and a product during a visiting session to the website. 98.25% of the product category textual information and 41.95% of the product brand information is missing. Other than the pricing information, we are only left with categorical ids, which are not easy to make sense of without corresponding textual explanation.

**2.2 Concept Drift**

Online shopping events greatly influence people’s purchasing behaviors. As in Fig.1*.* (in Appendix), the sales volume on Black Friday doubled the normal sales amount. Those extra purchases from sales events streamed into the system and we have no control over them. Thus, we believe that some degrees of concept drift exist within our data. Moreover, it is possible that the shopping platform has already utilized recommender systems on the website. The nature of data might have already been intervened, resulting in a negative feedback loop within the dataset.

**3 Data Preparation**

**3.1 Data Cleaning**

The five monthly datasets (Oct 2019 - Feb 2020) were merged together and duplicated rows were removed as they might have been collected due to some technical errors. Products with non-positive prices are filtered out as these products are probably some complimentary gifts, which are some temporary products that we can’t recommend to customers. Only users who brought at least 10 different products are kept to construct our user-product matrix. The rows of the matrix represent unique users and columns represent products. The (i,j)-entry represents the number of times that user i purchased product j.

When constructing the user-brand matrix, we filtered out rows with missing brand information. Since 41.95% of the product brand information is missing in the original datasets, we didn’t treat them all as a single“no name” brand as this “no name” brand would be the dominant brand with the most popularity. The user-brand matrix was constructed with users who have purchased at least 5 different brands. The rows of this matrix represent unique users and columns represent unique brands. The (i,j)-entry represents the number of times that user i purchased brand j.

We filtered the data to include only users with enough purchase records so that our recommender systems will have enough signals to learn from.

**3.2 Train, Validation, and Test split**

For a given user in our matrices, 20% of his or her purchase history is randomly selected to be in the test set, 20% of the purchase history in the validation set, and the remaining 60% of the purchase history in the training set (see Table 2. in Appendix). Therefore, the training, validation, and testing matrices have the same size and the entries of one matrix are zeroed out in the other two matrices.

**3.3 Target Variable**

To build the recommender system, we would like to know people’s preferences towards the products and brands. Since the datasets don’t have any explicit feedback such as users’ ratings, we decided to rely on the variable `event\_type` which conveys implicit feedback through users’ actions of view, add to cart, purchase and remove from cart. We consider `purchase` as the signal of preference towards certain products and brands. Therefore, we defined the target variable as the number of purchases for both products and brands (see Fig. 2. in Appendix for distributions).

**3.4 Feature Engineering**

To generate a numerical quantification of users’ preferences on certain brands or products from the raw browsing history of customers, we firstly combined all browsing data from October 2019 to February 2020 together and then grouped the whole dataframe by unique *user\_id*s.

For user-brand relationships data, we summed the purchase counts for each user-brand pair that had at least one ‘purchase’ event associated with them in the combined dataframe. Numerical *brand\_id* is also generated for corresponding *brand* information. See Table 3. In Appendix for a snippet of the sample data.

The user-product relationships data was generated in the same way as the above mentioned user-brand relationships data. See Table 4. In Appendix for a snippet of the sample data.

After generating user-brand and user-product relationships data, we then generate the user-brand and user-product matrix where each entry indicates how many times the user has purchased that brand or product from October 2019 to February 2020.

**4 Modeling & Evaluation**

**4.1 Evaluation Metrics**

The recommendation systems output a sorted list of items for every customer. Our goal is to have the sorted list accurately match customers’ preferences. Hence, Precision at K is an appropriate evaluation metric for model performances, where K is the number of products/brands to recommend for a specific user. For a given user:

To evaluate model performance, the metric is averaged over all customers. The number K is determined by business constraints. Since we do not know what would be an appropriate K for this particular online store, we calculated the mean Precision@K for a list of K values that we think are reasonable (K ∈ [1, 30]).

**4.2.1 Baseline Model: Popularity Model**

The popularity model takes the most popular products or brands for recommendation. It can help us detect the popularity bias that may exist in other models.Popularity model also resolves the cold-start problem present in the industry to some extent, especially in the case that only implicit purchasing history data is available.

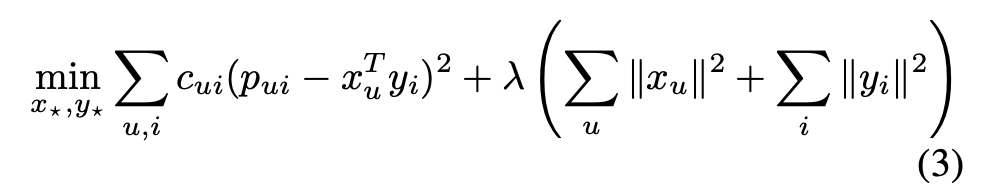
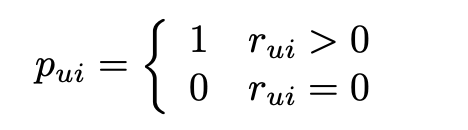
As indicated in Fig. 3 (in Appendix). Above, the baseline popularity model performs better for the User-Brand Matrix, which is not hard to understand as there are way more unique products than unique brands in the data.

**4.2.2 Model 1: Item-Based Collaborative Filtering**

Item-based collaborative filtering can be of great use for data containing way more users than items. The item-based approach computes a similarity matrix between items. For each customer, the predicted purchase counts for an item that he had not bought are taken as a weighted average of the purchase counts for S neighboring items. In this project, we fitted the model with 3 similarity measures(Cosine Similarity , Pearson Similarity, and Jaccard Similarity). After hyperparameter on S, we found that, for the User-Product matrix, Jaccard Similarity with S = 15 gives the smallest RMSE. For the User-Brand matrix, Jaccard Similarity with S = 25 gives the smallest RMSE on validation set. See Figure. 5. in Appendix for validation metric RMSE performance. See Figure. 4. in Appendix for test metric Precision@K performance.

**4.2.3 Model 2: Matrix Factorization with Implicit Alternative Least Squares**

Unlike explicit feedback such as rating, where a low rating indicates dislike and a high rating indicates like. It is hard to infer user preference from our target variable purchase counts. Also, a customer that did not purchase an item might have done so because he did not know about the item. The traditional Matrix Factorization method that ignores the zero entries in the matrix during optimization does not work well on our dataset. We followed the implicit Alternating Least Squares method proposed by Hu [12] to handle our implicit feedback dataset. The cost function is:

 where  and

In this setting, the variable is an indicator of our confidence in observing . For every use-item pair, as we observe more purchase counts, our confidence in P = 1 increases. The rate of increase is controlled by the hyperparameter . The After hyperparameter tuning, we found that for the User-Brand matrix, = 50, number of latent factors = 2, regularization = 1e-4, and number of max iterations = 40 gives the smallest RMSE on validation set. For the User-Product matrix, = 50, number of latent factors = 70, regularization = 1e-4, and number of max iterations = 30 gives the smallest RMSE on validation set. See Figure. 7. in Appendix for validation metric RMSE performance for User-Brand Matrix.See Figure. 8. in Appendix for validation metric RMSE performance for User-Product Matrix. See Figure. 6. in Appendix for test metric Precision@K performance.

**4.2.4 Model 3: Neural Network**

With deep learning frameworks from Keras, we built a collaborative filtering recommendation system with neural networks. This approach leverages the embedding layers to understand the interactions between the users and products or brands. Incorporating deep learning techniques help to tackle some drawbacks of traditional collaborative filtering. This approach also incorporates the ideas from matrix factorization by using inner product on the latent features of users and products.

**4.2.4.1 Structure of Neural Networks**

Our neural network consists of input layers for both users and items, embedding layers and reshape layers for users and items, and a dot layer combining the user embedding and item embedding via dot product. The embedding layers are important in recommender system implementation because they are used to map categorical objects such as user IDs with similarities. See Figure. 9. in Appendix for Structure of Neural Networks Model.

**4.2.4.2 Tuning hyperparameters**

Within the embedding layers, there are many hyperparameters such as the number of epochs, learning rates, batch sizes, dimensions of embedding layers and number of latent factors that can be tuned for better performance. The graph below shows the learning curve of loss against the number of epochs. The optimal number of epochs is the elbow point of the learning curve. Besides tuning parameters, model performance can also be improved by changing optimizers and adding dropout layers. See Figure. 10. in Appendix for validation metric RMSE performance. See Figure. 11. in Appendix for test metric Precision@K performance.

**4.2.5 Other Efforts**

**4.2.5.1 Matrix Factorization with SGD/ALS Algorithms**

Matrix factorization with SGD or ALS algorithms are popular choices for implicit feedback data such as ratings for movies. However, ratings are quite different from the purchase counts target variable in this project in that ratings with lower values (e.g. 0, 1) indicates that the user doesn’t really enjoy the item whereas purchase counts equal to 1 doesn’t necessarily mean that the user dislikes the item [11]. In fact, any nonzero purchase counts suggest some extent of satisfaction towards the item. In this sense, the difference between 0 and 1 is much larger than the difference between 1 and 2 for purchase count values. Therefore, matrix factorization with SGD or ALS algorithms doesn’t really work with our data. We decided to test the performance of matrix factorization with an implicit ALS algorithm instead.

**4.2.5.2 Random Forest Regressor**

In matrix factorization, vectors containing latent features for each user and item are learned. Rearranging the user vector, item vector, and observed preference between them, we can convert this to a regression problem. We trained a Random Forest Regressor on the user matrix and product matrix learned via matrix factorization with iALS algorithm. In this case, prediction for user preference on certain items can be achieved by combining corresponding user vector and item vector as the input. However, the validation metrics and test metrics for the random forest regressor didn’t look as ideal as compared to other choices of model. Considering that the prediction schema for random forest schema is not very different from matrix factorization with implicit ALS algorithm, we decided not to move forward with random forest regressor.

**4.3 Model Selection**

|  |  |  |
| --- | --- | --- |
| Model | Optimal Validation RMSE | Optimal Test Precision@K |
| Baseline Popularity Model | NA | 0.15 |
| Item-Based Collaborative Filtering  (S = 15) | 4.44 | 0.37 |
| Matrix Factorization with ALS Algorithm  (Number of Features = 20  Max iterations = 40  iALS Confidence Scaling Factor = 25  Log Scale Regularization = -4) | 3.995 | 0.3 |
| Neural Networks  (Epoch = 20) | 1.3 | 0.5 |

*Table 5. Optimal Model Performances for User-Brand Matrix*

|  |  |  |
| --- | --- | --- |
| Model | Optimal Validation RMSE | Optimal Test Precision@K |
| Baseline Popularity Model | NA | 2.8e-5 |
| Item-Based Collaborative Filtering  (S = 10) | 1.094 | 0.095 |
| Matrix Factorization with ALS Algorithm  (Number of Factors = 100  Max Iterations = 20  iALS Confidence Scaling Factor = 75  Log Scale Regularization = -4) | 0.55 | 0.065 |
| Neural Networks  (Epoch = 5) | 0.2 | 0.5 |

*Table 6. Optimal Model Performances for User-Product Matrix*

Based on the metrics performance presented above, we decided to select the neural network model using the user-product pairs as our final model. To improve the baseline model, we can perform clustering on the categories so that the popular items may be linked to customer groups.

**4.4 Model Results (performance on test data)**

With our selected model, we combined the training data with the validation data, and retained the model on the combined dataset. After predicting on the test set, we got a test precision@k, which is 0.10327541, which is lower than the precision@k on the training dataset.

As the neural network model has a relatively high precision@k, the product recommendations are more likely to satisfy customer needs. Also, deep learning techniques are quite flexible and generally perform better on the larger dataset, so as the firm keeps operating and attracting more new customers, the dataset will continue growing, and, therefore, the neural network model will become more sophisticated to make personalized recommendations. However, we need to be aware of the overfitting issues and control it through regularization.

**5 Deployment**

Our selected model attaches importance to the portion of our recommendations that are truty attractive to customers. In the real-world setting, not only are the products recommended to customers, but the overall customer experience also matters. When the retailing platform makes recommendations, it will be helpful to get immediate feedback through an explicit method such as a survey or an implicit method such as the click-through rate. Moreover, when presenting recommendations to customers, the interface is crucial as well. To improve customer shopping experience, we can conduct A/B Testing to get insights on people’s preferences towards interfaces. We can also recommend products and brands at various entry points such as the checkout page and add-to-cart page, comparing which entry got the best metrics.

In an actual production system, the performance of the model should be monitored and evaluated on a periodic basis. Before each model updates, we will use the new purchasing history as the test set to evaluate the performance of the previous model, and then update the model with those new history data. Considering that new products are continuously released each month, the model should be updated, say, every two months. The duration of model updates might need to be shortened to one month or even shorter when there are items from popular brands being released or an increased number of products on sale.

Depending on the actual application and context, the firm should be aware of the efficiency of the model operation and the computing resources needed. Sometimes, it would be worthwhile to sacrifice some accuracy or precision in order to reduce the costs. The firm should also be mindful of the fact that real-world model performance might deviate from expectations. The deep learning method usually requires large computational power and more memory, so the firm should make tradeoffs between the model performance and the explicit and implicit costs.

Recommender systems can have a significant effect on users, or even the society, as they guide customers’ choices via repeatedly sending recommendations to users [4]. Should the website just recommend products and brands strictly following the recommender system output? Or should the platform also recommend products or brands with great ideas (cruelty-free cosmetics products, etc.) Proprietary and privacy issues have also been the focus of debates in the field of recommender system applications. Generating personalized recommendations for users indicates that some information of the users are retrieved by the recommender system. The customer might feel vigilant if the recommended products match his or her needs and preferences too well.

One limitation of our model is that the model can only recommend products and brands to the registered customers. The new users may not get good recommendations because our model only considers the `purchase` event and ignores `view` and `add-to-cart`. Another limitation is that deep learning methods might perform poorer on small dataset compared with large dataset. Retailing platforms with limited users will have to find more efficient alternatives. Besides, we need to decide whether the model should keep the purchasing history long time ago. If a mother bought some baby products such as strollers five years ago, she might not need those now because her child has grown up already.

**6 Conclusion and Future Work**

The Recommendation system has brought huge impacts to the e-commerce industry, and it even influences the business model of those online retail service providers. In this project, we constructed the recommender systems mainly using techniques that belong to collaborative filtering and matrix factorization. The neural network using the user-product pair performed the best among the algorithms for both user-brand recommendation and user-product recommendation.

For future work, recommender systems builders may firstly group the customers according to explicit or implicit user information and then fine tune models for different groups of customers. RFM (Recency, Frequency, Monetary) analysis is a popular customer analysis technique for user segmentation [1]. Even with only implicit purchasing behaviors data, recommender systems builders can analyze the recency of a customer’s purchasing behavior, the frequency of a customer’s purchasing history, and the monetary value of a customer’s spending. See Figure. 12 in Appendix for RFM Analysis of Customers.

Fig. 12. indicates that the online cosmetics shop datasets has around 300,000 new users every month, the majority of customers purchased less than 2,500 times, and most consumers spent no more than 25,000 dollars. Note that the range of frequency and monetary for customers is very wide. There are customers who made more than 7,500 purchases and spent more than 75,000 dollars. Intuitively, recommender systems builders should expect those “VIP” users to have different preferences from the majority of customers who only purchased a few products from the online cosmetics shop.

Furthermore, recommender systems builders can combine explicit user information (if available) with the RFM analysis results to cluster customers into appropriate number of groups for more elaborate model training and evaluation.

**7 About Kaggle**

Although we retrieved the original datasets from Kaggle, they’re not associated with any competitions. So, unfortunately there is no leaderboard for us to compare our model with those of others.

**8 References**

[1] Daqing Chen, Sai Laing Sain, and Kun Guo, “Data Mining for the Online Retail Industry: A Case Study of RFM Model-Based Customer Segmentation Using Data Mining,” (August 27, 2012) Database Marketing & Customer Strategy Management 19, 197-208

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky, “Collaborative Filtering for Implicit Feedback Datasets,” 2008 Eighth IEEE International Conference on Data Mining, 2008. https://doi.org/10.1109/icdm.2008.22.

[3] Yehuda Koren, “Matrix Factorization Techniques for Recommender Systems,” Published by the IEEE Computer Society, IEEE 0018-9162/09, pp. 42- 49, ©IEEE, August 2009.

[4] Silvia Milano, Mariarosaria Taddeo, and Luciano Floridi, “Recommender Systems and Their Ethical Challenges,” AI & Soc 35, 957–967 (2020). <https://doi.org/10.1007/s00146-020-00950-y>

[5] Michael Kechinov, “eCommerce Events History in Cosmetics Shop,” (March 2020) distributed by REES46 Marketing Platform, <https://www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop>

[6] Dokyun Lee and Kartik Hosanagar. “Impact of Recommender Systems on Sales Volume and Diversity.” ICIS (2014).

[7] Faggella Daniel, (2018, August 14). The ROI of recommendation engines for marketing. Retrieved December 04, 2020, from https://martechtoday.com/roi-recommendation-engines-marketing-205787

**Contributions**

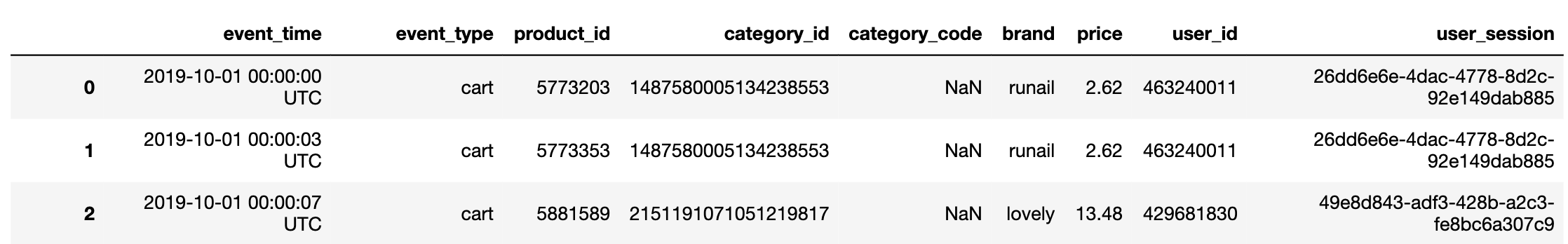
All members contributed to the problem formulation, data collection, model evaluation, and write-up of the final report.

Tiantian Zhang worked on data visualization, model-based collaborative filtering model building, matrix factorization with alternative least squares algorithm model building, and neural networks model building.

Shengshi Yuan worked on data preparation, baseline popularity model building, item-based collaborative filtering model building, matrix factorization with stochastic gradient descent algorithm model building, and matrix factorization with implicit alternative least squares algorithm model building.

Xinyi Tan worked on data visualization, data preparation, baseline popularity model building, matrix factorization with stochastic gradient descent algorithm model building, matrix factorization with alternative least squares algorithm model building, random forest regressor model building.

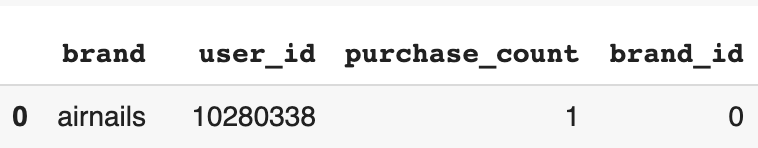
**Appendix**



*Table 1. Example Data Instances of the Original Datasets.*

|  |  |  |
| --- | --- | --- |
|  | User-Brand Matrix | User-Product Matrix |
| Shape | (22185, 248 ) | (37284,38053 ) |
| Number of non-zero entries in training set | 101335 | 554515 |
| Number of non-zero entries in validation set | 33779 | 184834 |
| Number of non-zero entries in testing set | 33778 | 184830 |
| Sparsity | 96.93% | 99.93% |

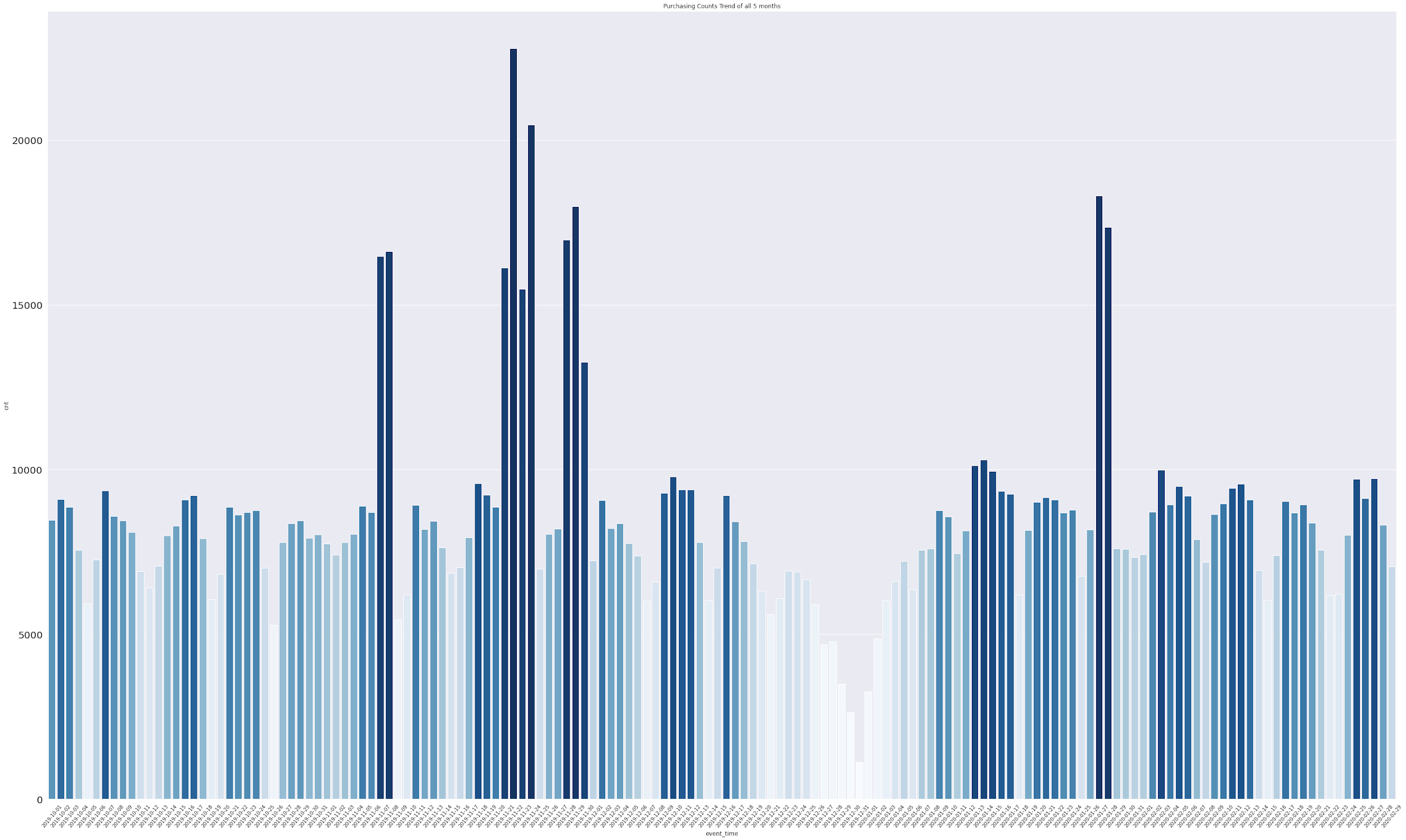
*Table 2. User-Brand Matrix and User-Product Matrix Information*

**

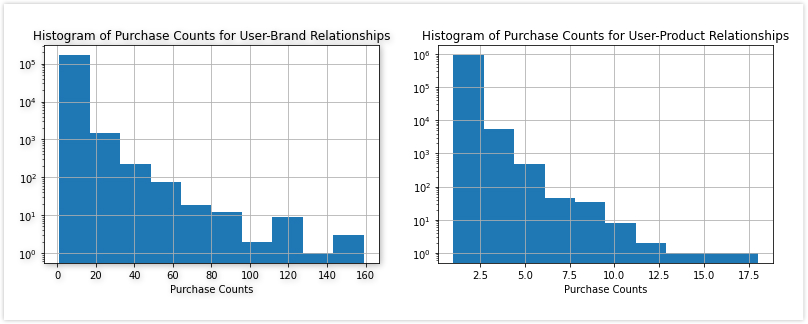
*Table 3. Sample Data Instance for User-Brand Relationships Data*

**

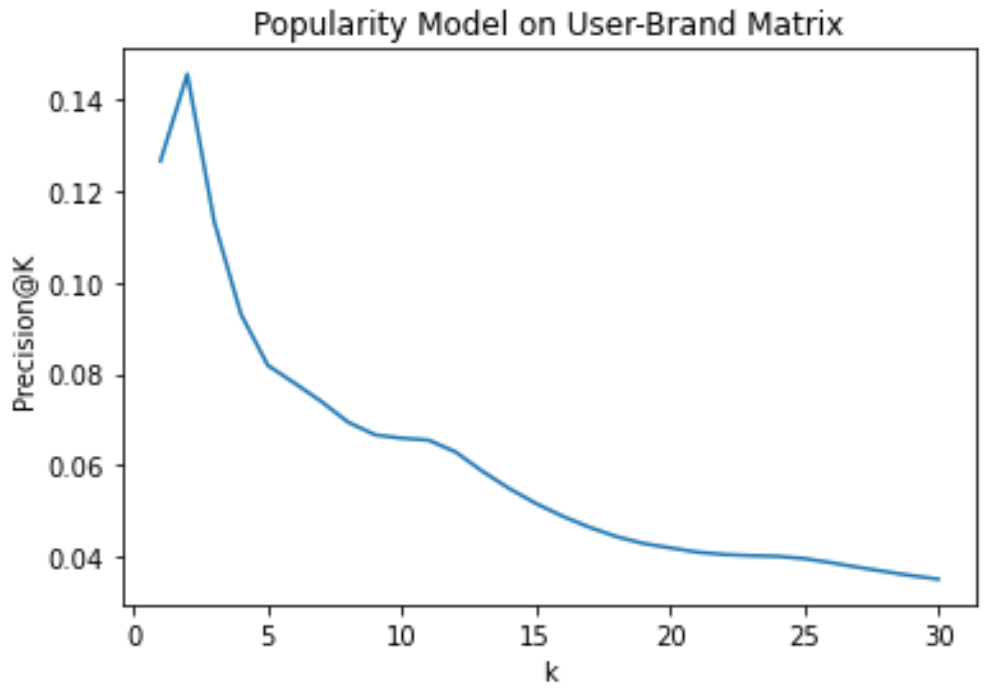
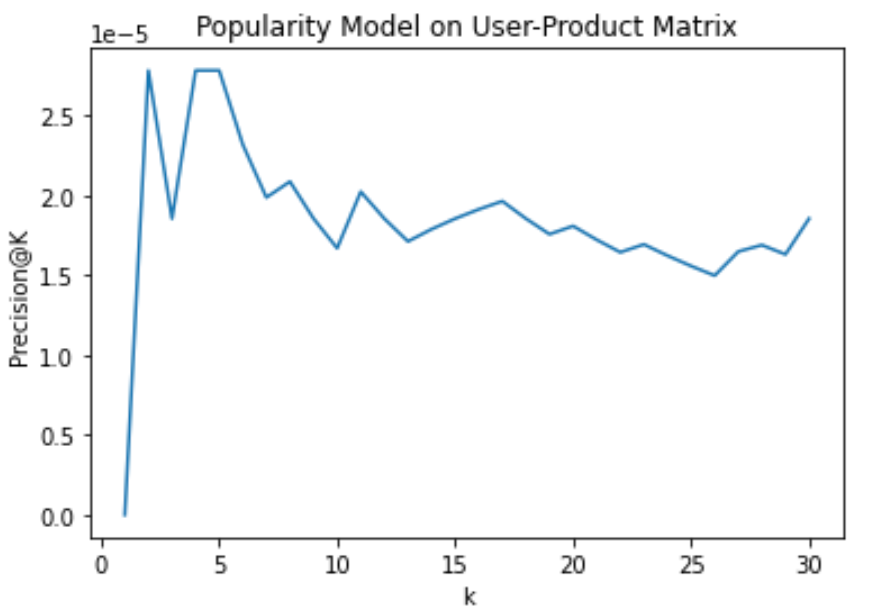
*Table 4. Sample Data Instance for User-Product Relationships Data*



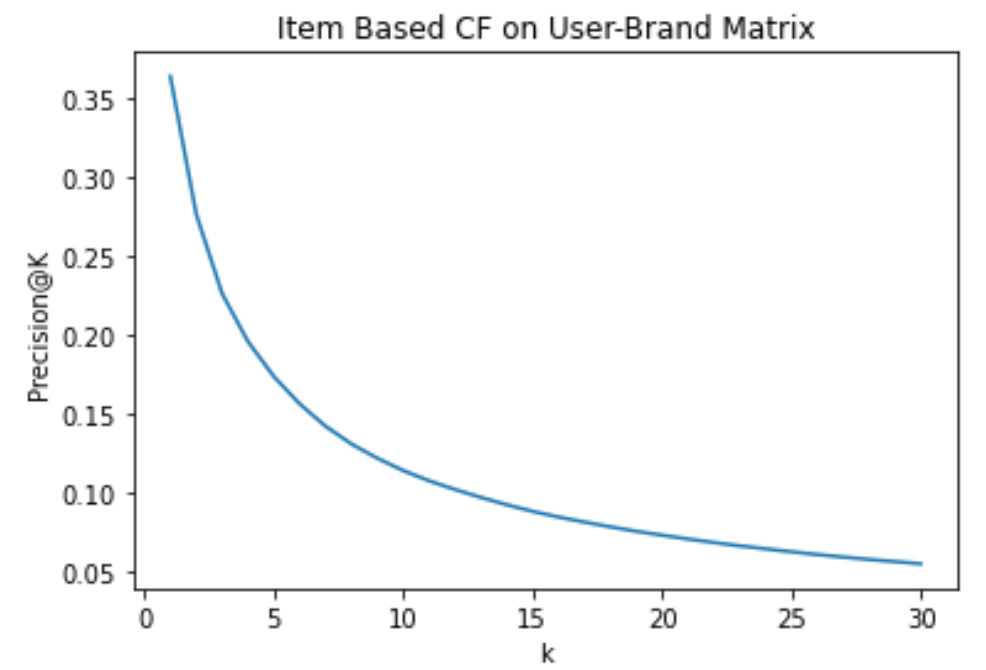
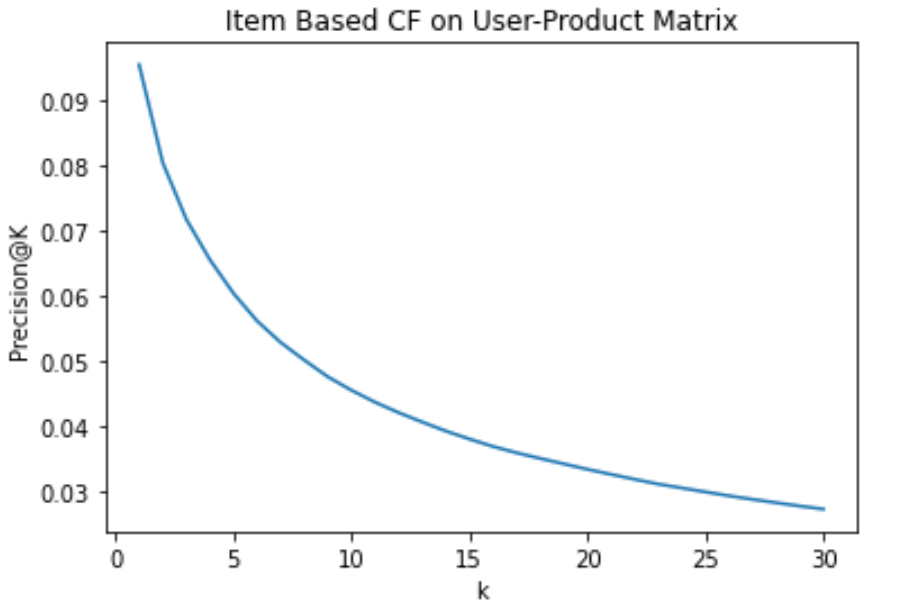
*Fig. 1. Number of Products Purchased from 2019-10-01 to 2020-02-29*



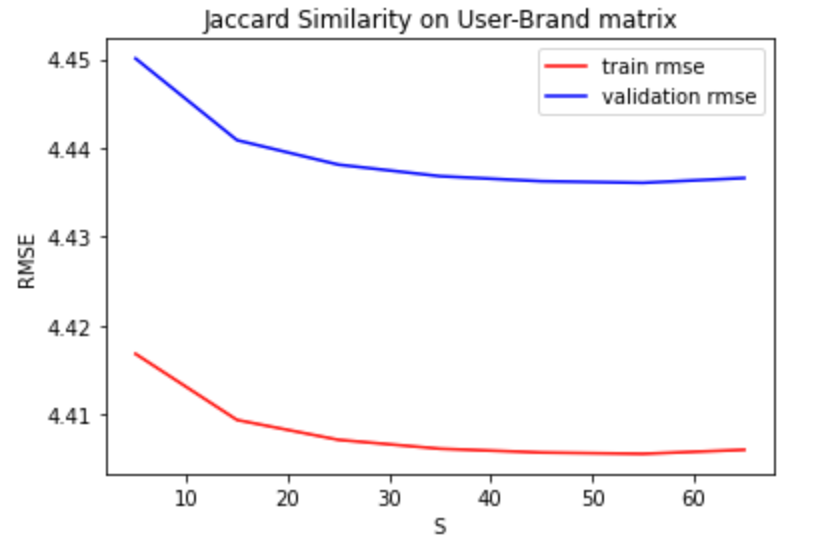
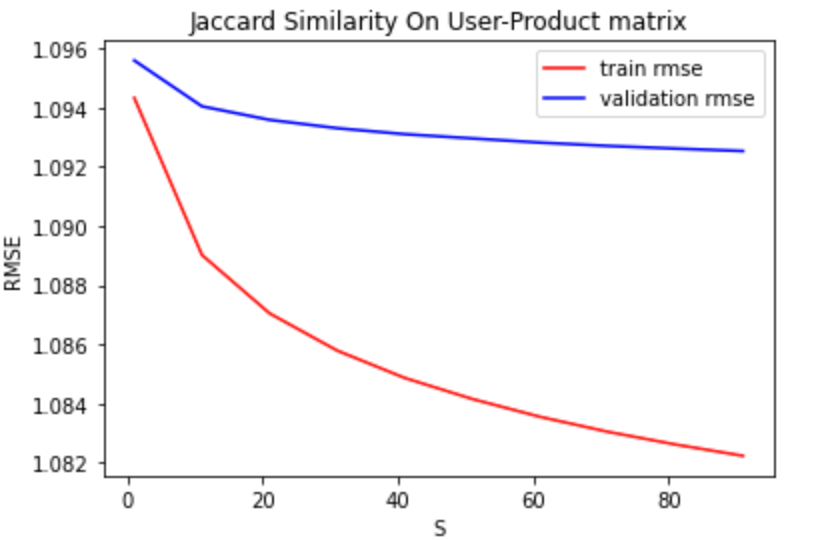
*Fig. 2. Distributions of Target Variable ‘purchase\_count’ for User-Brand Relationships Data (Left) & User-Product Relationships Data (Right)*



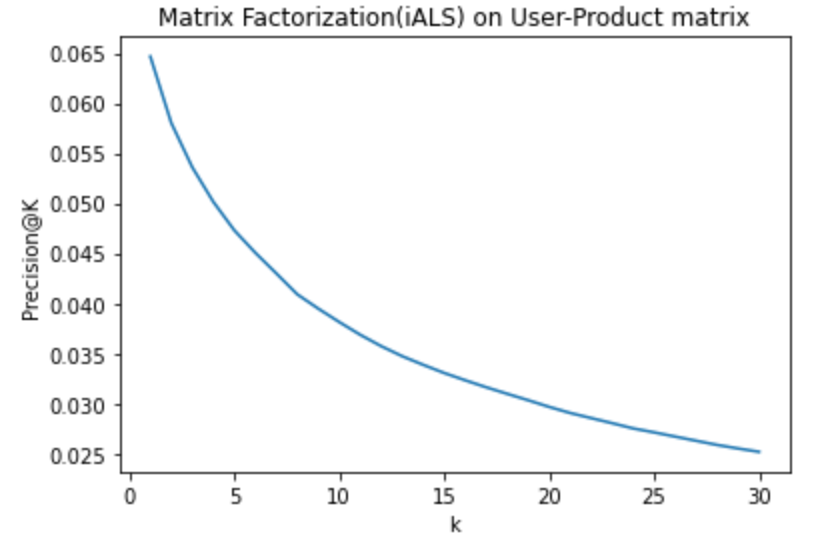
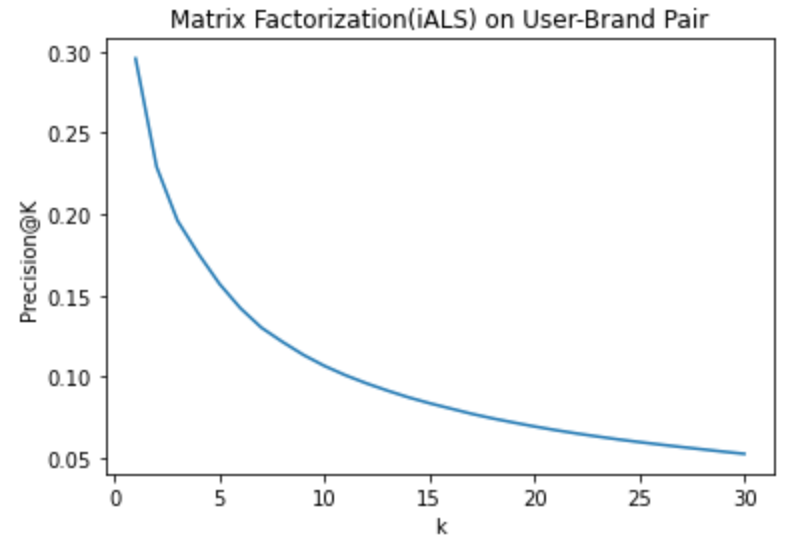
*Fig. 3. Precision@K for User-Product Baseline Model (Left) & User-Brand Baseline Model (Right)*



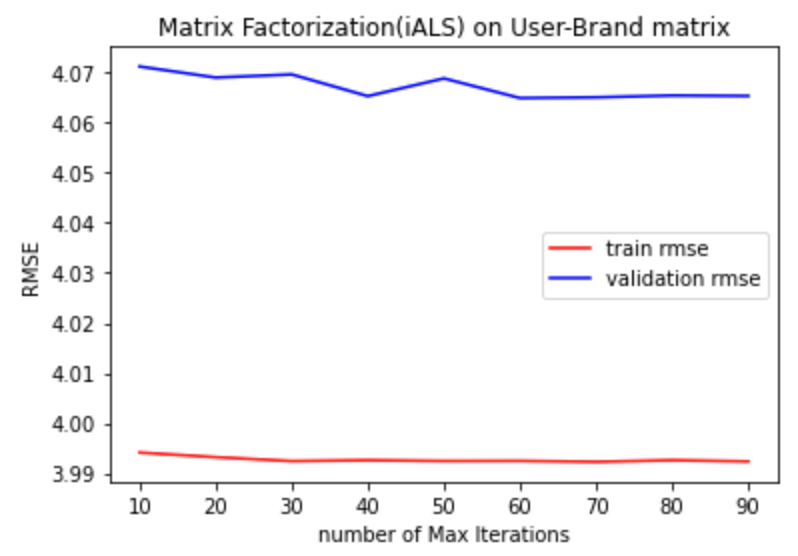
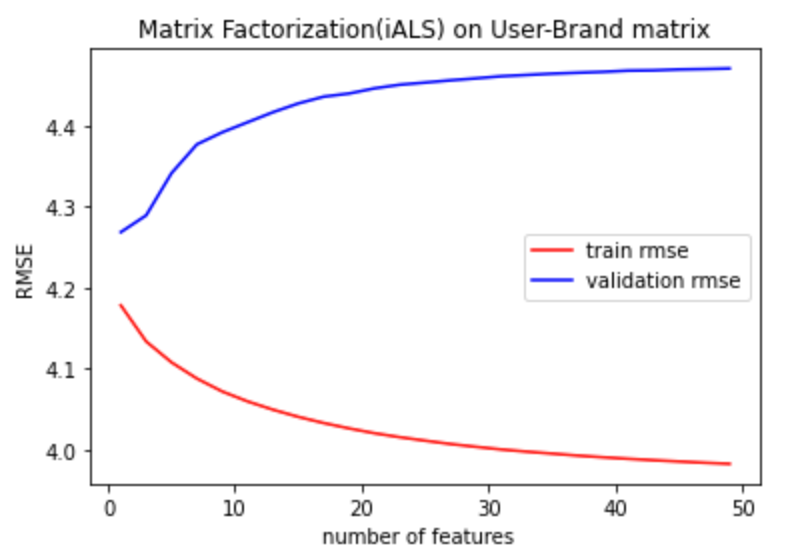
*Fig. 4. Test Precision@K for Item-Based Collaborative Filtering on User-Product Matrix (Left) & User-Brand Matrix (Right)*

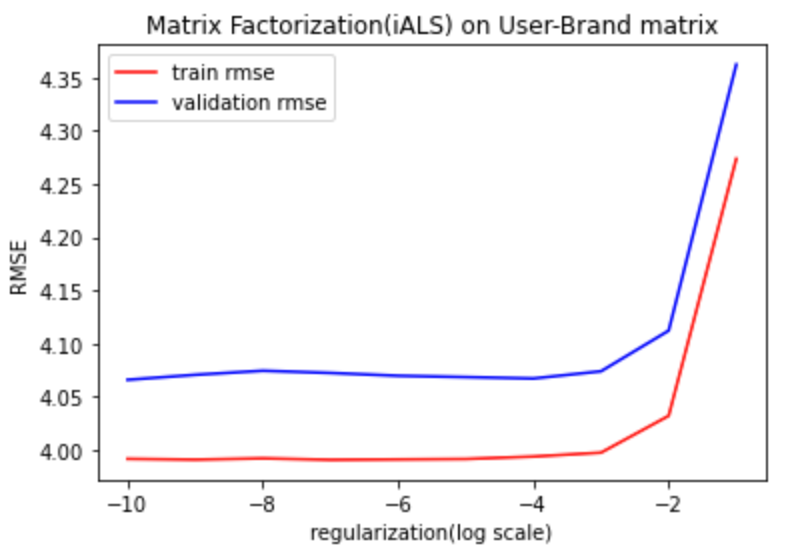
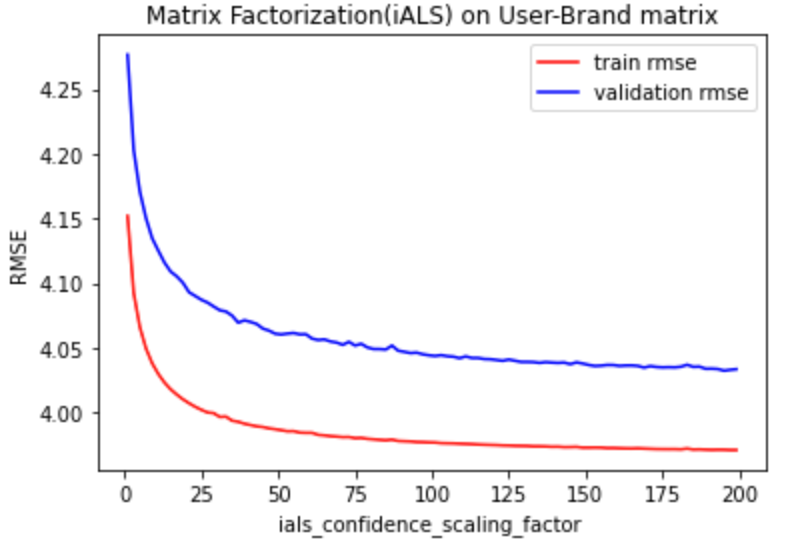


*Fig. 5. Validation RMSE v.s. S on User-Product Matrix (Left) & User-Brand Matrix (Right)*

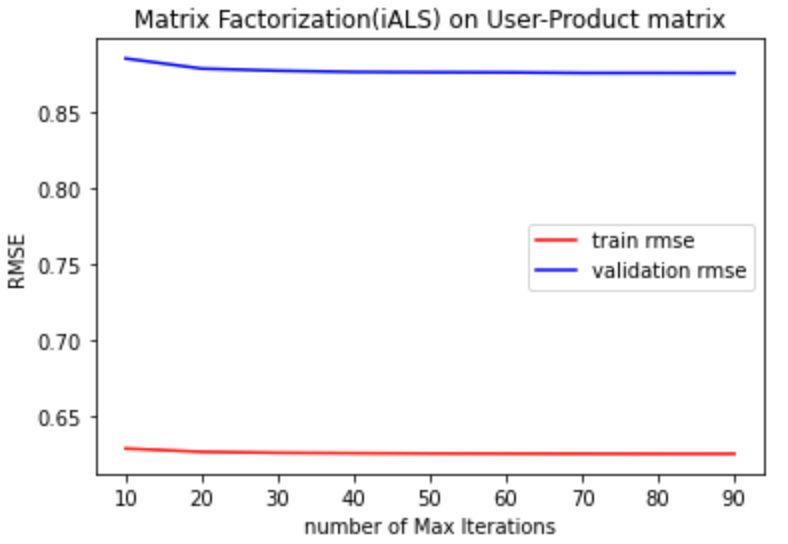
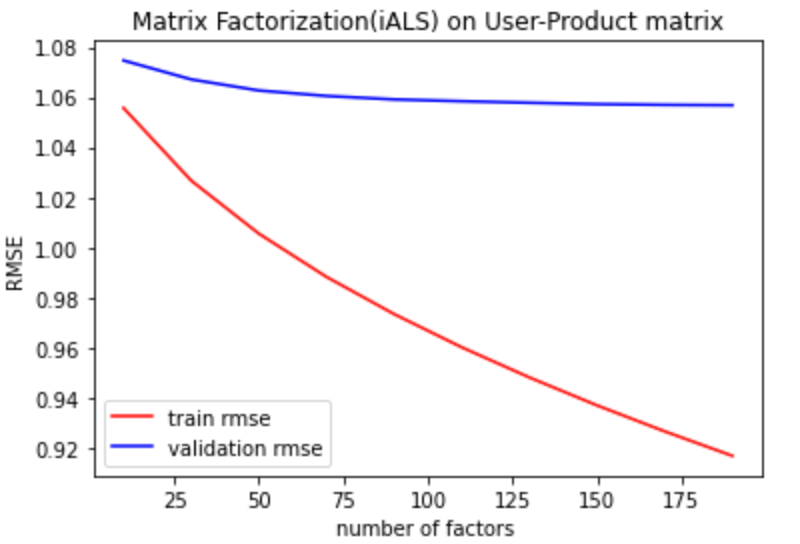


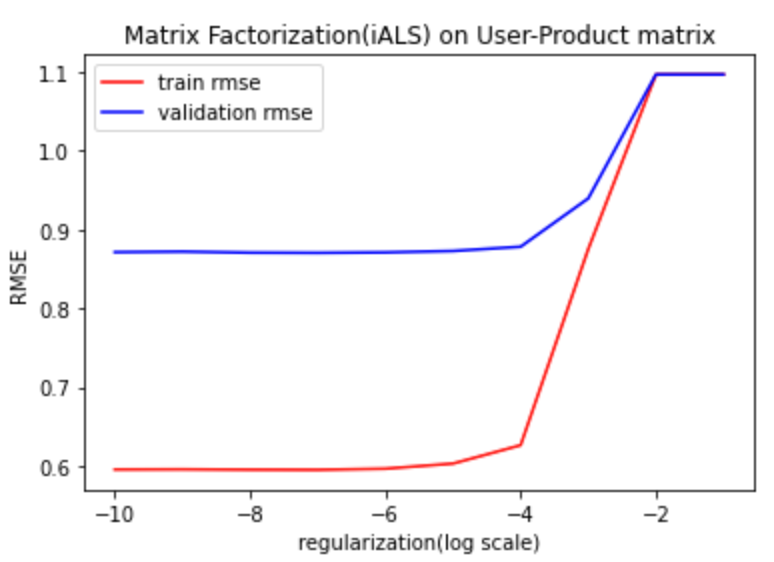
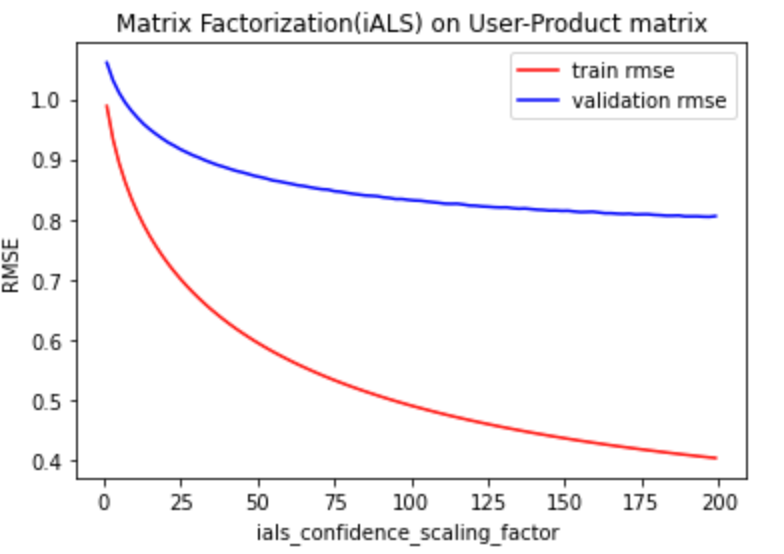
*Fig. 6. Precision@K for Matrix Factorization with iALS on User-Brand Pairs (Left) & User-Product Pairs (Right).*

**

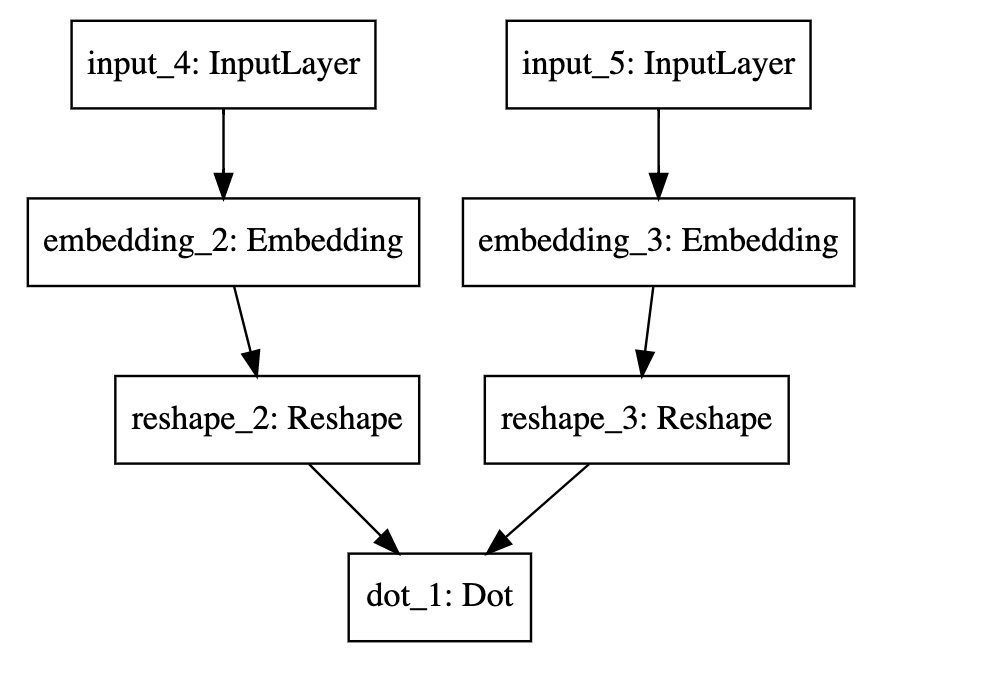
**

*Fig. 7. Hyper-Param Tuning for User-Brand Matrix*

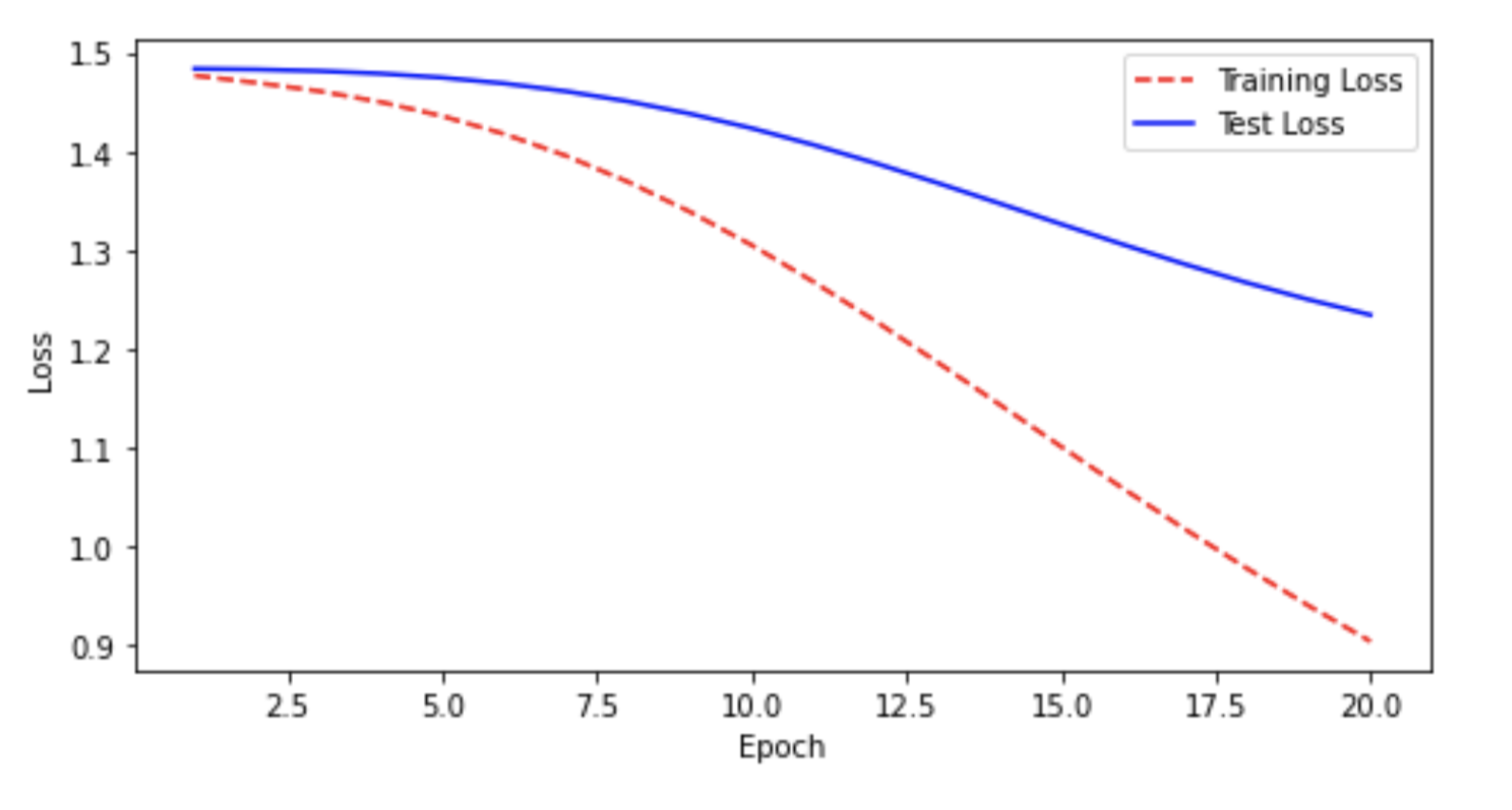
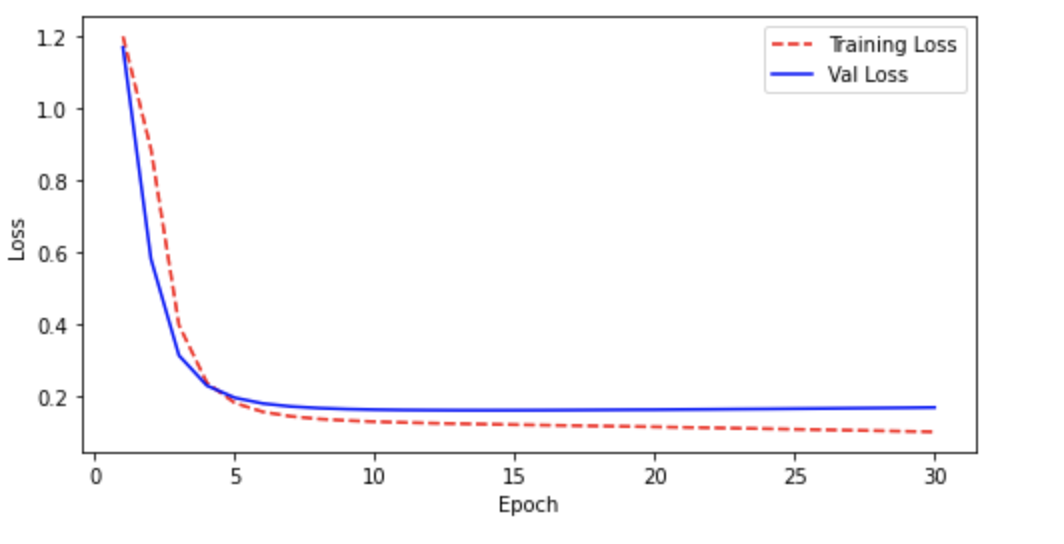
**

**

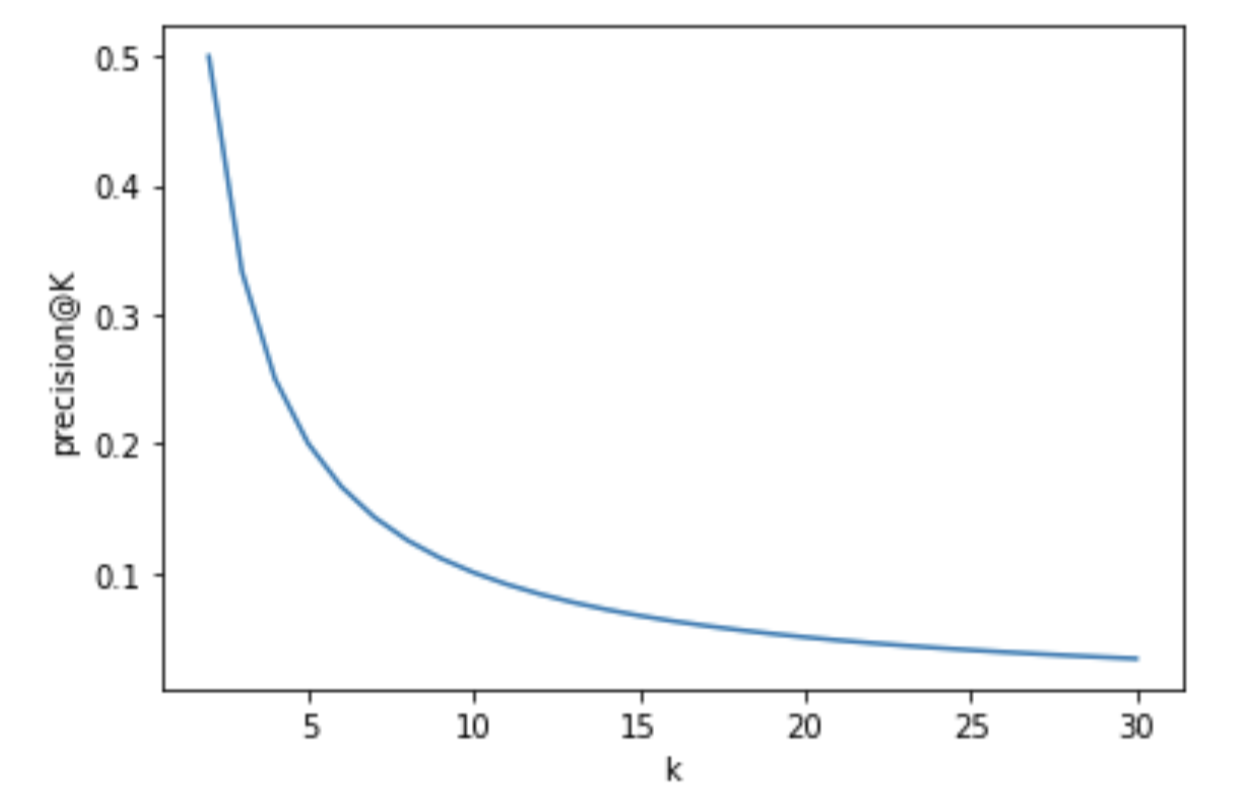
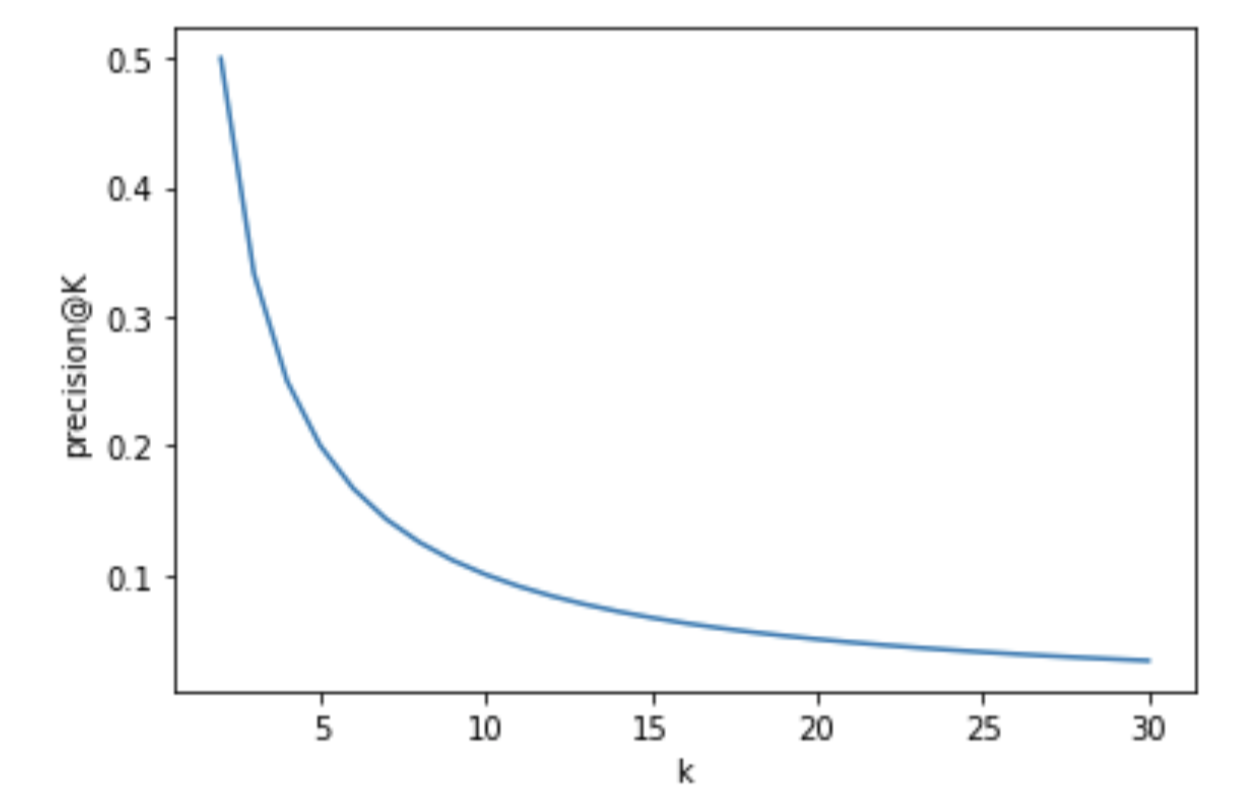
*Fig. 8. Hyper-Param Tuning for User-Product Matrix*



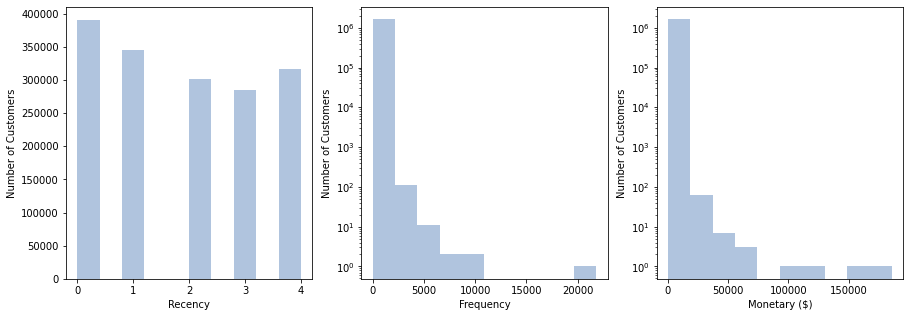
*Fig. 9. Structure of Neural Network*



*Fig. 10.Learning Curve for Loss v.s. Number of Epochs (Left: User-product; Right: User-brand)*

**

*Fig. 11. Learning Curve for Loss v.s. Number of Epochs (Left: User-product; Right: User-brand)*



*Fig. 12. RFM Analysis of Customers*